Lecture 10: Policy Gradient Methods

Tianpei Xie

Aug 13th., 2022

Contents

1 Value-based methods vs. Policy-based methods

By far, we mainly discussed the value-based methods, this chapter, we focus on policy-based methods.

- **Value-based methods** (or *action-value methods*): all these methods (DP, MC, TD with tabular or function approximation) learned the **values of actions** and then selected actions based on their estimated action values; their policies would not even exist without the action-value estimates.

- **Policy-based methods** (or *policy gradient methods*): methods that instead learn a **parameterized policy** that can select actions *without consulting* a value function. A *value* function may still be used to *learn the policy parameter*, but is not required for action selection.

  Perhaps the simplest **advantage** that **policy parameterization** may have over **action-value parameterization** is that the policy may be a simpler function to approximate. Problems vary in the complexity of their policies and action-value functions.

  Finally, we note that the choice of policy parameterization is sometimes a good way of **injecting prior knowledge** about the desired form of the policy into the reinforcement learning system. This is often the most important reason for using a policy-based learning method.

Denote $\boldsymbol{\theta} \in \mathbb{R}^m$, the parameterized policy distribution over $a \in \mathcal{A}(s)$ at time t is $\pi(a|\boldsymbol{s}, \boldsymbol{\theta}) := Pr\{A_t = a|S_t = \boldsymbol{s}, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$. The *goal* of policy optimization is to find policy π that ***maximizes*** some **objective function** $\mathcal{R}(\boldsymbol{\theta}_t)$. For instance, for continuing tasks with ergodic MDP, we choose **average rewards** $\mathcal{R}(\boldsymbol{\theta}_t) := r(\pi(a|\boldsymbol{s}, \boldsymbol{\theta}))$

$$r(\pi(a|\boldsymbol{s}, \boldsymbol{\theta})) = \sum_s \mu_\pi(s) \sum_a \pi(a|s, \boldsymbol{\theta}) \sum_{s',r} p(s', r|s, a)r. \tag{1}$$

1.1 Policy Gradient methods

In order to optimize the objective function (1), we use the *gradient acsent algorithm*

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} \widehat{\mathcal{R}(\boldsymbol{\theta}_t)}. \tag{2}$$

All methods that follow the general schema (2) we call ***policy gradient methods***, whether or not they also learn an approximate value function. Methods that learn approximations to *both policy and value functions* are often called ***actor-critic methods***, where '***actor***' is a reference to the learned policy, and '***critic***' refers to the learned value function, usually a state-value function.

2 Policy approximation

Depending on if the action space $\mathcal{A}$ is finite discrete or continous, we can approximate the paremeterized policy distribution using different functions:

- When $\mathcal{A}$ is **finite discrete**, and $\boldsymbol{s} \in \mathcal{S}$, we can ues the **soft-max function** to approximate the policy function

$$
\begin{aligned}
\pi(a|\boldsymbol{s},\boldsymbol{\theta}) &= \frac{\exp\left(h(\boldsymbol{s},a,\boldsymbol{\theta})\right)}{\sum_{a'}\exp\left(h(\boldsymbol{s},a',\boldsymbol{\theta})\right)} && (3)\\
\log\pi(a|\boldsymbol{s},\boldsymbol{\theta}) &= h(\boldsymbol{s},a,\boldsymbol{\theta}) - \underline{\log\sum_{a'}\exp\big(h(\boldsymbol{s},a',\boldsymbol{\theta})\big)} && (4)\\
\nabla_{\boldsymbol{\theta}}\log\pi(a|\boldsymbol{s},\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a,\boldsymbol{\theta}) - \underline{\mathbb{E}_{\pi(a|\boldsymbol{s},\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a,\boldsymbol{\theta})\right]} && (5)\\
&= \nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a,\boldsymbol{\theta}) - \sum_{a'}\pi(a'|\boldsymbol{s},\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a',\boldsymbol{\theta})\\
&= \textbf{sample } \nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a,\boldsymbol{\theta}) \text{ with action} - \textbf{policy action mean} \text{ of } \nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a,\boldsymbol{\theta})
\end{aligned}
$$

