Lecture 9: Control with Function Approximation

Tianpei Xie

Aug 12th., 2022

Contents

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

Figure 1: The episodic semi-gradient Sarsa

1 Introduction

Recall that in Tabular methods, the control problem is solved by estimation of action-value fucntion $q_\pi(s, a)$. We have on-policy and off-policy control updates:

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha_t \left[\hat{G}_t - Q(S_t, A_t)\right]. \\
\textbf{Sarsa} \quad \hat{G}_t &= R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \\
\textbf{Q-Learning} \quad \hat{G}_t &= R_{t+1} + \gamma \max_{a' \in \mathcal{A}(S_{t+1})} Q(S_{t+1}, a') \\
\textbf{Expected Sarsa} \quad \hat{G}_t &= R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1}) Q(S_{t+1}, a')
\end{aligned}
$$

Like in on-policy prediction with approximation, we use **parameterized** function to approximate action-value $q_\pi(s, a) \approx \hat{q}(\boldsymbol{s}, a, \boldsymbol{w})$.

Also recall that when we parameterize the value function in prediction task, the update rule is modified: for instance in TD(0)/Semi-gradient,

$$
\begin{aligned}
\text{(taular method) } V(S_t) &\leftarrow V(S_t) + \alpha_t \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \\
\text{(fun. approx.) } \boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t + \alpha \left[R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}_t)\right] \nabla_{\boldsymbol{w}} \hat{v}(S_t, \boldsymbol{w}_t)
\end{aligned}
$$

That is, the function V is replaced by its parameter representation $\boldsymbol{w}$ and also the *error* term is multiplied by the *gradient* term $\nabla_{\boldsymbol{w}} \hat{v}(S_t, \boldsymbol{w}_t)$. We can see that after the similar modification, the control updates can still be applied.

2 Episodic Semi-gradient Control

Based on similar derivation and using Tabular Sarsa as reference, we have the following control updates under action-value function approximation $\hat{q}(\boldsymbol{s}, a, \boldsymbol{w})$:

$$\text{(SGD) } \boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha \left[\hat{G}_t - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\right] \nabla_{\boldsymbol{w}} \hat{q}(S_t, A_t, \boldsymbol{w}_t) \tag{1}$$

(**semi-gradient (SARSA)**)

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha \left[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\right] \nabla_{\boldsymbol{w}} \hat{q}(S_t, A_t, \boldsymbol{w}_t) \tag{2}$$

We call this method in (2) ***episodic semi-gradient one-step Sarsa***. In SGD (1), we assume that we have access to a set of input-output pairs $((S_t, A_t), \hat{G}_t)$, where input is the sample state-action pair (S_t, A_t). In semi-gradient, the output is replaced by reward plus previous estimates $R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t)$. For a constant policy, this method converges in the same way that TD(0) does, with the same kind of error bound.

To form control methods, we need to couple such action-value prediction methods with techniques for policy improvement and action selection. For large continuous actions, or to actions from large discrete sets, this is a challenging task and an ongoing research topic. For action set that is **discrete** and **not too large**, we can use ϵ-greedy algorithm to find the optimal action $A_t^* = \text{argmax}_a \, \hat{q}(S_t, a, \boldsymbol{w}_t)$ with probability $1 - \epsilon$. Figure 1 shows the episodic semi-gradient SARSA for on-policy control. This is essentially the same as Tabular SARSA except that replacing the value function updates with the updates on its parameters.

Similar to SARSA, we can construct the update rule for off-policy **Q-learning** and **Expected SARSA** for function approximation based on its tabular form:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha \, \delta_t \, \nabla_{\boldsymbol{w}} \hat{q}(S_t, A_t, \boldsymbol{w}_t)$$

$$\textbf{Q-Learning} \quad \delta_t := \left[R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\right] \tag{3}$$

$$\textbf{Expected Sarsa} \quad \delta_t = \left[R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1}) \hat{q}(S_{t+1}, a', \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)\right] \tag{4}$$

Similar to SARSA, these algorithms apply to discrete and small action space, since it requires a complete scan of all possible actions at given state. Replacing (2) with (3) or (4) in Figure 1, we have complete description of **semi-gradient off-control Q-learning** and **Expected SARSA**.

