1 On-policy Control with Approximation

We consider attempts to solve the control problem using parametrised function approximation to estimate action-values. We consider only the on-policy case for now.

1.1 Episodic Semi-gradient Control

Extension of the semi-gradient update rules to action-values is straightforward

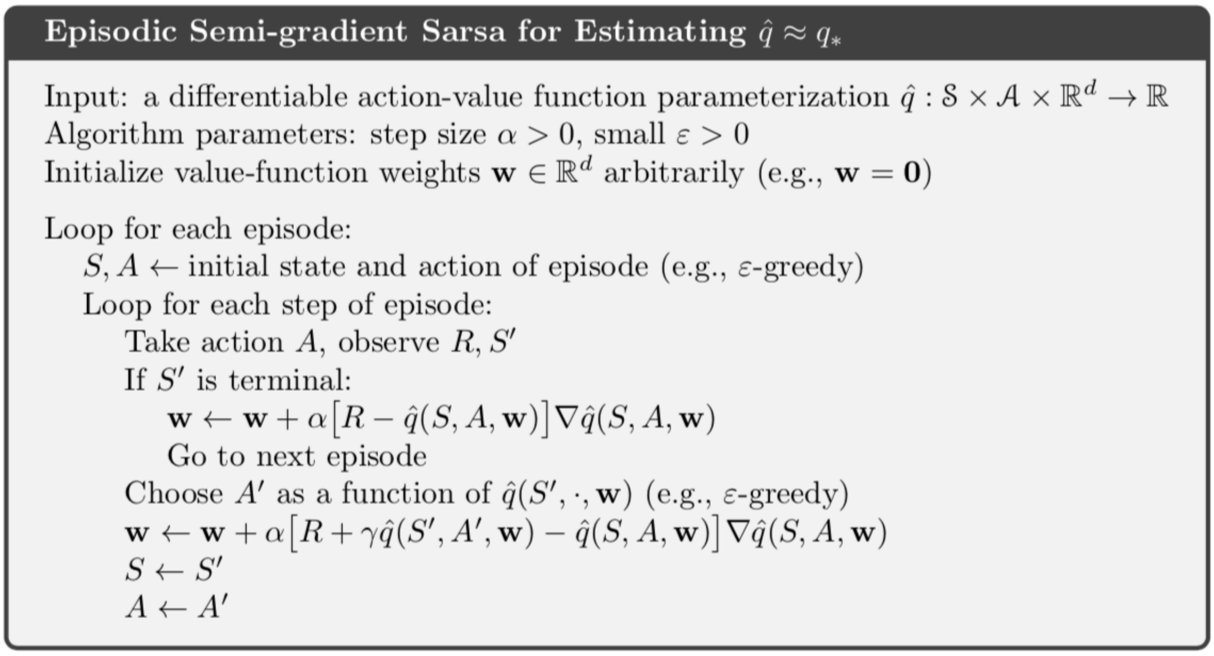
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\right] \nabla_{\mathbf{w}_t} \hat{q}(S_t, A_t, \mathbf{w}_t) \tag{1}$$

where U_t is the update target at time t. For example, one-step Sarsa has the update target is

$$U_t = R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t).$$

We call this method *episodic semi-gradient one-step Sarsa*. For a constant policy, this method converges in the same with as TD(0), with a similar kind of error bound.

In order to form control methods, we must couple the prediction ideas developed in the previous chapter with methods for policy improvement. Policy improvement methods for continuous actions or actions from large discrete spaces are an active area of research, with no clear resolution. For actions drawn from smaller discrete sets, we can use the same idea as we have before, which is to compute action values and then take an ε-greedy action selection. Episodic semi-gradient sarsa can be used to estimate the optimal action-values as in the box below.



Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

```
Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$
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., $\varepsilon$-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., $\varepsilon$-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$
```

1.2 Semi-gradient n-step Sarsa

We can use an n-step version of the episodic Sarsa that we defined above by incorporating the bootstrapped n-step return

$$G_{t:t+n} \doteq \sum_{i=1}^{n-1} \gamma^i R_{i+1} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, W_{t+n-1}) \tag{2}$$

where $G_{t:t+n} = G_t$ if $t + n \geq T$, as usual. This update target is used in the pseudocode in the box below. As we have seen before, performance is generally best with amn intermediate value of n.

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 $\pi$ (if estimating $q_\pi$)
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 $\varepsilon$-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 $\varepsilon$-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
  |  $\tau \leftarrow t - n + 1$    ($\tau$ 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 $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$          $(G_{\tau:\tau+n})$
  |      $\mathbf{w} \leftarrow \mathbf{w} + \alpha [G - \hat{q}(S_\tau, A_\tau, \mathbf{w})] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$
```

1.3 Average Reward: A New Problem Setting for Continuing Tasks

We introduce a third classical setting for formulating the goal in Markov decision problems (MDPs) (to go along with episodic and continuing). This new setting is called the *average reward setting*. This setting applies to continuing problems with no start or end state, but also no discounting. (Later we will see that the lack of a start state introduces a symmetry that makes discounting with function approximation pointless.)

In the average reward setting, the ordering of policies is (most often) defined with respect to the *average reward* while following the policy

$$r(\pi) \doteq \lim_{h \to \infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] \tag{3}$$

$$= \lim_{t \to \infty} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] \tag{4}$$

$$= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) r. \tag{5}$$

We will consider policies that attain the maximal value of $r(\pi)$ to be optimal (though there are apparently some subtly distinctions here that are not gone into).

The distribution $\mu_\pi(s)$ is the steady-state distribution defined by

$$\mu_\pi(s) \doteq \lim_{t\to\infty} \mathbb{P}(S_t = s | A_{0:t-1} \sim \pi) \tag{6}$$

which we assume to exist for any π and to be independent of the starting state S_0. This assumption is known as *ergodicity*, and it means that the long run expectation of being in a state depends only on the policy and MDP transition probabilities – not on the start state. The steady-state distribution has the property that it is invariant under actions taken by π, in the sense that the following holds

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

In the average-reward setting we define returns in terms of the difference between the reward and the expected reward for the policy

$$G_t \doteq \sum_{i\geq t} (R_{i+1} - r(\pi)) \tag{7}$$

we call this quantity the *differential return* and the corresponding value functions (defined in the same way, just with this return instead) *differential value functions*. These new value functions also have Bellman equations:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r - r(\pi) + v_\pi(s')\right] \tag{8}$$

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s',a')\right] \tag{9}$$

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a) \left[r - r(\pi) + v_*(s')\right] \tag{10}$$

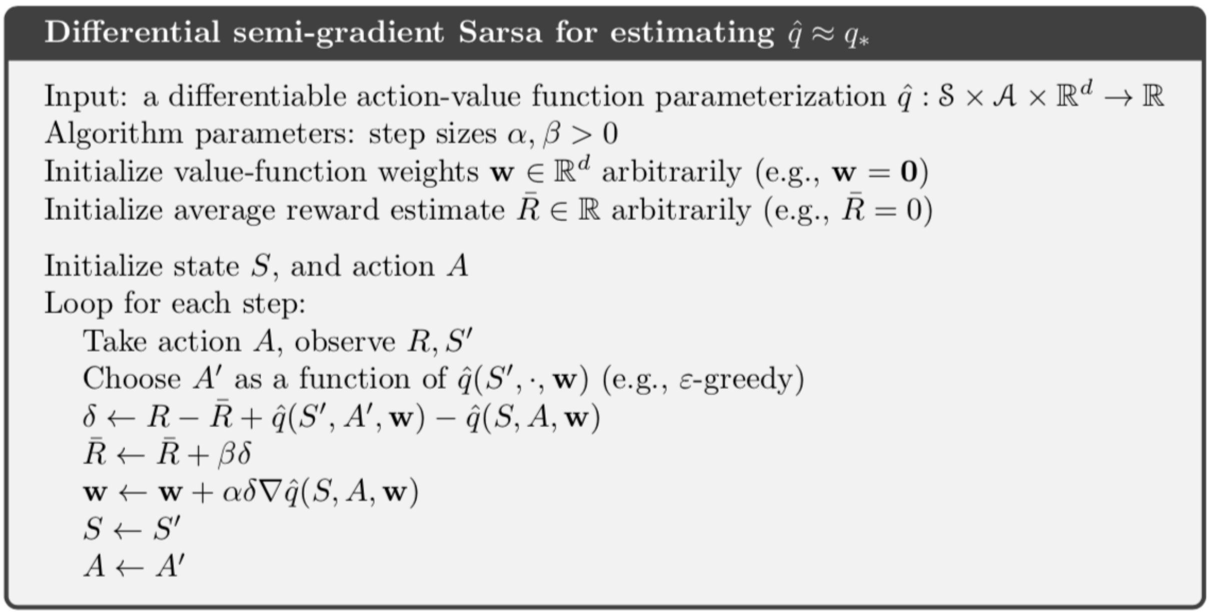
$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left[r - r(\pi) + \max_{a'} q_*(s',a')\right]. \tag{11}$$

We also have differential forms of the TD errors, where $\bar{R}_t$ is the estimate of $r(\pi)$ at t,

$$\delta_t \doteq R_{t+1} - \bar{R}_{t+1} + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t) \tag{12}$$

$$\delta_t \doteq R_{t+1} - \bar{R}_{t+1} + \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t). \tag{13}$$

Many of the previous algorithms and theoretical results carry over to this new setting without change. For instance, the update for the semi-gradient Sarsa is defined in the same way just with the new TD error, corresponding pseudocode given in the box below.



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 $\alpha, \beta > 0$
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., $\varepsilon$-greedy)
    $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$
```

