CS294: Deep Reinforcement Learning [1]

Lecture Notes: Policy Gradient [2]

Fall 2017[3]

[1] Course Instructors: Sergey Levine

[2] Author: InnerPeace

[3] Un-official Lecture Notes

Keyphrases: Policy Gradient. Variance Reduction.
This set of notes introduces derivation of the policy gradient algorithm. We then discuss the basic variance reduction techniques like causality and baselines. Lastly, we will illustrate some examples of policy gradient which helps to understand practical considerations for policy gradients.

1 *Derivation*

Recall that the goal of RL is maximizing the expected cumulative rewards of trajectories distributed according to product of probabilities of state-action sequences:

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \left(\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\right) \tag{1}$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \Big[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \Big] \tag{2}$$

What we are trying to do is evaluating the expected cumulative rewards, and it's hard unless the distribution is clean like Gaussian. The straightforward solution is approximating by Monte Carlo *i.e.*, by drawing samples. For similarity, we denote the expected cumulative rewards as $J(\theta)$, and $r(\tau)$ as the cumulative rewards over T time-steps.

$$r(\tau) = \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \Big[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \Big] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \tag{3}$$

We generate samples from trajectory distribution $p_\theta(\tau)$, and approximating $J(\theta)$ by averaging. The more samples, the more accurate it will be. Or we can also rewrite $J(\theta)$ as the integral of probability of trajectory times the reward according to the definition of expectation.

$$J(\theta) = \int \pi_\theta(\tau) r(\tau) d\tau \tag{4}$$

Following the intuition of DL, firstly, we evaluate the gradients of the objective w.r.t the parameters, then, applying gradient descent or

ascent to improve the objective value. Before derivation, we introduce a convenient identity, for any distribution $p_\theta(\tau)$:

$$p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) = p_\theta(\tau)\frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau) \tag{5}$$

Just using the property of derivation of log function. The gradient of $J(\theta)$ w.r.t parameters of policy networks is:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) d(\tau) \\
&= \int \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} r(\tau) d(\tau) \\
&= \int \pi_\theta(\tau)\nabla_\theta \log(\pi_\theta(\tau)) r(\tau) d(\tau) \\
&= E_{\tau\sim p_\theta(\tau)}\left[\nabla_\theta \log(\pi_\theta(\tau)) r(\tau)\right]
\end{aligned} \tag{6}$$

We take the gradients into the integral with the property that differentiation is linear, and replace $\nabla_\theta \pi_\theta(\tau)$ with the identity of equation 5, then rewrite it with form of expectation. To get the derivation of log of probability of trajectory, we can take *logarithm* to both sides of equation 1:

$$\begin{aligned}
\log \pi_\theta(\tau) &= \log p(\mathbf{s}_1) + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \\
\nabla_\theta \log \pi_\theta(\tau) &= \cancel{\nabla_\theta \log p(\mathbf{s}_1)} + \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \cancel{\nabla_\theta \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \\
&= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)
\end{aligned} \tag{7}$$

Then we substitute $\nabla_\theta \log \pi_\theta(\tau)$ of equation 6 with result of equation 7:

$$\begin{aligned}
\nabla_\theta J(\theta) &= E_{\tau\sim p_\theta(\tau)}\Big[\Big(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\Big)\Big(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\Big)\Big] \\
&\approx \frac{1}{N}\sum_{i=1}^{N}\Big[\Big(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\Big)\Big(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})\Big)\Big]
\end{aligned} \tag{8}$$

The wonderful thing happens to equation 8 is that the gradients of the objective does not depend on the probability of initial state and transition distribution. If we can sample from the dynamical system, then we can evaluate the gradient w.r.t to policy π and improve policy with gradient updating:

$$\theta \quad \leftarrow \quad \theta + \alpha \nabla_\theta J(\theta) \tag{9}$$

Combining these components together to the *REINFORCE algorithm*:

Algorithm 1: REINFORCE algorithm

1: sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ by running the policy.
2: $\nabla_\theta J(\theta) \approx 1/N \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$
3: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4: Back to step 1: policy sampling.

