A Tour through Policy Gradient Method and Its Variants

Jiarui Wang

November 3, 2023

Policy Gradient and its variants have been the state-of-the-art methods to train reinforcement learning models. Although there have been many online materials talking about the policy gradient, they only include the final update rule or over-simplified derivation of the policy gradient. In this note, I want to make a self-contained tutorial to go through the development of policy-gradient methods and their theoretical basis (but it requires basic knowledge in reinforcement learning like value function and Q function), making the knowledge graph about policy gradient into a knowledge "linked list" so that you can follow the material sequentially. Hopefully, this note can be good material for those who (and, of course, myself) want to systematically study policy gradient methods and a good supplementary material for David Silver's RL course and CS294-112.

Contents

1 Notations and Problem Setup

In this note, we'll consider the scenario that the initial state follows a certain distribution ρ_{s_0}, and we want to maximize the expected accumulative reward from the starting state.

- a: Action
- s: State
- γ: Discount factor
- π: Policy
- $P_{ss'}^a$: The transition probability from s to s' by taking action a.
- ρ_{s_o}: The distribution of starting state
- $V^\pi(s)$: The expected (discounted) total reward starting from s by following π
- $\eta(\pi) = E_\pi[\sum_{t=0}^\infty \gamma^t r_t]$: The expected (discounted) total reward
- $Q^\pi(s,a)$: The expected (discounted) total reward starting from s by taking action a and then following π
- $P(s_t = s|\pi)$: The probability of ending up in s' at time t starting from s by following π
- $\rho_\pi(s) = \sum_{t=0}^\infty \gamma^t P(s_t = s|\pi)$: The (unnormalized) discounted visitation frequency
- $\tau = \{s_0, a_0, s_1, a_1, ...\}$ is a full trajectory

The unlisted notation will be made clear when it's used.

2 Policy Gradient Theorem

The policy gradient theorem was first introduced by [1]. In that paper, the author considered two cases: The first case is when we want to maximize the average per-step reward. The second case is when the initial state s_0 is fixed or follows a certain distribution, and we want to maximize the long-term total reward. We'll only consider the second case for simplicity. In this case, the objective of maximizing total reward becomes $\max_\theta V^{\pi_\theta}(s_0)$ We first give the derivation of the policy gradient and then go through it.

$$\begin{aligned}
\frac{\partial}{\partial\theta}V^{\pi_\theta}(s_0) &= \frac{\partial}{\partial\theta}\sum_a \pi_\theta(a|s_0)Q^{\pi_\theta}(s_0,a) \qquad (2.1)\\
&= \sum_a Q^{\pi_\theta}(s_0,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s_0) + \sum_a \pi_\theta(a|s_0)\frac{\partial}{\partial\theta}Q^{\pi_\theta}(s_0,a) \qquad (2.2)\\
&= \sum_s\sum_a\sum_{t=0}^{0}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \sum_a \pi_\theta(a|s_0)\frac{\partial}{\partial\theta}Q^{\pi_\theta}(s_0,a) \qquad (2.3)\\
&= \sum_s\sum_a\sum_{t=0}^{0}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \sum_a \pi_\theta(a|s_0)\gamma\sum_{s'}P^a_{ss'}\frac{\partial}{\partial\theta}V^{\pi_\theta}(s') \qquad (2.4)\\
&= \sum_s\sum_a\sum_{t=0}^{0}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \gamma\sum_s P(s_1=s|s_0,\pi_\theta)\frac{\partial}{\partial\theta}V^{\pi_\theta}(s) \qquad (2.5)\\
&= \sum_s\sum_a\sum_{t=0}^{0}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \gamma\sum_s P(s_1=s|s_0,\pi_\theta)\Bigg(\sum_a Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s) + \sum_a \pi_\theta(a|s)\frac{\partial}{\partial\theta}Q^{\pi_\theta}(s,a)\Bigg) \qquad (2.6)\\
&= \sum_s\sum_a\sum_{t=0}^{0}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \gamma\sum_s P(s_1=s|s_0,\pi_\theta)\sum_a Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \gamma\sum_s P(s_1=s|s_0,\pi_\theta)\sum_a \pi_\theta(a|s)\frac{\partial}{\partial\theta}Q^{\pi_\theta}(s,a) \qquad (2.7)\\
&= \sum_s\sum_a\sum_{t=0}^{1}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \gamma\sum_s P(s_1=s|s_0,\pi_\theta)\sum_a \pi_\theta(a|s)\gamma\sum_{s'}P^a_{ss'}\frac{\partial}{\partial\theta}V^{\pi_\theta}(s') \qquad (2.8)\\
&= \sum_s\sum_a\sum_{t=0}^{1}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s)\\
&\quad + \gamma^2\sum_s P(s_2=s|s_0,\pi_\theta)\frac{\partial}{\partial\theta}V^{\pi_\theta}(s) \qquad (2.9)\\
&= \ldots\ldots\\
&= \sum_s\sum_a\sum_{t=0}^{\infty}\gamma^t P(s_t=s|s_0,\pi_\theta)Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\pi_\theta(a|s) \qquad (2.10)
\end{aligned}$$