1.4 Deprecating the Discounted Setting

Suppose we want to optimise the discounted value function $v_\pi^\gamma(s)$ over the on-policy distribution, we would choose an objective $J(\pi)$ with

$$\begin{aligned}
J(\pi) &\doteq \sum_s \mu_\pi(s) v_\pi^\gamma(s) && (14)\\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi^\gamma(s')\right] && (15)\\
&= r(\pi) + \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \gamma v_\pi^\gamma(s') && (16)\\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s', |s, a) && (17)\\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \mu_\pi(s') && (18)\\
&= r(\pi) + \gamma J(\pi) && (19)\\
&\vdots && (20)\\
&= \frac{1}{1-\gamma} r(\pi) && (21)
\end{aligned}$$

so we may as well have optimised for the *undiscounted* average reward.

The root cause (note: why *root* cause?) of the difficulties with the discounted control setting is that when we introduce function approximation we lose the policy improvement theorem. This is because when we change the discounted value of one state, we are not guaranteed to have improved the policy in any useful sense (e.g. generalisation could ruin the policy elsewhere). This is an area of open research.

1.5 Differential Semi-gradient n-step Sarsa

We generalise n-step bootstrapping by introducing an n-step version of the TD error in this new setting. In order to do that, we first introduce the differential n-step return using function approximation

$$G_{t:t+n} \doteq \sum_{i=t}^{n-1} \big(R_{i+1} - \bar{R}_{i+1}\big) + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}) \tag{22}$$

with $G_{t:t+n} = G_t$ if $t + n \geq T$ as usual and where $\bar{R}_i$ are the estimates of $\bar{R}$. The n-step TD error is then defined as before just with the new n-step return

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

Pseudocode for the use of this return in the Sarsa framework is given in the box below. Note that $\bar{R}$ is updated using the TD error rather than the latest reward (see Exercise 10.9).

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 $\pi$
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 $\varepsilon$-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    $\tau \leftarrow t - n + 1$    ($\tau$ is the time whose estimate is being updated)
    If $\tau \geq 0$:
        $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n}(R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
        $\bar{R} \leftarrow \bar{R} + \beta\delta$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$
```