Like prediction task, we can use deep neural network to estimate the action-value function Q such as Deep Q-Network. For more discussion on Deep RL, see in [François-Lavet et al., 2018]

3 Average Rewards as New Problem Setting for Continuing Tasks

The goal for reinforcement learning is to maximize the cumulative rewards or returns. By far, our definitions of returns in episodic tasks and continuing tasks are focusing on cumulation i.e. summation. This section, we introduce a new concept for continuing task, the average rewards.

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!

Figure 2: Use of discount factor in continuing task is futile

- **Returns for episodic task**: In episodic task, the task is terminated at some time T. The return is a finite number

$$G_t = \sum_{\tau=0}^{T-t-1} R_{\tau+t+1} < \infty \tag{5}$$

- **Discounted returns for continuing task**: In continuing task, there is no termination and the natural sum is infinite thus improper. In order to obtain finite return, we need to add discount factor $\gamma \in [0, 1)$

$$G_t = \sum_{\tau=0}^{\infty} \gamma^\tau R_{\tau+t+1} < \infty \tag{6}$$

  Note that the choice of γ has significant impact on the behavior of learning agents. When $\gamma \to 1$, the agent emphasize the long-term gain over shot-term gain. When $\gamma \to 0$, it does the opposite.

  Unlike the tabular cases, in which the returns from each state can be separately identified and averaged, using the discounted returns for function approximation in counting tasks is **questionable**, esp. when the MDP is ergodic and has become **stationary** in the long run.

  Consider an infinite sequence of returns with no beginning or end, and no clearly identified states. The states might be represented only by feature vectors, which may do little to distinguish the states from each other. As a special case, *all of the feature vectors may be the same*. Thus one really has only the **reward sequence** (and the actions), and performance has to be assessed purely from these. In particular, the *ordering* of all policies in the average discounted return setting would be exactly **the same** as in the *average-reward* setting. When the MDP for contining task reach to **steady state** (i.e. the maginal distribution of each state is not changing over time), due to transition-symmetry, **each time step is exactly the same as every other**. You cannot tell where it is since there is no beginning and no end for a sequence of returns $(\ldots, G_t, G_{t+1}, \ldots)$. With discounting, every reward will appear exactly once in each position in some return. (i.e. R_t appears in every $G_{t-1}, G_{t-2}, \ldots,$). The weight on the t-th reward is thus $\sum_{\tau=0}^{\infty} \gamma^\tau = \frac{1}{1-\gamma}$.

  In fact, we can show that in ergodic continuing task, because all states are the same, they are all weighted by this, and thus the *average* of the returns will be the asympototic discount factor $\frac{1}{1-\gamma}$ times the **average reward**, or $r(\pi)/(1-\gamma)$. Figure 2 shows a proof of this.

  The root cause of the difficulties with the **discounted control** setting is that with **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 in any useful sense. In fact, the lack of a policy improvement theorem is also a theoretical lacuna for the total-episodic and average-reward settings. Once we introduce function approximation we can no longer guarantee improvement for any setting.

- **Average reward and Differential returns for continuing task**: Unlike discounted returns, average reward **does not** have *discount factor* γ. In the average-reward setting, the quality of a policy π is defined as the ***average rate of reward***, or simply **average reward**,

while following that policy, which we denote as $r(\pi)$:

$$\begin{aligned} r(\pi) &= \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_\pi \left[G_{0:T}\middle|S_0, A_{0:t-1}\sim\pi\right] \\ &= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}_\pi \left[R_t\middle|S_0, A_{0:t-1}\sim\pi\right] \quad (7) \\ &= \lim_{t\to\infty} \mathbb{E}_\pi \left[R_t\middle|S_0, A_{0:t-1}\sim\pi\right] \quad (8) \\ &= \sum_s \mu_\pi(s)\sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)r, \quad (9) \end{aligned}$$

where the expectations are conditioned on the initial state, S_0, and on the subsequent actions, $A_0, A_1, \ldots, A_{t-1}$, being taken according to π. $\mu_\pi(s)$ is the **steady-state distribution**.