For *POMDP*, since we don't use Markov Property during derivation, we can apply policy gradient to *POMDP* by substituting $\mathbf{s}_{i,t}$ with $\mathbf{o}_{i,t}$ and without further modification.

To dive deeper, if we apply maximum likelihood as the objective in imitation learning, then the gradient should be:

$$\nabla_\theta J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \Big) \tag{10}$$

Compare it with equation 8, they are extremely similar to each other except that policy gradient in equation 8 is weighted by the cumulative sum of rewards of the trajectory, and the sum of rewards could be positive or negative. In general, *Maximum Likelihood* will always increase the probability of trajectory in all the data, while policy gradient prefer the trajectory with high reward and increase the probabilities of the these trajectories, decrease the probability of trajectories with low reward.

1.1 Example: Gaussian Policies

For the *Humanoid* task in gym of openai where the robot has continuous actions, we need the output of policy to be distribution over continuous vector. We choose to use conditional Gaussian and neural network to look at the state $\mathbf{s}_t$ and output the mean of distribution, and for the covariance matrix, we can either use constant like 1 and only care about the mean, or take it as a trainable variable in practice.

$$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \mathcal{N}(f_{NN}(\mathbf{s}_t); \Sigma) \tag{11}$$

Where we use the multivariate Gaussian distribution [4] which is commonly expressed in terms of parameters μ and Σ, where μ is an $n \times 1$ vector and Σ is an $n \times n$, symmetric matrix.

[4] Refer lecture notes of Multivariate Gaussian by Jordan, UCB for more details

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \right\} \tag{12}$$

Suppose that we have an i.i.d distributed data set $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$, and we form the log likelihood function by taking the logarithm of

product of N Gaussians:

$$\begin{aligned}\log(\mu, \Sigma|\mathcal{D}) &= \log\left(p(x_1|\mu,\Sigma)p(x_2|\mu,\Sigma)\cdots p(x_N|\mu,\Sigma)\right) \\ &= -\frac{N}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu) + \text{const} \quad (13)\end{aligned}$$

If we take Σ as constant and substitute with policy at time-step t:

$$\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = -\frac{1}{2}(\mathbf{a}_t - f(\mathbf{s}_t))^T\Sigma^{-1}(\mathbf{a}_t - f(\mathbf{s}_t)) + \text{const} \quad (14)$$

Along with the properties:

$$(AB)^T = B^T \cdot A^T \quad (15)$$

$$\frac{\partial \mathbf{u}^T\mathbf{v}}{\partial \mathbf{v}} = \frac{\partial \mathbf{v}^T\mathbf{u}}{\partial \mathbf{v}} = \mathbf{u} \quad (16)$$

Then differentiate equation 14 w.r.t parameters θ

$$\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \left(\mathbf{a}_t - f(\mathbf{s}_t)\right)^T\Sigma^{-1}\frac{df}{d\theta} \quad (17)$$

Substitute $\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ in equation 8, we obtain the the gradient w.r.t parameters of policy and improve policy by gradient updating.

2 *Problems of Policy Gradient*

In general, there are two problems in policy gradient.