The reasoning for the derivation is as follows:

(2.1) : Plug in the definition of $V^{\pi_\theta}(s_0)$

(2.2) : Derivative of the product

(2.3) : Rewrite the first term trivially. Note for each s, it's non-zero only when $s = s_0, t = 0$

(2.4) : Use the definition $Q(s,a) = r(s,a) + \gamma \sum_s' P^a_{ss'} V(s')$. Note $r(s,a)$ does not depend on θ

(2.5) : Reinterpret the probability chain of the second term in (2.4). Note that s in (2.5) is the s' in (2.4).

(2.6) : Plug in the definition of V

(2.7) : Separate the second term of (2.6)

(2.8) : Merge the first two terms of (2.7) and use the definition $Q(s,a) = r(s,a) + \gamma \sum_s' P^a_{ss'} V(s')$

(2.9) : Reinterpret the probability chain of the second term in (2.8). Note that s in (2.9) is the s' in (2.8)

(2.10) : Note the recursive relationship shown in (2.5) and (2.9). We unroll (2.10) infinitely.

Now we can compute the gradient of V^{π_θ} as follows by taking the expectation

$$\begin{aligned}
\frac{\partial}{\partial\theta}\eta(\pi_\theta) &= \sum_{s_0} \rho_0(s_0) \frac{\partial}{\partial\theta} V^{\pi_\theta}(s_0) \\
&= \sum_{s_0} \rho_0(s_0) \sum_s \sum_a \sum_{t=0}^{\infty} \gamma^t P(s_t = s|s_0, \pi_\theta) Q^{\pi_\theta}(s,a) \frac{\partial}{\partial\theta}\pi_\theta(a|s) \\
&= \sum_s \sum_a \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi_\theta) Q^{\pi_\theta}(s,a) \frac{\partial}{\partial\theta}\pi_\theta(a|s) \\
&= \sum_s \rho_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) \frac{1}{\pi_\theta(a|s)} Q^{\pi_\theta}(s,a) \frac{\partial}{\partial\theta}\pi_\theta(a|s) \\
&= E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta(s)}\big[Q^{\pi_\theta}(s,a) \frac{\partial}{\partial\theta} \log \pi_\theta(a|s)\big] \qquad (2.11)
\end{aligned}$$

(2.11) says that we can estimate $\frac{\partial}{\partial\theta}V^{\pi_\theta}$ by sampling from ρ_{π_θ}. But ρ_{π_θ} is hard to sample from since it's a discounted (unnormalized) distribution. However, we can rewrite (2.11) as

$$\frac{\partial}{\partial\theta}\eta(\pi_\theta) = E_{\tau\sim\pi_\theta}\big[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \frac{\partial}{\partial\theta} \log \pi_\theta(a_t|s_t)\big] \qquad (2.12)$$

since

$$
\begin{aligned}
E_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{\infty}\gamma^t Q^{\pi_\theta}(s_t,a_t)\frac{\partial}{\partial\theta}\log\pi_\theta(a_t|s_t)\Big] &= \sum_{t=0}^{\infty}\gamma^t E_{\tau\sim\pi_\theta}\Big[Q^{\pi_\theta}(s_t,a_t)\frac{\partial}{\partial\theta}\log\pi_\theta(a_t|s_t)\Big]\\
&= \sum_{t=0}^{\infty}\gamma^t \sum_s P(s_t=s|\pi_\theta)E_{a\sim\pi_\theta(s)}\Big[Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\log\pi_\theta(a|s)\Big]\\
&= E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta(s)}\Big[Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\log\pi_\theta(a|s)\Big]
\end{aligned}
$$