The function $h(\boldsymbol{s},a,\boldsymbol{\theta})$ defines the **preferences** of some actions. This kind of policy parameterization is called ***soft-max in action preferences***. We can define the preference as *linear functions* with respect to some features:

$$
\begin{aligned}
h(\boldsymbol{s},a,\boldsymbol{\theta}) &:= \langle\boldsymbol{\theta}\,,\,\boldsymbol{\phi}(\boldsymbol{s},a)\rangle\\
\nabla_{\boldsymbol{\theta}}h(\boldsymbol{s},a,\boldsymbol{\theta}) &= \boldsymbol{\phi}(\boldsymbol{s},a)
\end{aligned}
$$

where $\boldsymbol{\phi}(\boldsymbol{s},a) := [\phi_k(\boldsymbol{s},a)]$ can be any feature representation we discussed in lecture 8, for instance, *tile coding* on state on fixed action and stacked multiple actions. Combining with (5), we can see that the gradient of log of π can be easily represented as the **difference** between the ***sample (mean)*** *of state-action feature* for specific action $\boldsymbol{\phi}(\boldsymbol{s},a)$ and *the* ***policy mean*** *of state-action feature over all actions* $\mathbb{E}_{\pi(a|\boldsymbol{s},\boldsymbol{\theta})}\left[\boldsymbol{\phi}(\boldsymbol{s},a)\right]$.

- One **advantage** of parameterizing policies according to the soft-max in action preferences is that the **approximate policy can approach a deterministic policy**, whereas with ϵ-greedy action selection over action values there is always an ϵ probability of selecting a random action. Of course, one could ***select*** *according to a* ***soft-max distribution*** based on action values, but this alone would not allow the policy to approach a **deterministic policy**. Instead, the **action-value estimates** would converge to their corresponding **true values**, which would *differ by a finite amount*, translating to specific probabilities other than 0 and 1. **Action preferences** are different because they do not approach specific values; instead they are driven to produce the ***optimal stochastic policy***. If the optimal policy is ***deterministic***, then the preferences of the optimal actions will be driven infinitely higher than all suboptimal actions (if permitted by the parameterization).

- A second **advantage** of parameterizing policies according to the soft-max in action preferences is that it enables the selection of actions with **arbitrary probabilities**. In problems with significant function approximation, the **best approximate policy** may be **stochastic**. For example, in card games with imperfect information the optimal play is often to do two different things with specific probabilities, such as when bluffng in Poker. Action-value methods have no natural way of finding stochastic optimal policies, whereas policy approximating methods can.

- When $\mathcal{A} \subset \mathbb{R}^k$ is **continuous space**, and $\boldsymbol{s} \in \mathcal{S}$, $\boldsymbol{\theta} = [\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma]$

$$\pi(a|\boldsymbol{s}, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)} \exp\left(-\frac{(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2}{2\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2}\right) \tag{6}$$

$$\log \pi(a|\boldsymbol{s}, \boldsymbol{\theta}) = -\frac{(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2}{2\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2} - \log \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma) + \frac{1}{2}\log 2\pi \tag{7}$$

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log \pi(a|\boldsymbol{s}, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} h(\boldsymbol{s}, a, \boldsymbol{\theta}) - \underline{\mathbb{E}_{\pi(a|\boldsymbol{s},\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} h(\boldsymbol{s}, a, \boldsymbol{\theta})\right]} \qquad (8) \\ &= \textbf{sample of } \nabla_{\boldsymbol{\theta}} h(\boldsymbol{s}, a, \boldsymbol{\theta}) - \textbf{policy action mean of } \nabla_{\boldsymbol{\theta}} h(\boldsymbol{s}, a, \boldsymbol{\theta}) \end{aligned}$$

where

$$\begin{aligned} h(\boldsymbol{s}, a, \boldsymbol{\theta}) &= -\frac{(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2}{2\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2} \\ \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu) &= \left\langle \boldsymbol{\theta}_\mu \,,\, \boldsymbol{\phi}_\mu(\boldsymbol{s}) \right\rangle \\ \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma) &= \exp(\langle \boldsymbol{\theta}_\sigma \,,\, \boldsymbol{\phi}_\sigma(\boldsymbol{s}) \rangle) \end{aligned}$$