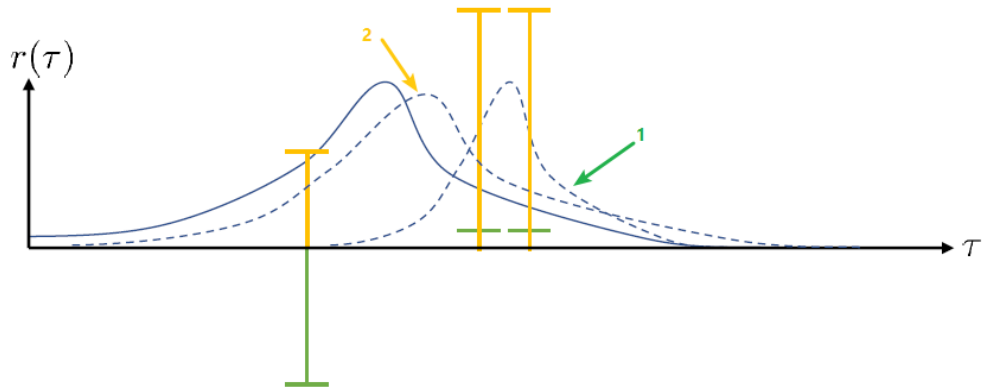


Figure 1: Toy example of High Variance problem

Figure 1 illustrates a simplified example where the trajectory is 1-D along x-axis, and the y-axis indicates the rewards of specific trajectory, the funny blue curve is the policy *i.e.*, the distribution of trajectories. Firstly, we sample 3 trajectories and represent their rewards with green bar where one sample has big negative reward and the other two have small positive rewards. Policy gradient algorithm

will try to shift to the right (indicated by number 1) so that the distribution has more mass on the good trajectories. Then if we add a constant to the reward function, the results of sampled trajectories look like the yellow bar. Besides, adding constant to reward has no impact on gradient, we will prove it later. The situation is that the policy gradient will increase the probabilities of all the three samples, and the curve will move slightly to the right, even choose to increase its variance to become wider (curve indicated by number 2). This example shows that two kinds of rewards which are functionally equivalent but produce different updates. The process will highly depend on the initial position of the distribution if we push down the two positive rewards to zero. This is referred as the *High Variance*. One straightforward way to address this is to take infinite samples where all the differences will eventually cancel out.

Another problem is *slow convergence*. Suppose another one-dimensional policy which distributed according to Gaussian distribution:

$$\pi_\theta(a_t|s_t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(a_t - ks_t)^2}{2\sigma^2}\right\}$$

The logarithm of policy is:

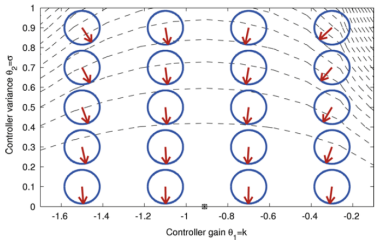
$$\log \pi_\theta(a_t|s_t) = -\frac{1}{2\sigma^2}(a_t - ks_t)^2 + \text{const}$$

And we set the reward as:

$$r(s_t, a_t) = -s_t^2 - a_t^2$$

Where the gradient tries to decrease the state and take small action. And the parameters of policy are linear constant k and variance σ, $\theta = (k, \sigma)$. Figure 2 indicates that the gradient always want to decrease σ, since decreasing the variance will help to keep from choosing big action. At the same time, the objective also want to decrease k to lower the mean of the distribution. But the problem is that the gradient w.r.t σ is much larger than that of k, and as σ decreasing, gradient becomes larger and σ decrease even faster. In contrast, the gradient w.r.t k becomes smaller which results in *slow convergence*. Later, we will introduce *natural gradient estimation* to address the problem.

Figure 2: Illustration of gradients w.r.t $\theta = (k, \sigma)$, image from Peters & Schaal 2008.



2.1 *Causality*

For the problem of *High Variance*, there are two methods to address it.

The first idea is *causality* which means policy at time t' cannot affect reward at time t when $t < t'$. It's intuitive *e.g.* what I do today won't change what happened yesterday as time flows forward. With

this idea, we obtain a better policy gradient estimator by rewriting equation 8 as:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\Big[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\Big(\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'},\mathbf{a}_{i,t'})\Big)\Big] \tag{18}$$

Where we distribute the sum of rewards into the sum of gradients and compute the cumulative rewards starting from time-step t, which is referred as *reward to go*. The intuition is that the variance goes down with summing over fewer numbers. We can denote *reward to go* w.r.t Q-function:

$$Q_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'},\mathbf{a}_{i,t'}) \tag{19}$$

And this is trivial to compute if you starting from the end of the sequences, people always use this idea in practice.

2.2 *Baselines*

The "vanilla" policy gradient change the probabilities of trajectories based how good or bad there are w.r.t the reward, and the toy example shows that adding constant to reward could be problematic. Is there another metric to "good" or "bad" of the trajectory which are more stable? The idea is changing the probability based on how better or worse the trajectory is than the average [5]. For simplicity, we rewrite equation 8 w.r.t trajectory τ.

