10. On-policy Control with Approximation

Notes

- In this chapter, we will look to find a parametric approximation of the action-value function $\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$. We will extend the semi-gradient TD(0) to the semi-gradient Sarsa algorithm by using action values and bringing it to on-policy control.
- The extension to a control problem is natural for an episodic cases, but will need need some re-examining for the continuing case.
- Episodic Semi-gradient Control:
  - We want approximate the action-value function $\hat{q} \approx q_\pi$, which is represented as a parameterised functional form with weight vector $\mathbf{w}$. Here, we consider training examples of $S_t, A_t \mapsto U_t$, where $U_t$is an approximation of $q_\pi(S_t, A_t)$, which can include full MC returns (G_t) or any of the n-step Sarsa returns. For example, the one step Sarsa method is

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

    which is called *episodic semi-gradient one-step Sarsa*. If the action set is discrete and not too large, we can get the greedy action via $A^*_{t+1} = \arg\max_a \hat{q}(S_{t+1}, a, \mathbf{w}_t)$. Pseudocode can be found on page 244.
- Semi-gradient n-step Sarsa:
  - The n-step return from tabular form can easily be generalised to a function approximation form of

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

    with $G_{t:t+n} \doteq G_t,\ t + n \geq T$. The n-step update equation is given by

    $$\mathbf{w_{t+n}} \doteq \mathbf{w_{t+n-1}} + \alpha[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w_{t+n-1}})]\nabla\hat{q}(S_t, A_t, \mathbf{w_{t+n-1}}).$$

    Pseudocode can be found in page 247.

- As in the tabular case, the performance is best if an intermediate level of bootstrapping is used.

- Average Reward: A new Problem Setting for Continuing Tasks:
  - The idea of *average reward* applies to the continuing case, but is undiscounted. This idea is useful for function approximation as the discounted setting is problematic for function approximation.
  - The quality of a policy π is defined as the average rate of reward, or *average reward*, while following that policy, and is given by

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

    Note that the second and third equation holds if the steady state distribution $\mu_\pi(s) \doteq \lim_{t\to\infty} \Pr\{S_t = s | A_{0:t-1} \sim \pi\}$ exists and is independent of S_0, in other words, *ergodic*. This also means that the effect of the start state and early decisions only have a temporary effect.
  - Note that the steady state distribution is the distribution under which if you select actions according to π, you remain in the same distribution:

$$\sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s'|s, a) = \mu_\pi(s')$$

  - In the average reward setting, returns are defined in terms of the difference between rewards and the average rewards:

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

    and this is known as the differential return. The corresponding value functions are the differential value functions. The Bellman equations for differential value functions are identical to the ones for normal value functions, but without the γs and with the rewards replaced by the difference between the reward and the average reward. The equations can be found in page 250.

    - One limitation here is that it does not converge to the differential values, but to the differential values plus an arbitrary offset.
    - Pseudocode can be found in page 251.
- Deprecating the Discounting Setting:
    - Consider a special case where all of the feature vectors are the same, and we can only assess the performance based on the reward sequence. Note that this is in the continuing case. Average reward can be used to judge the performance of the system.
    - We can try to use discounting, where we measure the discounted return at each time step. We would still need to average over a large time interval to judge the performance of the system. However, since we are averaging over the large time interval, each time step is exactly the same as another, and discounting doesn't affect the system where all states were weighted the same. In fact, the average of the discounted returns is always $r(\pi)/(1-\gamma)$.
    - The general argument is that if we optimised discounted value over the on-policy distribution, then the effect would be identical to optimising undiscounted average reward. The discount value γ has no effect.
    - Discounting algorithms with function approximation do not optimise discounted value over the on-policy distribution, and thus are not guaranteed to optimise average reward. This happens because we do not have the policy improvement theorem in function approximation any more.
    - Proof given in page 254.
- Differential Semi-gradient n-step Sarsa:
    - Here, the n-step TD error can be written as:

$$\begin{aligned}\delta_t &= G_{t:t+n}\hat{q}(S_t, A_t, \mathbf{w}) \\ G_{t:t+n} &= R_{t+1} - \bar{R}_{t+n-1} + \cdots + R_{t+n} + \bar{R}_{t+n-1} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1})\end{aligned}$$

      Pseudocode can be found in page 255.

Exercises

10.1 MC methods are generalised by the n-step Sarsa method, where in the MC case, the return is the actual return of the episode, where the rewards are summed till the

end of the episode ($G_{t:t+T}$). The pseudocode for MC methods will require an episode to play out till the end first, then the value estimates of each state visited in the episode will be computed in the reversed order of the episode. Given Figure 10.4, where the larger n gives a better performance than smaller values of n, so one might think that the MC method will give the best performance. However, we should also consider the computational and memory cost, where the MC method will be more computationally expensive, and it will not be able to learn in an online manner.

10.2 We can use the same pseudocode from page 247 for the episodic semi-gradient n-step Sarsa, but making a few tweaks to it. First, we would need to change the part in evaluating the return, where the term $\gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ should be replaced with $\gamma^n \sum_a \pi(a|S_{\tau+n})\hat{q}(S_{\tau+n}, a, \mathbf{w})$. In order to make it one step, we can just set $n = 1$.

10.3 In the first few episodes, there will be a high variance in the rewards received since the actions taken in theses first few episodes will generally be quite random. For a larger n, the n-step return will capture these high variances much more than than for lower values of n. ???

10.4 Recall that Q-learning uses the maximum value function for the update, where

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

for the discounted, tabular case.