Note that

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_\mu} \log \pi(a|\boldsymbol{s}, \boldsymbol{\theta}) &= \frac{1}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2} \left(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu)\right) \nabla_{\boldsymbol{\theta}_\mu} \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu) \\ &= \frac{1}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2} \left(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu)\right) \boldsymbol{\phi}_\mu(\boldsymbol{s}) \\ \nabla_{\boldsymbol{\theta}_\sigma} \log \pi(a|\boldsymbol{s}, \boldsymbol{\theta}) &= (a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2 \frac{1}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^3} \nabla_{\boldsymbol{\theta}_\sigma} \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma) - \frac{1}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)} \nabla_{\boldsymbol{\theta}_\sigma} \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma) \\ &= \left(\frac{(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2 - \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^3}\right) \nabla_{\boldsymbol{\theta}_\sigma} \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma) \\ &= \left(\frac{(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2 - \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^3}\right) \sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma) \boldsymbol{\phi}_\sigma(\boldsymbol{s}) \\ &= \left(\frac{(a - \mu(\boldsymbol{s}, \boldsymbol{\theta}_\mu))^2}{\sigma(\boldsymbol{s}, \boldsymbol{\theta}_\sigma)^2} - 1\right) \boldsymbol{\phi}_\sigma(\boldsymbol{s}) \end{aligned}$$

3 The Policy Gradient Theorem

With continuous policy parameterization the action probabilities **change smoothly** as a function of the learned parameter, whereas in ϵ-greedy selection the action probabilities may change dramatically for an arbitrarily small change in the estimated action values, if that change results in a different action having the maximal value. Largely because of this **stronger convergence guarantees** are available for <u>policy-gradient methods</u> than for action-value methods.

The objective function $\mathcal{R}(\boldsymbol{\theta})$ for episodic and continuing tasks are different.

- For **episodic task**, the objective function is the expected returns i.e. the value function under policy π of the start state of the episode:

$$\begin{aligned} \mathcal{R}(\boldsymbol{\theta}) &:= v_{\pi(\boldsymbol{\theta})}(\boldsymbol{s}_0) \\ &= \sum_a \pi(a|\boldsymbol{s}_0, \boldsymbol{\theta}) q_\pi(\boldsymbol{s}_0, a) \qquad (9) \end{aligned}$$

- For **continuing task**, the objective function is the average rewards

$$\begin{aligned}\mathcal{R}(\boldsymbol{\theta}) &:= r(\pi(a|\boldsymbol{s},\boldsymbol{\theta})) \\ &= \sum_s \mu_{\pi(\boldsymbol{\theta})}(s) \sum_a \pi(a|s,\boldsymbol{\theta}) \sum_{s',r} p(s',r|s,a)r. \end{aligned} \tag{10}$$

  where $\boldsymbol{\mu}$ is the **steady-state distribution**. It is also called the **limiting state distribution** under π, $\mu(s) = \lim_{t\to\infty} Pr\{S_t = s|S_0, \pi\}$, or **on-policy distribution** under policy π.

With function approximation, it may seem challenging to change the *policy* parameter in a way that ensures **improvement**. The problem is that performance depends on both the **action selections** and the **distribution of states** in which those selections are made, and that *both of these are affected by the policy* parameter. Note that the effect of the policy on the <u>state distribution</u> $\boldsymbol{\mu}(s)$ is a function of the environment and is typically unknown.

