Policy Gradient Method

In this section, we briefly introduce the learning framework we use for this task.

Reinforcement learning has several methods to maximize the cumulative rewards. This paper focuses on one of them – policy optimization. Policy is the strategy that agents utilize to take actions. In mathematics, a stochastic policy $\pi(a|s)$ is a function that takes previous states as input and generates a probability distribution of actions.

Optimizing policy $\pi(a|s)$ is equivalent to finding one that maximizes the expected cumulative rewards of trajectories,

$$\mathrm{E}[\sum_{t=0}^{T-1} r_t|\pi]. \tag{1}$$

A trajectory refers to a sequence of states and actions $(s_0, a_0, s_1, a_1, ..., s_{T-1}, a_{T-1})$.

When the policy is parameterized by θ, we are equivalently solving

$$\max_\theta \mathrm{E}[\sum_{t=0}^{T-1} r_t|\pi_\theta] = \max_\theta \sum_{t=0}^{T-1} \mathrm{E}[r_t|\pi_\theta]$$

To learn parameter θ, gradient-based updating algorithms are usually used [**?**]. The main step is to compute the gradient of the sum of the expectations,

$$\nabla_\theta \sum_{t=0}^{T-1} \mathrm{E}[r_t|\pi_\theta] = \sum_{t=0}^{T-1} \nabla_\theta \,\mathrm{E}[r_t|\pi_\theta]. \tag{2}$$

The linearity of expectation tells us finding the gradient of each reward is enough. Then each step-wise expected reward $r_t, 0 \le t \le T-1$, which is a function of $\tau = (s_0, a_0, s_1, a_1, ..., s_t, a_t)$ by Markov decision process, can be represented in integral form

$$\mathrm{E}[r_t|\pi_\theta] = \int_\tau r_t(\tau) P_\theta(\tau) d\tau, \tag{3}$$

where $P_\theta(\tau)$ can be further expanded by chain rule into

$$P_\theta(\tau) = \mu_0(s_0)\pi_\theta(a_0|s_0)P(s_1|s_0, a_0)\pi_\theta(a_1|s_1)...P(s_t|s_{t-1}, a_{t-1})\pi_\theta(a_t|s_t)$$

with $\mu_0(s_0)$ being the initial distribution of s_0.

Taking the gradient to (3) we have

$$\begin{aligned}
\nabla_\theta \operatorname{E}[r_t|\pi_\theta] &= \nabla_\theta \int_\tau r_t(\tau) P_\theta(\tau) d\tau \\
&= \int_\tau r_t(\tau) \nabla_\theta P_\theta(\tau) d\tau \\
&= \int_\tau r_t(\tau) (\nabla_\theta \log P_\theta(\tau)) P_\theta(\tau) d\tau \\
&= \int_\tau r_t(\tau) \left(\nabla_\theta \sum_{t'=0}^{t} \log \pi_\theta(a_{t'}|s_{t'}) \right) P_\theta(\tau) d\tau \\
&= \underset{\tau \sim P_\theta(\tau)}{\operatorname{E}} \left[r_t(\tau) \left(\sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \right) \right]
\end{aligned}$$

where the derivative utilizes the equality $\nabla_\theta P_\theta(\tau) = (\nabla_\theta \log P_\theta(\tau)) P_\theta(\tau)$ and the fact that when differentiating with respect to θ, unrelated terms drop out.

So we can rewrite the objective function 2 as

$$\sum_{t=0}^{T-1} \nabla_\theta \operatorname{E}[r_t|\pi_\theta] = \underset{\tau \sim P_\theta(\tau)}{\operatorname{E}} \left[\sum_{t=0}^{T-1} \left(r_t(\tau) \left(\sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \right) \right) \right] \tag{4}$$

$$= \underset{\tau \sim P_\theta(\tau)}{\operatorname{E}} \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} r_{t'}(\tau) \right) \right) \right]. \tag{5}$$

The last formula (5) provides a convenient way to approximate the policy gradient: sample a trajectory, compute the empirical gradient $\hat{g}$ by 5, then update θ through

$$\theta := \theta + \delta \hat{g}$$

where δ is the learning rate. Furthermore, one can update the parameter with more sophisticated optimizing algorithms like Adam [1] and RMSProp [2].

0.0.1 Variance Reduction

If the trajectory is quite long, chances are that our noisy gradient estimator is of high variance[1]. One method is to subtract a baseline function $b(s_t)$

[1] To see this, consider random walk $y_0 = \epsilon_0; y_t = y_{t-1} + \epsilon_t$ where ϵ_i are Gaussian iid noise with variance σ^2. Then $\operatorname{Var}[\sum_{t=1}^{N} y_t] = N\sigma^2$.

from our estimator (5):

$$\sum_{t=0}^{T-1} \nabla_\theta \, \mathrm{E}[r_t|\pi_\theta] = \mathop{\mathrm{E}}_{\tau \sim P_\theta(\tau)} \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} r_{t'}(\tau) - b(s_t) \right) \right) \right].$$

The equality holds for arbitrary function $b(s_t)$. See details in [3]. The quantity $\sum_{t'=t}^{T-1} r_{t'}(\tau) - b(s_t)$, which is used to scale the score function, can be regarded as the advantage of taking action a_t at step t. A near optimal baseline function is the state-value function[4]

$$V^\pi(s_t) = \mathrm{E}\left[\sum_{i=t}^{T-1} r_i | s_t, \pi \right].$$

This choice also reflects the idea of advantage in that the difference between actual rewards and expected rewards is naturally a good representation of the advantage function.

Another method can be utilized to reduce the variance is to add discount factor $0 < \gamma < 1$ to future rewards[2]. This reflects a bias-variance trade-off because adding γ decreases the variance but increases the bias. Then rewrite our estimator with such a state-value function as

$$\mathop{\mathrm{E}}_{\tau \sim P_\theta(\tau)} \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}(\tau) - V^{\pi,\gamma}(s_t) \right) \right) \right]$$

where

$$V^{\pi,\gamma}(s_t) = \mathrm{E}\left[\sum_{i=t}^{T-1} \gamma^{t'-t} r_i | s_t, \pi \right].$$

We denote $\sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}(\tau) - V^{\pi,\gamma}(s_t)$ by $A(s_t, a_t)$ for simplicity.

The third method is to use multiple agents to go for independent trajectories and then take the average of their gradients to update the parameter[3].

All these three methods add up to asynchronous advantage actor-critic (A3C) method[5].

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] Briefly, $\mathrm{Var}[\gamma X] = \gamma^2 \mathrm{Var}[X]$.

[3] Suppose $X_i, i = 1, ..., N$ are iid, then $\mathrm{Var}\left[\frac{\sum_{i=1}^N X_i}{N}\right] = \frac{\mathrm{Var}[X]}{N}$.

[2] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[3] John Schulman. *Optimizing expectations: From deep reinforcement learning to stochastic computation graphs*. PhD thesis, UC Berkeley, 2016.

[4] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[5] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.