However, in practice, people usually approximate (2.12) with

$$
\frac{\partial}{\partial\theta}\eta(\pi_\theta) \approx E_{\tau\sim\pi_\theta}\Big[Q^{\pi_\theta}(s_t,a_t)\frac{\partial}{\partial\theta}\log\pi_\theta(a_t|s_t)\Big] \tag{2.13}
$$

This works fine in practice. But I'm not sure why this is a valid approximation. One intuition might be we optimize the gain from not only the starting state but from every state.

3 Reducing Variance with A Baseline

Most content of the section is derived from [2]
(2.13) directly suggests an algorithm as follows called *REINFORCE*

Algorithm 1 REINFORCE

```
Initialize $\pi_\theta$
while not converge do
    Sample episode $s_0, a_0, r_0, ...s_T \sim \pi_\theta$
    Compute $G_t = \sum_{\tau=t}^{T}\gamma^{\tau-t}r_\tau$
    for t=1,...,T-1 do
        $\theta := \theta + \alpha G_t \nabla_\theta \log\pi_\theta(a_t|s_t)$
    end for
end while
```

This seems to be a reasonable gradient ascent method, but it does not always work in practice since G_t is high-variance even though it's an unbiased estimator of $V^{\pi_\theta}(s_t)$. One way is to subtract a baseline $B(s)$ form $Q(s,a)$. It's fine if this is not clear for now, let's see how it works.

For any function $B(s)$ (not that $B(s)$ depends only on s, not on a), we have

$$
\begin{aligned}
E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta}\big[\nabla_\theta\log\pi_\theta(a|s)B(s)\big] &= \sum_s \rho_{\pi_\theta}(s)\sum_a \nabla_\theta\pi_\theta(a|s)B(s)\\
&= \sum_s \rho_{\pi_\theta}(s)B(s)\nabla_\theta\sum_a \pi_\theta(a|s)\\
&= \sum_s \rho_{\pi_\theta}(s)B(s)\nabla_\theta 1\\
&= 0
\end{aligned}
$$

thus we can write

$$E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta(s)}\big[Q^{\pi_\theta}(s,a)\nabla_\theta\log\pi_\theta(a|s)\big]=E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta(s)}\big[\big(Q^{\pi_\theta}(s,a)-B(s)\big)\nabla_\theta\log\pi_\theta(a|s)\big]$$

The intuition is as follows: $B(s)$ how good is state s over all actions. And we define the advantage function $A^{\pi_\theta}(s,a)=Q^{\pi_\theta}(s,a)-B(s)$ is how much more "goodness" we can get at s by taking a.

One good choice for the baseline function is $B(s)=V^{\pi_\theta}(s)$, then we have the policy gradient

$$\begin{aligned}\frac{\partial}{\partial\theta}\eta(\pi_\theta)&=E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta(s)}\big[Q^{\pi_\theta}(s,a)\frac{\partial}{\partial\theta}\log\pi_\theta(a|s)\big]\\&=E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta(s)}\big[\frac{\partial}{\partial\theta}\log\pi_\theta(a|s)\big(Q^{\pi_\theta}(s,a)-V^{\pi_\theta}(s)\big)\big]\\&=E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta(s)}\big[\frac{\partial}{\partial\theta}\log\pi_\theta(a|s)\big(r(s,a)+\gamma V^{\pi_\theta}(s')-V^{\pi_\theta}(s)\big)\big]\end{aligned}\tag{3.1}$$

We can see in (3.1), the advantage function is exactly the form of TD error. $V^{\pi_\theta}(s)$ is how good s is, and $r(s,a)+\gamma V^{\pi_\theta}(s')-V^{\pi_\theta}(s)$ is how much better we get by taking action a at s. If $r(s,a)+\gamma V^{\pi_\theta}(s')-V^{\pi_\theta}(s)$ is large, it means taking action a at s gives us much juice, thus we increase $\log\pi_\theta(a|s)$ by the amount of juice.

As in (2.13), people usually sample trajectories instead of ρ_{π_θ} in practice, and it works well.