Fortunately, there is an excellent theoretical answer to this challenge in the form of the **policy gradient theorem**, which provides an analytic expression for the gradient of performance with respect to the policy parameter (which is what we need to approximate for gradient ascent) that **does not involve the derivative of the state distribution**.

Theorem 3.1 ***(Policy gradient theorem)*** *For both the episodic case and continuing case under ergodic MDP, the objective functions $\mathcal{R}(\boldsymbol{\theta})$ are defined as in* (9) *and* (10) *respectively. Then*

$$\nabla_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta}) \propto \sum_s \mu_{\pi(\boldsymbol{\theta})}(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(\boldsymbol{s},a) \tag{11}$$

$$= \mathbb{E}_{\boldsymbol{s}\sim\boldsymbol{\mu}_{\pi(\boldsymbol{\theta})}} \left[\sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(\boldsymbol{s},a) \right] \tag{12}$$

$$= \mathbb{E}_{\boldsymbol{s}\sim\boldsymbol{\mu}_{\pi(\boldsymbol{\theta})}} \left[\mathbb{E}_{a\sim\pi(a|\boldsymbol{s},\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \log \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(\boldsymbol{s},a) \right] \right], \tag{13}$$

where $\boldsymbol{\mu}_\pi$ is the limiting state distribution under π, $\mu(s) = \lim_{t\to\infty} Pr\{S_t = s|S_0, \pi\}$, (or on-policy distribution under policy π). In particular, the gradient of objective does not depend on the gradient of state distribution $\nabla\boldsymbol{\mu}$. For ergodic MDP with average reward objective, the equation (11) *is exact. For episodic task, the constant of proportionality is the average length of an episode.*

Proof: We prove the case for the continuing task with ergodic MDP. For the episodic task, please refer the book [Sutton and Barto, 2018] chapter 13.2. We check on the definition of state-value function

$$
\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) \\
\nabla_{\boldsymbol{\theta}} v_\pi(s) &= \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) + \sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} q_\pi(s,a) \\
&\text{due to } q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r - r(\pi) + v_\pi(s')\right] \\
&= \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) + \\
&\sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \sum_{s',r} p(s',r|s,a)\left[r - r(\pi) + v_\pi(s')\right] \\
&= \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) + \\
&\sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \sum_{s',r} p(s',r|s,a)\left[-\nabla_{\boldsymbol{\theta}} r(\pi) + \nabla_{\boldsymbol{\theta}} v_\pi(s')\right]
\end{aligned}
$$

Rearrange this equation and note that $(\sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}))(\sum_{s',r} p(s',r|s,a)) = 1$:

$$
\nabla_{\boldsymbol{\theta}} \mathcal{R} := \nabla_{\boldsymbol{\theta}} r(\pi) = -\nabla_{\boldsymbol{\theta}} v_\pi(s) + \sum_a \left[\nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) + \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \sum_{s',r} p(s',r|s,a) \nabla_{\boldsymbol{\theta}} v_\pi(s')\right] \tag{14}
$$

Now note that left hand side of (14) is not function of state $\boldsymbol{s}$ since the average reward does not depend on state. Therefore the right hand side of (14) is not a function of $\boldsymbol{s}$ either. Since $\sum_s \mu_{\pi(\boldsymbol{\theta})}(s) = 1$, we can mulitple both sides $\sum_s \mu_{\pi(\boldsymbol{\theta})}(s)$:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{R} &= \sum_s \mu_\pi(s) \left\{ -\nabla_{\boldsymbol{\theta}} v_\pi(s) + \sum_a \left[\nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) + \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \sum_{s',r} p(s',r|s,a) \nabla_{\boldsymbol{\theta}} v_\pi(s')\right]\right\} \\
&= \sum_s \mu_\pi(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) + \sum_s \mu_\pi(s) \left\{ -\nabla_{\boldsymbol{\theta}} v_\pi(s) + \sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \sum_{s',r} p(s',r|s,a) \nabla_{\boldsymbol{\theta}} v_\pi(s') \right\} \\
&= \sum_s \mu_\pi(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) - \sum_s \mu_\pi(s) \nabla_{\boldsymbol{\theta}} v_\pi(s) + \\
&\sum_{s'} \left(\sum_s \mu_\pi(s) \sum_a \pi(a|\boldsymbol{s},\boldsymbol{\theta}) \sum_r p(s',r|s,a) \right) \nabla_{\boldsymbol{\theta}} v_\pi(s')
\end{aligned}
$$