[5] Refer to Peters & Schaal 2008 for overview of optimal baselines and natural gradient

$$\begin{aligned}\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{i=1}^{N}\Big[\Big(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\Big)\Big(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\Big)\Big] \\ &= \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log \pi_\theta(\tau) r(\tau)\end{aligned}$$

Then we substract reward of each trajectory with baseline b:

$$b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log \pi_\theta(\tau)[r(\tau) - b] \tag{20}$$

Does adding the baseline mathematically legitimate? The result is that adding a constant to the reward, the expectation of gradient is unbiased. The derivation as follows:

$$E[\nabla_\theta \log \pi_\theta(\tau) b] = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) b d\tau = \int \nabla_\theta \pi_\theta(\tau) b d\tau = b\nabla_\theta \int \pi_\theta(\tau) d\tau = b\nabla_\theta 1 = 0 \tag{21}$$

Where we applied the identity in equation 5, and took grad operation and constant b outside the integral, finally, the integral of probability

over trajectories is 1. In fact, the average baseline is not the best baseline, but it's pretty good. What's the optimal baseline? what we are trying to do is reducing the variance, and mathematically, variance of variable x is:

$$\text{Var}[x] = E[x^2] - E[x]^2$$

For policy gradient we substitute x with $\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)$ and the expectation is:

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)] \tag{22}$$

The variance is

$$\text{Var} = E_{\tau \sim \pi_\theta(\tau)}[(\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b))^2] - E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)]^2 \tag{23}$$

We proved that adding constant to reward has no impact on gradient with equation 21, which means:

$$E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)]^2 = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]^2 \tag{24}$$

We substitute grad log pi with $g(\tau)$:

$$g(\tau) = \nabla_\theta \log \pi_\theta(\tau)$$

Then we differentiate Var w.r.t to baseline b, ans set it to 0:

$$\begin{aligned} \frac{d\text{Var}}{db} &= \frac{d}{db} E[g(\tau)^2 (r(\tau) - b)^2] \\ &= \frac{d}{db} (\cancel{E[g(\tau)^2 r(\tau)^2]} - 2E[g(\tau)^2 r(\tau) b] + b^2 E[g(\tau)^2]) \\ &= -2E[g(\tau)^2 r(\tau)] + 2bE[g(\tau)^2] \\ &= 0 \end{aligned} \tag{25}$$

The optimal baselines is:

$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]} \tag{26}$$

This is just expected reward, but weighted by gradient magnitudes. And in practice, average reward works just fine.

3 *Importance Sampling*

We introduced the concept of *on-policy* in notes of lecture 3. And policy gradient is one of the instances where we have to sample new trajectories each time we update the policy network. The problem is that neural networks change only a little bit with each gradient update, results in sampling inefficient. So can we take advantage of

the old samples? The idea is we can sample from other distribution instead of the new policy $\pi_\theta(\tau)$ with *importance sampling*.

Suppose that we want to estimate expectation of $f(x)$ with x distributed according to $p(x)$, but if we have x from distribution $q(x)$, we still calculate the expectation with *importance sampling*:

$$\begin{aligned} E_{x\sim p(x)}[f(x)] &= \int p(x)f(x)dx \\ &= \int \frac{q(x)}{q(x)}p(x)f(x)dx \\ &= \int q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right] \end{aligned} \tag{27}$$

Where we sample x from distribution $q(x)$ and insert the correction $\frac{p(x)}{q(x)}$ which is also referred as *importance wight*. We can apply this idea to policy sampling where we sample from $\pi'(\tau)$ instead of $\pi(\tau)$:

$$J(\theta) = E_{\tau\sim\pi'(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi'(\tau)}r(\tau)\right] \tag{28}$$