4 Actor-Critic with TD Learning

One problem in (3.1) is that we don't have V^{π_θ}. One way to solve this problem is to we approximate V^{π_θ} using an approximator, and like in generalized policy gradient, we switch between estimating the value function and improving the policy. This paradigm suggests an algorithm called Actor-Critic, where we have an policy called "actor" and a value function approximation called "critic" at the same time.

The actor-critic algorithm is as follow

Algorithm 2 TD Actor Critic

```
Initialize actor $\pi_\theta$
Initialize critic $V_w$
while not converge do
    Sample a batch of $(s,a,r,s')$ tuple from $\pi_\theta$
    Compute TD target $y(s)=r(s)+V_w(s').detach()$ // no gradient though the target
    Compute the TD error $\delta(s)=y(s)-V_w(s)$ // gradient passes through $y(s)$
    Compute critic loss $closs=\|\delta\|_2^2$
    Compute actor loss $aloss=-\sum_s\nabla_\theta\log\pi_\theta(a|s)*\delta(s).detach()$ // no gradient through the advantage function
    Take a gradient step to minimize $aloss+closs$
end while
```

One thing to mention is that, similar to Q-learning, I think we should fix the TD target (that's why $V_w(s')$ is detached or even has a pair of neural nets approximating the V function that switches over and over as in DQN. However, it works just fine even if we don't detach anything and let the gradient flows through all paths.

5 Monotonic Improvement Theorem

If you're not interested in the detailed derivation, you can go through Lemma 1 and jump to section 6.2. You'll be good.

5.1 Monotonic Improvement Theorem Derivation

Most content of the section is derived from [3], but re-organized. Some of the derivations/definitions are not exactly the same as in the paper, those are based on my understanding. You can find the lemma/definition/theorem by the same index number in the paper.

Lemma 1 *Given 2 policies π, $\tilde{\pi}$, let $\bar{A}(s;\pi,\tilde{\pi}) = E_{a\sim\tilde{\pi}(s)}[A^\pi(s,a)]$, we have*

$$\begin{aligned}
\eta(\tilde{\pi}) &= \eta(\pi) + E_{\tau\sim\tilde{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t,a_t)\big] \\
&= \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s)\sum_a \tilde{\pi}(a|s)A^\pi(s,a) \\
&= \eta(\pi) + E_{\tau\sim\tilde{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t \bar{A}(s_t;\pi,\tilde{\pi})\big]
\end{aligned}$$

proof:

$$\begin{aligned}
E_{\tau\sim\tilde{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t,a_t)\big] &= E_{\tau\sim\tilde{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t\big(r(s_t,a_t)+\gamma V^\pi(s_{t+1})-V^\pi(s_t)\big)\big] \\
&= E_{\tau\sim\tilde{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\big] - E_{\tau\sim\tilde{\pi}}\big[V^\pi(s_0)\big] \\
&= \eta(\tilde{\pi}) - \eta(\pi) \\
E_{\tau\sim\tilde{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t,a_t)\big] &= \sum_{t=0}^{\infty}\gamma^t E_{\tau\sim\tilde{\pi}}\big[A^\pi(s_t,a_t)\big] \\
&= \sum_{t=0}^{\infty}\gamma^t\sum_s P(s_t=s|\tilde{\pi})\sum_a\tilde{\pi}(a|s)A^\pi(s,a) \\
&= \sum_s \rho_{\tilde{\pi}}(s)\sum_a\tilde{\pi}(a|s)A^\pi(s,a)
\end{aligned}$$

Definition 1 *$(\pi,\tilde{\pi})$ is an $\alpha-$coupled policy pair if exists a joint distribution $(A,\tilde{A})|S$ such that*

- *$P(A\neq\tilde{A}|s)\leq\alpha \quad \forall\ s$*
- *Marginal of A is $\pi(s) \quad \forall\ s$*

- *Marginal of $\tilde{A}$ is $\tilde{\pi}(s)$ $\forall$ s*

If you're not familiar with coupling, [4] is a good material for background knowledge.