Recall that the steady-state distribution $\boldsymbol{\mu}$ has the following equation

$$
\sum_s \mu_\pi(s) \sum_a \pi(a|s,\boldsymbol{\theta}) \sum_r p(s',r|s,a) = \mu_\pi(s'),
$$

which is the term inside parentheses. Therefore we have:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{R} &= \sum_s \mu_\pi(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(s,a) - \sum_s \mu_\pi(s) \nabla_{\boldsymbol{\theta}} v_\pi(s) + \sum_{s'} \mu_\pi(s') \nabla_{\boldsymbol{\theta}} v_\pi(s') \\
&= \sum_s \mu_\pi(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s},\boldsymbol{\theta}) q_\pi(\boldsymbol{s},a). \qquad \blacksquare
\end{aligned}
$$

Parameterized policy methods have an **important theoretical advantage** over action-value methods in the form of the *policy gradient theorem*, which gives an exact formula for how performance is affected by the policy parameter that does not involve <u>derivatives of the state distribution</u>. This theorem provides a theoretical foundation for all policy gradient methods.

4 REINFORCE: Monte Carlo Policy Gradient

Based on policy gradient theorem (11), we can compute the gradient of objective exactly using the value function q_π, the on-policy distribution $\boldsymbol{\mu}$ and the gradient of policy function $\nabla_{\boldsymbol{\theta}}\pi(a|\boldsymbol{s},\boldsymbol{\theta})$. In practice, we can *sample the state sequence* S_t under the policy π. In expectation, samples of S_t replace the steady state distribution $\boldsymbol{\mu}$ since when MDP is stationary, the distribution of state does not change over time. Thus we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{R} &= \sum_s \mu_\pi(s)\sum_a \nabla_{\boldsymbol{\theta}}\pi(a|\boldsymbol{s},\boldsymbol{\theta})q_\pi(\boldsymbol{s},a)\\ &= \mathbb{E}_\pi\left[\sum_a \nabla_{\boldsymbol{\theta}}\pi(a|S_t,\boldsymbol{\theta})q_\pi(S_t,a)\right]\end{aligned}$$

And we can develop the **stochastic gradient asent algorithm** as

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha\sum_a \nabla_{\boldsymbol{\theta}}\pi(a|S_t,\boldsymbol{\theta})\hat{q}(S_t,a) \tag{15}$$

where we replace q_π by its estimate $\hat{q}$. This algorithm is called ***all-actions* method** because its update involves all of the actions, is promising and deserving of further study.

The all-actions method need to scan the entire action space, which is not efficient. We consider replace the expectation with the sample action A_t from policy π.

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{R} &= \mathbb{E}_\pi\left[\sum_a \nabla_{\boldsymbol{\theta}}\pi(a|S_t,\boldsymbol{\theta})q_\pi(S_t,a)\right]\\ &= \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta})\frac{\nabla_{\boldsymbol{\theta}}\pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}q_\pi(S_t,a)\right]\\ &= \mathbb{E}_\pi\left[q_\pi(S_t,A_t)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]\\ &= \mathbb{E}_\pi\left[G_t\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]\\ &= \mathbb{E}_\pi\left[G_t\nabla_{\boldsymbol{\theta}}\log\pi(A_t|S_t,\boldsymbol{\theta})\right]\end{aligned}$$