To get the pseudocode for Q-learning, we use the same pseudoocde in page 251 for the differential semi-gradient Sarsa. However, we replace the fifth to last line of $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$ with $\delta \leftarrow R - \bar{R} + \max_a \hat{q}(S', a, \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$. We would also remove the line above it, since we are not using an ϵ-greedy policy to pick the next action.

10.5 We would need equation 10.11 and 10.12 to specify the differential version of TD(0).

10.6 In the case of reward of +1, 0, +1, 0, ..., the average reward is 0.5. The values of states A and B is given by

$$
\begin{aligned}
v_\pi(s) &\doteq \lim_{\gamma\to 1}\lim_{h\to\infty}\sum_{t=0}^{h}\gamma^t\big(\mathbb{E}_\pi[R_{t+1}|S_0=s]-r(\pi)\big)\\
v_\pi(A) &= \lim_{\gamma\to 1}\lim_{h\to\infty}\sum_{t=0}^{h}\gamma^t\frac{(-1)^t}{2}\\
&= \lim_{\gamma\to 1}\lim_{h\to\infty}\frac{1}{2}\frac{1-(-\gamma)^h}{1-(-\gamma)}\\
&= \frac{1}{2}\lim_{\gamma\to 1}\frac{1}{1+\gamma}\\
&= \frac{1}{4}
\end{aligned}
$$

$$
\begin{aligned}
v_\pi(B) &= \lim_{\gamma\to 1}\lim_{h\to\infty}\sum_{t=0}^{h}\gamma^t\frac{(-1)^{t+1}}{2}\\
&= -\lim_{\gamma\to 1}\lim_{h\to\infty}\frac{1}{2}\frac{1-(-\gamma)^h}{1-(-\gamma)}\\
&= -\frac{1}{2}\lim_{\gamma\to 1}\frac{1}{1+\gamma}\\
&= -\frac{1}{4}
\end{aligned}
$$

10.7 Here, the average reward is $\frac{1}{3}$. The values of each state are

$$
\begin{aligned}
v_\pi(A) &= \lim_{\gamma\to 1}\lim_{h\to\infty}\sum_{t=0}^{h}\gamma^{3t}\left[-\frac{1}{3}-\frac{1}{3}\gamma+\frac{2}{3}\gamma^2\right] \\
&= \lim_{\gamma\to 1}\lim_{h\to\infty}\frac{1-\gamma^{3h}}{1-\gamma^3}\left[-\frac{1}{3}-\frac{1}{3}\gamma+\frac{2}{3}\gamma^2\right] \\
&= \lim_{\gamma\to 1}\frac{2\gamma^2-\gamma-1}{3(1-\gamma^3)} \\
&= -\lim_{\gamma\to 1}\frac{(2\gamma+1)\cancel{(1-\gamma)}}{\cancel{3(1-\gamma)}(1+\gamma+\gamma^2)} \\
&= -\frac{1}{3}
\end{aligned}
$$

$$
\begin{aligned}
v_\pi(B) = &= \lim_{\gamma\to 1}\lim_{h\to\infty}\sum_{t=0}^{h}\gamma^{3t}\left[-\frac{1}{3}+\frac{2}{3}\gamma-\frac{1}{3}\gamma^2\right] \\
&= \lim_{\gamma\to 1}\lim_{h\to\infty}\frac{1-\gamma^{3h}}{1-\gamma^3}\left[-\frac{1}{3}+\frac{2}{3}\gamma-\frac{1}{3}\gamma^2\right] \\
&= \lim_{\gamma\to 1}\frac{\gamma^2-2\gamma+1}{3(\gamma^3-1)} \\
&= -\lim_{\gamma\to 1}\frac{(\gamma-1)^{\cancel{2}}}{\cancel{3(\gamma-1)}(1+\gamma+\gamma^2)} \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
v_\pi(C) &= \lim_{\gamma\to 1}\lim_{h\to\infty}\sum_{t=0}^{h}\gamma^{3t}\left[\frac{2}{3}-\frac{1}{3}\gamma-\frac{1}{3}\gamma^2\right] \\
&= \lim_{\gamma\to 1}\lim_{h\to\infty}\frac{1-\gamma^{3h}}{1-\gamma^3}\left[\frac{2}{3}-\frac{1}{3}\gamma-\frac{1}{3}\gamma^2\right] \\
&= \lim_{\gamma\to 1}\frac{\gamma^2+\gamma-2}{3(\gamma^3-1)} \\
&= -\lim_{\gamma\to 1}\frac{(\gamma+2)\cancel{(\gamma-1)}}{\cancel{3(\gamma-1)}(1+\gamma+\gamma^2)} \\
&= \frac{1}{3}
\end{aligned}
$$

10.8 With $R_t - \bar{R}_t$, the sequence starting with A will be $-\frac{1}{3}, -\frac{1}{3}, +\frac{2}{3}, -\frac{1}{3}, -\frac{1}{3}, +\frac{2}{3}, \ldots$. with δ_t, the sequence starting with A will be $\left(-\frac{1}{3}+\right.$

$0-(-\frac{1}{3})), (-\frac{1}{3}+\frac{1}{3}+0), (\frac{2}{3}+(-\frac{1}{3})-\frac{1}{3}), \cdots = 0, 0, 0, \ldots$.

10.9 The constant-step-size trick is to use step size of $\beta_n \doteq \beta/\bar{o}_n$, where $\bar{o} \doteq \bar{o}_{n-1} + \beta(1-\bar{o}_{n-1})$ for $n \geq 0$, with $\bar{o}_0 \doteq 0$. In the pseudocode, we would need to replace β with β_n, as well as add variables that track β_n and $\bar{o}_n$.