Lemma 2 *Let $(\pi, \tilde{\pi})$ be an α-coupled policy pair, then*

$$|\bar{A}(s; \pi, \tilde{\pi})| \leq 2\alpha \max_{s,a} |A^{\pi}(s,a)|$$

proof:

$$\begin{aligned}
\bar{A}(s; \pi, \tilde{\pi}) &= E_{\tilde{a} \sim \tilde{\pi}(s)} \left[A^{\pi}(s, \tilde{a}) \right] \\
&= E_{(a,\tilde{a}) \sim (\pi(s), \tilde{\pi}(s))} \left[A^{\pi}(s, \tilde{a}) - A^{\pi}(s,a) \right] \\
&= P(a \neq \tilde{a} | s) E_{(a,\tilde{a}) \sim (\pi(s), \tilde{\pi}(s)) | a \neq \tilde{a}} \left[A^{\pi}(s, \tilde{a}) - A^{\pi}(s,a) \right] \\
|\bar{A}(s; \pi, \tilde{\pi})| &\leq 2\alpha \max_{s,a} |A^{\pi}(s,a)|
\end{aligned}$$

Lemma 3 *Let $(\pi, \tilde{\pi})$ be an coupled policy pair, then*

$$\left| E_{s_t \sim \tilde{\pi}}\left[\tilde{A}(s_t; \pi, \tilde{\pi})\right] - E_{s_t \sim \pi}\left[\tilde{A}(s_t; \pi, \tilde{\pi})\right] \right| \leq 2\alpha \max_s \bar{A}(s_t; \pi, \tilde{\pi}) \leq 4\alpha\big(1 - (1-\alpha)^t\big) \max_{s,a} |A^\pi(s,a)|$$

proof:

[1] Consider we sample $\tau \sim \pi, \tilde{\tau} \sim \tilde{\pi}$, let $n_t = \sum_{i=1}^{t-1} 1\{a_i \neq \tilde{a}_i\}$. Since $P(a_t = \tilde{a}_t | s_t) \geq 1 - \alpha$, we have $P(n_t = 0) \geq (1-\alpha)^t, P(n_t > 0) \leq 1 - (1-\alpha)^t$

$$\begin{cases} E_{s_t \sim \tilde{\pi}}[\tilde{A}(s_t; \pi, \tilde{\pi})] = P(n_t = 0) E_{s_t \sim \tilde{\pi} | n_t = 0}[\bar{A}(s_t; \pi, \tilde{\pi})] + P(n_t > 0) E_{s_t \sim \tilde{\pi} | n_t > 0}[\bar{A}(s_t; \pi, \tilde{\pi})] \\ E_{s_t \sim \pi}[\tilde{A}(s_t; \pi, \tilde{\pi})] = P(n_t = 0) E_{s_t \sim \pi | n_t = 0}[\bar{A}(s_t; \pi, \tilde{\pi})] + P(n_t > 0) E_{s_t \sim \pi | n_t > 0}[\bar{A}(s_t; \pi, \tilde{\pi})] \end{cases}$$

Note that $P(n_t = 0) E_{s_t \sim \tilde{\pi} | n_t = 0}[\bar{A}(s_t; \pi, \tilde{\pi})] = P(n_t = 0) E_{s_t \sim \pi | n_t = 0}[\bar{A}(s_t; \pi, \tilde{\pi})]$. Subtracting two equations and taking the absolute value we have

$$\begin{aligned} &\left| E_{s_t \sim \tilde{\pi}}[\tilde{A}(s_t; \pi, \tilde{\pi})] - E_{s_t \sim \pi}[\tilde{A}(s_t; \pi, \tilde{\pi})] \right| \\ =& P(n_t > 0) \left| E_{s_t \sim \tilde{\pi} | n_t > 0}[\bar{A}(s_t; \pi, \tilde{\pi}) - E_{s_t \sim \pi | n_t > 0}[\bar{A}(s_t; \pi, \tilde{\pi}) \right| \\ \leq& \big(1 - (1-\alpha)^t\big) 2 \max_s \left| \bar{A}(s_t; \pi, \tilde{\pi}) \right| \\ \leq& 4\alpha\big(1 - (1-\alpha)^t\big) \max_{s,a} \left| A^\pi(s,a) \right| \end{aligned}$$

In the final step, we use the result of Lemma 2.