The second last equation holds by replacing the action-value function by the **sample returns** since $q_\pi(s,a) = \mathbb{E}_\pi\left[G_t|S_t = s, A_t = a\right]$. The last equation holds by chain rule $\nabla_{\boldsymbol{\theta}}\log\pi(A_t|S_t,\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}$. In computation, the $\nabla_{\boldsymbol{\theta}}\log\pi(A_t|S_t,\boldsymbol{\theta})$ is numerically more **stable** to compute vs. the directly ratio. Therefore, we have the **REINFORCE update:**

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha\, G_t\nabla_{\boldsymbol{\theta}}\log\pi(A_t|S_t,\boldsymbol{\theta}), \tag{16}$$

where G_t is the sample returns under the policy π following the sequence $(S_t, A_t, \ldots)$ in an episode. REINFORCE is a <u>**Monte Carlo policy gradient method**</u>, since the algorithm will not update

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Figure 1: The REINFORCE: Monte Carlo policy gradient

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Figure 2: The REINFORCE with value function baseline: Monte Carlo policy gradient

the policy until the **end of each episode**, in order to obtain the sample return G_t. Figure 1 shows the **REINFORCE as Monte Carlo policy gradient method** for *episodic task*.

Like many Monte Carlo methods, REINFORCE is **unbiased** but have **high variance** and **slow learning**. As a stochastic gradient method, REINFORCE has good *theoretical* ***convergence*** *properties*. By construction, the expected update over an episode is in the same direction as the performance gradient. This *assures an improvement* in expected performance for sufficiently small α, and convergence to a **local optimum** under standard stochastic approximation conditions for decreasing α.

The REINFORCE update has some appealing properties: Each increment is proportional to the product of a return G_t and a vector, the gradient of the log-probability of taking the sample action. This vector is the direction in paramete space that ***most*** **increases** the ***probability*** *of* ***repeating*** *the action A_t on* ***future*** *visits* to state S_t. The update increases the parameter vector in this direction **proportional to the return**, and **inversely proportional to the action probability**. The former makes sense because it causes the parameter to move most in the directions that *favor actions* that yield the highest return. The latter makes sense because otherwise actions that are selected frequently are at an advantage (the updates will be more often in their direction) and might win out even if they do not yield the highest return.

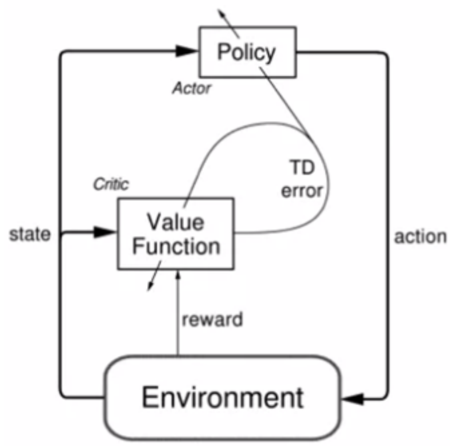

Figure 3: The Actor-Critic methods

4.1 REINFORCE with baseline

The policy gradient theorem can be generalized to include arbitrary baseline $b(\boldsymbol{s})$

$$\nabla_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta}) \propto \sum_s \mu_{\pi(\boldsymbol{\theta})}(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s}, \boldsymbol{\theta}) \left(q_\pi(\boldsymbol{s}, a) - b(\boldsymbol{s})\right)$$

The equation holds since $b(\boldsymbol{s}) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|\boldsymbol{s}, \boldsymbol{\theta}) = 0$. Thus the **REINFORCE-with-baseline** update

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha \left(G_t - b(S_t)\right) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta}), \tag{17}$$

In general, the baseline leaves the expected value of the update unchanged, but the **variance will be reduced** significantly. A natural choice of baseline function is the approximate value function $\hat{v}(S_t, \boldsymbol{w}_t)$, whose function parameter is updated using Monte Carlo prediction. Figure 2 describes the REINFORCE with value function estimate as baseline.