$$\mu_\pi(s) = \lim_{t\to\infty} \Pr\{S_t = s|A_{0:t-1}\sim\pi\},$$

which is assumed to exist for any π and to be **independent** of S_0. This assumption about the MDP is known as **ergodicity** In other words, it relies on there being a long term stable distribution of states under any particular policy. The defintion of average rewards for continuing tasks can only be applied when the underling MDP is **ergodic**. From (8) we can see that **the average reward is the reward obtained at when MDP reaches the stationary state**.

We consider $\pi_* = \text{argmax}_\pi\, r(\pi)$ as the **optimal policy**, which is the basis for **policy-based learning** algorithm.

Moreover, according to MDP, the stationary distribution is the **fixed-point solution** the equation below:

$$\mu_\pi(s') = \sum_s \mu_\pi(s)\left(\sum_a \pi(a|s)\sum_{s'} p(s'|s,a)\right) \quad (10)$$

With the average rewards, we can define the the **differential return** as the cumulative *difference* between rewards and the average reward:

$$G_t = \sum_{\tau=0}^{\infty} (R_{\tau+t+1} - r(\pi)) < \infty \quad (11)$$

Replacing (6) or (5) with (11), we can obtain the definition of ***differential value function*** of expected differential returns given state or state-action pair: $v_\pi(s) := \mathbb{E}_\pi\left[G_t|S_t = s\right]$ and $q_\pi(s,a) := \mathbb{E}_\pi\left[G_t|S_t = s, A_t = a\right]$. The differential returns measures how well our policy behaves as compared to the average reward under a **fixed policy**. We can also reformulate the **Bellman equation** and **Bellman optimality equation** for **average reward MDP**

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

Figure 3: Differential semi-gradient SARSA

by removing γ and replacing reward r with differential reward $r - r(\pi)$:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r - r(\pi) + v_\pi(s')\right], \quad \forall\ s \in \mathcal{S} \tag{12}$$

$$= \mathbb{E}_\pi \left[R_{t+1} - r(\pi) + v_\pi(S_{t+1}) | S_t = s\right] \tag{13}$$

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a) \left[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a')\right] \quad \forall\ s \in \mathcal{S},\ a \in \mathcal{A}(s) \tag{14}$$

$$= \mathbb{E}_\pi \left[R_{t+1} - r(\pi) + q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right] \tag{15}$$

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a) \left[r - \max_\pi r(\pi) + v_*(s')\right] \tag{16}$$

$$= \max_{a \in \mathcal{A}(s)} \mathbb{E} \left[R_{t+1} - \max_\pi r(\pi) + v_*(S_{t+1}) | S_t = s, A_t = a\right] \tag{17}$$

$$q_*(s, a) = \sum_{s'} \sum_r p(s', r|s, a) \left[r - \max_\pi r(\pi) + \max_{a' \in \mathcal{A}(s)} q_*(s', a')\right] \tag{18}$$

$$= \mathbb{E} \left[R_{t+1} - \max_\pi r(\pi) + \max_{a' \in \mathcal{A}(s)} q_*(S_{t+1}, a') \middle| S_t = s, A_t = a\right] \tag{19}$$

Note that in Bellman optimality equation for differential returns, we also need to choose **optimal average rewards** $\max_\pi r(\pi)$ instead of using average reward itself since the optimal value function is not a function of policy π. Based on these Bellman equations, we can derive the ***Differential TD error*** for differential returns:

$$\textbf{Sarsa} \quad \delta_t = R_{t+1} - \bar{R}_t + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \tag{20}$$

$$\textbf{Q-Learning} \quad \delta_t = R_{t+1} - \bar{R}_t + \max_{a' \in \mathcal{A}(S_{t+1})} Q(S_{t+1}, a') - Q(S_t, A_t) \tag{21}$$

$$\textbf{Expected Sarsa} \quad \delta_t = R_{t+1} - \bar{R}_t + \sum_{a'} \pi(a'|S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t) \tag{22}$$

Note that instead of computing $r(\pi)$ which requires knowledge of model $p(s'|s, a)$, we use

the sample average reward $\bar{R}_t$ as estimate, which is updated by accumulation of TD error $\bar{R}_{t+1} = \bar{R}_t + \beta\,\delta_t = \beta \sum_t \delta_t$. Figure 3 shows the **differential semi-gradient SARSA**.