Theorem 1 *Let $L(\tilde{\pi}; \pi) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a)$ and $\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a)$ (by lemma 1), we have*

$$\begin{aligned} \eta(\tilde{\pi}) &\geq L(\tilde{\pi}; \pi) - \frac{4\gamma}{(1-\gamma)^2} \max_{s,a} \left| A^\pi(s,a) \right| D_{TV}^{\max}(\pi, \tilde{\pi})^2 \\ &\geq L(\tilde{\pi}; \pi) - \frac{4\gamma}{(1-\gamma)^2} \max_{s,a} \left| A^\pi(s,a) \right| D_{KL}^{\max}(\pi, \tilde{\pi})^2 \end{aligned}$$

proof:

Let $\alpha = D_{TV}^{\max}(\pi, \tilde{\pi})$, then $\forall\ s, \alpha \geq D_{TV}(\pi(s), \tilde{\pi}(s)) = \inf_{(A, \tilde{A})} P(A \neq \tilde{A} | s)$ where the inf is taken over all $(A, \tilde{A})$ that is a coupling of $(\pi(s), \tilde{\pi}(s))$. Thus $(\pi, \tilde{\pi})$ is an α-coupled policy pair.

$$\begin{aligned} \eta(\tilde{\pi}) - L(\tilde{\pi}, \pi) &= \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a) - \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a) \\ &= E_{\tau \sim \tilde{\pi}}\left[\sum_{t=0}^\infty \gamma^t \bar{A}(s_t; \pi, \tilde{\pi}) \right] - E_{\tau \sim \pi}\left[\sum_{t=0}^\infty \gamma^t \bar{A}(s_t; \pi, \tilde{\pi}) \right] \\ &= \sum_{t=0}^\infty \gamma^t \Big(E_{\tau \sim \tilde{\pi}}\Big[\bar{A}(s_t; \pi, \tilde{\pi}) \Big] - E_{\tau \sim \pi}\Big[\bar{A}(s_t; \pi, \tilde{\pi}) \Big] \Big) \end{aligned}$$

[1] I don't really trust this step. α-couple policy pair gives $P(A \neq \tilde{A} | s) \leq \alpha$, which says two policies agrees with high probability **given** s, which says $P(n_t > 0 | s_1, ..., s_{t-1}) \leq 1 - (1-\alpha)^t$. Note here it should be a conditional probability (in my opinion).

$$\begin{aligned}
\Big|\eta(\tilde{\pi}) - L(\tilde{\pi}, \pi)\Big| &\le \sum_{t=0}^{\infty} \gamma^t \Big| E_{\tau\sim\tilde{\pi}}\Big[\bar{A}(s_t; \pi, \tilde{\pi})\Big] - E_{\tau\sim\pi}\Big[\bar{A}(s_t; \pi, \tilde{\pi})\Big]\Big| \\
&\le \sum_{t=0}^{\infty} \gamma^t 4\alpha\big(1 - (1-\alpha)^t\big) \max_{s,a} \big|A^\pi(s,a)\big| \\
&= \left[\frac{1}{1-\gamma} - \frac{1}{1-\gamma(1-\alpha)}\right] 4\alpha \max_{s,a} \big|A^\pi(s,a)\big| \\
&\le \frac{4\alpha^2\gamma}{(1-\gamma)^2} \max_{s,a} \big|A^\pi(s,a)\big| \\
&= \frac{4\gamma}{(1-\gamma)^2} \max_{s,a} \big|A^\pi(s,a)\big| D_{TV}^{\max}(\pi, \tilde{\pi})^2 \\
&\le \frac{4\gamma}{(1-\gamma)^2} \max_{s,a} \big|A^\pi(s,a)\big| D_{KL}^{\max}(\pi, \tilde{\pi})
\end{aligned}$$

In the last step, we use Pinsker's inequality $D_{TV}(P,Q) \le \sqrt{\frac{1}{2} D_{KL}(P,Q)}$.

5.2 Monotonic Improvement Theorem: In a Nutshell

Let $L(\tilde{\pi}; \pi) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a)$ and $\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a)$. Suppose we are training a reinforcement learning agent and we have policy π (π might not be a good policy) at some iteration, we are interested in finding a next policy $\tilde{\pi}$ that maximizes $\eta(\tilde{\pi})$. However, it is hard to optimize $\eta(\tilde{\pi})$ since it depends on $\rho_{\tilde{\pi}}$, a visitation distribution of an unknown policy.