And recall that probability of trajectory is the product of probabilities of state-action sequences, then the *importance weight* in equation 28 is:

$$\begin{aligned} \frac{\pi_\theta(\tau))}{\pi'(\tau)} &= \frac{\cancel{p(\mathbf{s}_1)}\prod_{t=1}^{T}\left(\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}\right)}{\cancel{p(\mathbf{s}_1)}\prod_{t=1}^{T}\left(\pi'(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}\right)} \\ &= \frac{\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T}\pi'(\mathbf{a}_t|\mathbf{s}_t)} \end{aligned} \tag{29}$$

In theory, with sampling from $\pi'(\tau)$, we can evaluate new policy $\pi_\theta(\tau)$ without samples. Besides, we derive policy gradient with importance sampling. Firstly, we rewrite equation 28 with policy $\pi_\theta(\tau)$ and $\pi_{\theta'}(\tau)$:

$$J(\theta') = E_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}r(\tau)\right] \tag{30}$$

And we estimate gradient w.r.t new parameter θ':

$$\begin{aligned} \nabla_{\theta'}J(\theta') &= E_{\tau\sim\pi_\theta(\tau)}\left[\frac{\nabla_{\theta'}\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}r(\tau)\right] \\ &= E_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)\right] \end{aligned} \tag{31}$$

Where we applied the identity in equation 5. And we estimate the gradient locally at $\theta = \theta'$:

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\theta(\tau)}\nabla_\theta\log\pi_\theta(\tau)r(\tau)\right] \tag{32}$$

This is just what we derived earlier. What if $\theta \neq \theta'$? We can plug the result of equation 29 into equation 31:

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\frac{\prod_{t=1}^{T} \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \right) \left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t'=1}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) \right] \tag{33}$$

And we can also apply the idea of *causality*: only the importance weight up to current time-step is helpful as future actions don't affect current weight and replace $r(\tau)$ with reward to go.

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\frac{\prod_{t'=1}^{t} \pi_{\theta'}(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\prod_{t'=1}^{t} \pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})} \right) \left(\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) \right] \tag{34}$$

However the product of importance weight is exponential in time-steps which leads to zero or infinite when time-step t is large. We will dive deeper in following lecture to address the problem, intuitively, we can rewrite the objective with converting MDP into *Markov Chain* as discussed in lecture 2:

$$\begin{aligned} J(\theta) &= \sum_{t=1}^{T} E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_\theta(\mathbf{s}_t, \mathbf{a}_t)} \left[r(\mathbf{s}_t, \mathbf{a}_t) \right] \\ &= \sum_{t=1}^{T} E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)] \right] \end{aligned} \tag{35}$$

Where we divide it into expectation over two distributions: one over probability of states and another one over policy. And for the new parameter θ', we can sample from distributions based on parameter θ and apply importance sampling to equation 35:

$$J(\theta') = \sum_{t=1}^{T} E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[\frac{p_{\theta'}(\mathbf{s}_t)}{p_\theta(\mathbf{s}_t)} E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t, \mathbf{a}_t)} \left[\frac{\pi_{\theta'}(\mathbf{s}_t, \mathbf{a}_t)}{\pi_\theta(\mathbf{s}_t, \mathbf{a}_t)} r(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \tag{36}$$

The new objective without the product of importance weight seems practical, but it's more complicated which requiring the probability of a particular state. Later, we will learn how cross out the importance wight over probability of state reasonably.

In practice, we should keep in mind that policy gradient has high variance which is noisy and larger batches helps to reduce the variance. Adaptive step size rule like Adam work fine, but we will learn more policy gradient-specific learning rate adjustment methods later.

4 *Suggested Readings*

Classic Papers

- Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduce REINFORCE algorithm: introduces REINFORCE algorithm. link.

- Baxter & Bartlett (2001). Infinite-horizon policy-gradient estimation: temporally decomposed policy gradient. link.
- Peters & Schaal (2008). Reinforcement learning of motor skills with policy gradients: overview of optimal baselines and natural gradient. link.

Deep Reinforcement Learning

- Levine & Koltun (2013). Guided policy search: deep RL with importance sampled policy gradient. link.
- Schulman et al. (2015). Trust region policy optimization: deep RL with natural gradient and adaptive step size. link.
- Schulman et al. (2017). Proximal policy optimization algorithms: deep RL with importance sampled policy gradient. link.