Exercise 3.1 *Proof of Bellman equation* (13) *for average rewards MDP.*

Solution:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi\left[G_t | S_t = s\right] \\
&= \mathbb{E}_\pi\left[\sum_{\tau=t+1}^{\infty} \left(R_\tau - r(\pi)\right) \Big| S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} - r(\pi) + \sum_{\tau=t+2}^{\infty} \left(R_\tau - r(\pi)\right) \Big| S_t = s\right] \\
&= \mathbb{E}\left[\mathbb{E}_\pi\left[R_{t+1} - r(\pi) + \sum_{\tau=t+2}^{\infty} \left(R_\tau - r(\pi)\right) \Big| S_{t+1}, S_t = s\right] \Big| S_t = s\right] \\
&= \mathbb{E}\left[\mathbb{E}_\pi\left[R_{t+1} - r(\pi) + G_{t+1} \Big| S_{t+1}, S_t = s\right] \Big| S_t = s\right] \\
&\text{(constant since policy fixed)} \\
&= \mathbb{E}\left[R_{t+1} - r(\pi) + \mathbb{E}_\pi\left[G_{t+1} \Big| S_{t+1}, S_t = s\right] \Big| S_t = s\right] \\
&\text{(by Markov Property)} \\
&= \mathbb{E}\left[R_{t+1} - r(\pi) + \mathbb{E}_\pi\left[G_{t+1} \Big| S_{t+1}\right] \Big| S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} - r(\pi) + v_\pi(S_{t+1}) \Big| S_t = s\right]. \quad \blacksquare
\end{aligned}
$$

Exercise 3.2 *Proof of Bellman optimality equation* (17) *for average rewards MDP.*

Solution:

$$
\begin{aligned}
v_*(s) &:= \max_{a \in \mathcal{A}(s)} \max_\pi \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \\
&= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a) \\
&\text{(the optimal policy } \pi_* = \operatorname{argmax}_\pi \mathbb{E}_\pi\left[G_t | S_t\right] \text{ is chosen)} \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}_\pi\left[\sum_{\tau=t+1}^{\infty} \left(R_\tau - r(\pi_*)\right) \Big| S_t = s, A_t = a\right] \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}_\pi\left[R_{t+1} - r(\pi_*) + \sum_{\tau=t+2}^{\infty} \left(R_\tau - r(\pi_*)\right) \Big| S_t = s, A_t = a\right] \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}_\pi\left[\mathbb{E}\left[R_{t+1} - r(\pi_*) + \sum_{\tau=t+2}^{\infty} \left(R_\tau - r(\pi_*)\right) \Big| S_{t+1}, S_t = s, A_t = a\right] \Big| S_t = s, A_t = a\right] \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}\left[R_{t+1} - r(\pi_*) + \mathbb{E}_\pi\left[\sum_{\tau=t+2}^{\infty} \left(R_\tau - r(\pi_*)\right) \Big| S_{t+1}\right] \Big| S_t = s, A_t = a\right] \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}\left[R_{t+1} - r(\pi_*) + \max_\pi \mathbb{E}_\pi\left[G_{t+1} | S_{t+1}\right] \Big| S_t = s, A_t = a\right] \\
&\text{(the optimal policy } \pi_* = \operatorname{argmax}_\pi \mathbb{E}_\pi\left[G_{t+1} | S_{t+1}\right] = \operatorname{argmax}_\pi r(\pi)) \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}\left[R_{t+1} - \max_\pi r(\pi) + v_*(S_{t+1}) \Big| S_t = s, A_t = a\right]. \quad \blacksquare
\end{aligned}
$$

References

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.