Theorem 1 bounded the difference between $\eta(\tilde{\pi})$ and $L(\tilde{\pi}; \pi)$ by $D_{KL}^{\max}(\pi, \tilde{\pi})$, which says $L(\tilde{\pi}; \pi)$ is a good approximation of $\eta(\tilde{\pi})$ when two policies π and $\tilde{\pi}$ are similar to each other. Thus we can instead optimize $L(\tilde{\pi}; \pi)$ while keeping $D_{KL}^{\max}(\pi, \tilde{\pi})$ small. Thus we transform the policy optimization as follows

$$\begin{aligned}
\max_{\tilde{\pi}}.\quad & \eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a) \\
& \downarrow \text{by Theorem 1} \\
\max_{\tilde{\pi}}.\quad & L(\tilde{\pi}; \pi) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a) \\
s.t.\quad & D_{KL}^{\max}(\pi, \tilde{\pi}) \le \delta \\
& \downarrow \text{Importance Sampling} \\
\max_{\tilde{\pi}}.\quad & \sum_s \rho_\pi(s) \sum_a \pi(a|s) \frac{\tilde{\pi}(a|s)}{\pi(a|s)} A^\pi(s,a) \\
s.t.\quad & D_{KL}^{\max}(\pi, \tilde{\pi}) \le \delta \\
& \downarrow \text{Approximation}
\end{aligned}$$

$$\begin{aligned} \max_{\tilde{\pi}}. \quad & \mathcal{L}(\tilde{\pi};\pi) = E_{s\sim\rho_\pi, a\sim\pi}\left[\frac{\tilde{\pi}(a|s)}{\pi(a|s)}A^\pi(s,a)\right] \\ s.t. \quad & E_{s\sim\rho_\pi}\big[D_{KL}(\pi(s),\tilde{\pi}(s))\big] \le \delta \end{aligned} \tag{5.1}$$

The main takeaway is that we finally get the optimization problem (5.1). Here we define $\mathcal{L}(\tilde{\pi};\pi) = E_{s\sim\rho_\pi, a\sim\pi}\left[\frac{\tilde{\pi}(a|s)}{\pi(a|s)}A^\pi(s,a)\right]$ for later use.

6 Natural Policy Gradient

Lots of content in the remaining sections is derived from [5].

6.1 Intuition for Trust Region: Natural Gradient Descent

Before we dive again into algorithms, let's see natural gradient descent which will give us a good intuition about Natural Policy Gradient.

As you may know, one thing in gradient descent is that we should not use too large a step size which may lead to divergence. However, in policy gradient, even if we take a small step size to θ' close to θ in the parameter space, $\pi_{\theta'}$ is not necessarily π_θ in the policy space. If at one iteration, we perform a gradient step and reach a bad policy, it may be hard to recover from the bad policy since the gradient step uses the samples collected from the bad policy.

One way to deal with this problem is to keep $\pi_{\theta'}$ and π_θ close explicitly. And we demonstrate this idea with natural gradient descent.

Suppose we have a distribution $P_\theta(X)$ parameterized by $\theta \in \mathbb{R}^n$. For example, θ can be the mean and covariance of a multivariate Gaussian. Suppose we now want to minimize some loss function $L(\theta)$ with respect to θ. For example, $L(\theta)$ can be the negative likelihood of some data under P_θ. Then we can write the optimization problem as

$$\min_\theta \; L(\theta)$$

For non-convex problems, we typically want to update the parameters θ in a "smooth" way, which means we do not want to change the model too drastically (the model is $P_\theta(\cdot)$ in this case). One way is to constrain the difference between the model before and after each step (for example, before and after each gradient step). To this end, we can solve the following constrained optimization at each step

$$\begin{aligned} \min_{\theta'} \quad & L(\theta') \\ s.t. \quad & KL\big(P_\theta \| P_{\theta'}\big) \le \delta \end{aligned} \tag{6.1.1}$$

Here, θ is the current value of parameter, $KL(P\|Q) = \int p(x)\log\frac{p(x)}{q(x)}dx$ is the KL-divergence, and p, q denote the probability densities of P, Q. At each step, we solve (1), which means we want to minimize $L(\theta')$ while making $P_{\theta'}$ stay close to P_θ. We call this a trust region method.