Note that although the REINFORCE-with-baseline method learns both a policy and a state-value function, we do not consider it to be an actorcritic method because its *state-value function* is used only as a ***baseline***, not as a critic. That is, it is not used for bootstrapping that updates the value estimate for a state from the estimated values of subsequent states. This is a useful distinction, for only through bootstrapping do we introduce **bias** and an **asymptotic dependence** on the quality of the function approximation.

5 Actor-Critic Methods

The idea behind the Actor-Critic Methods is to use **boostrapping** which estimates the value estimate for current state based on value estimate of its successor states. The **TD error** is used for both ***value estimation*** (prediction) and ***policy gradient*** (control). By introducing bias and an **asymptotic dependence** on the quality of the *function approximation*, Actor-Critic Methods learn policy faster with less variance. First consider **one-step actorcritic methods**. There are two roles for the agent:

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Figure 4: The Actor-Critic methods

- **Actor**: the role of an **actor** is to ***update the policy distribution*** via policy gradient algorithm. The analog of the TD methods introduced, we choose to replace the sample return in the *REINFORCE-with-baseline* with boostrapping $\hat{G}_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t)$. The updates for *actor* is shown below

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}} \, (\hat{G}_t - \hat{v}(S_t, \boldsymbol{w}_t)) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta}) \\ &= \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}} \, (R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}_t)) \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta}) \\ &= \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}} \, \underline{\delta_t} \, \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta}) \end{aligned} \tag{18}$$

  where

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}_t)$$

  is the **TD error**. When TD error is positive, the selected action resulted in a higher value than expected, which is desireable. *The actor updates the policy distribution **based on value function provided by critic**.*

- **Critic**: Given the learned policy $\pi(a|\boldsymbol{s}, \boldsymbol{\theta})$, the role of a **critic** is to ***evalute* the *value* of the policy** and used it as a **feedback** for actor's performance. As shown in Figure 3, the same TD error δ_t is used by **semi-gradient methods** (e.g. TD(0)) for function apprximation.

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha_{\boldsymbol{w}} \, \underline{\delta_t} \, \nabla_{\boldsymbol{w}} \hat{v}(S_t, \boldsymbol{w}_t) \tag{19}$$

  As oppose to actor which maximize the value by improving the policy, the update (19) will adjust the value function to **match** the ***target value***, i.e. moving in direction to minimize the Mean Squared Value Error.

Figure 4 shows the *actor-critic algorithm with one-step TD updates.* Since the algorithm use the value function estimate as a baseline, it is also called the **Advantage Actor Critic (A2C)**.

Note that we can replace the state-value function with the **action-value function**, which includes TD error for SARSA, Q-learning and Expected SARSA.

$$\nabla_{\boldsymbol{\theta}} \mathcal{R} = \mathbb{E}_{\pi} \left[q_{\pi}(S_t, A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t | S_t, \boldsymbol{\theta})}{\pi(A_t | S_t, \boldsymbol{\theta})} \right]$$

In the **Q Actor Critic**, for the actor, the policy gradient update is the stochastic gradient ascent (use sample action in (15)) as below:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}} \, \hat{q}(S_t, A_t, \boldsymbol{w}_t) \, \nabla_{\boldsymbol{\theta}} \log \pi(A_t | S_t, \boldsymbol{\theta})$$

and for critic, the semi-gradient methods for value function update as below:

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

where the TD error δ_t are defined as below:

$$\textbf{SARSA} \qquad \delta_t := [R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)] \tag{20}$$

$$\textbf{Q-Learning} \qquad \delta_t := \left[R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}, a', \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t) \right] \tag{21}$$

$$\textbf{Expected Sarsa} \qquad \delta_t = \left[R_{t+1} + \gamma \sum_{a'} \pi(a' | S_{t+1}) \hat{q}(S_{t+1}, a', \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t) \right] \tag{22}$$

References

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.