We first approximate the KL divergence using a second-order Taylor expansion

$$KL(P_\theta\|P_{\theta'}) \approx KL(P_\theta\|P_\theta) + \nabla_{\theta'} KL(P_\theta\|P_{\theta'})\big|_{\theta'=\theta}^{T}(\theta'-\theta) + \frac{1}{2}(\theta'-\theta)^T\nabla^2_{\theta'}KL(P_\theta\|P_{\theta'})\big|_{\theta'=\theta}(\theta'-\theta)$$

$KL(P_\theta\|P_\theta) = 0$. $KL(P_\theta\|P_{\theta'}) = 0$ when $\theta' = \theta$, thus θ' is the local/global minimum of $KL(P_\theta\|P_{\theta'})$, so the gradient $\nabla_{\theta'}KL(P_\theta\|P_{\theta'})\big|_{\theta'=\theta} = 0$. Thus the first 2 terms are both 0, $KL(P_\theta\|P_{\theta'}) \approx \frac{1}{2}(\theta'-\theta)^T H(\theta'-\theta)$

To approximately solve the optimization problem (4.1), we apply first-order approximation to the objective function $L(\theta') \approx L(\theta) + \nabla L(\theta)^T(\theta'-\theta)$, and second-order approximation to the constraint $KL(P_\theta\|P_{\theta'}) \approx \frac{1}{2}(\theta'-\theta)^T\nabla^2_{\theta'}KL(P_\theta\|P_{\theta'})\big|_{\theta'=\theta}(\theta'-\theta)$, then we get the optimization problem

$$\begin{aligned} \min_{\theta'}.\ & g^T(\theta'-\theta) \\ s.t.\ & \frac{1}{2}(\theta'-\theta)^T H(\theta'-\theta) \le \delta \end{aligned} \tag{6.1.2}$$

where $g = \nabla_{\theta'}L(\theta')\big|_{\theta'=\theta}, H = \nabla^2_{\theta'}KL(P_\theta\|P_{\theta'})\big|_{\theta'=\theta}$. We can derive a closed-form solution to (4.2) $\theta'^* = \theta + -\frac{\sqrt{2\delta}H^{-1}g}{\sqrt{g^TH^{-1}g}}$ with a change of variable.

$H = \nabla^2_{\theta'}KL(P_\theta\|P_{\theta'})\big|_{\theta'=\theta}$ can be computed as

$$\begin{aligned} H &= \nabla^2_{\theta'}KL(P_\theta\|P_{\theta'})\Big|_{\theta'=\theta} \\ &= \nabla^2_{\theta'}\int P_\theta(x)\log\frac{P_\theta(x)}{P_{\theta'}(x)}dx \\ &= -E_{x\sim P_\theta}\left[\nabla^2_{\theta'}\log P_{\theta'}(x)\right]\Big|_{\theta'=\theta} \\ &= -E_{x\sim P_\theta}\left[\nabla_{\theta'}\frac{\nabla_\theta P_{\theta'}(x)}{P_\theta(x)}\right]\Bigg|_{\theta'=\theta} \\ &= -E_{x\sim P_\theta}\left[\frac{P_\theta(x)\nabla^2_\theta P_{\theta'}(x) - \nabla_{\theta'}P_{\theta'}(x)\nabla_{\theta'}P_{\theta'}(x)^T}{P_\theta(x)^2}\right]\Bigg|_{\theta'=\theta} \\ &= -\nabla^2_\theta\int P_{\theta'}(x)dx + E_{x\sim P_\theta}\left[\nabla_{\theta'}\log P_{\theta'}(x)\nabla_{\theta'}\log P_{\theta'}(x)^T\right]\Bigg|_{\theta'=\theta} \\ &= E_{x\sim P_\theta}\left[\nabla_{\theta'}\log P_{\theta'}(x)\nabla_{\theta'}\log P_{\theta'}(x)^T\right]\Bigg|_{\theta'=\theta} \end{aligned} \tag{6.1.3}$$

6.2 Natural Policy Gradient

If you can understand natural gradient descent, then Natural Policy Gradient is just an application. Similar to (5.1), we have the optimization problem

$$\begin{aligned} \max_{\theta'}.\ & \mathcal{L}(\theta';\theta) = E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}A^{\pi_\theta}(s,a)\right] \\ s.t.\ & E_{s\sim\rho_{\pi_\theta}}\left[D_{KL}(\pi(s), \pi_{\theta'}(s))\right] \le \delta \end{aligned} \tag{6.2.1}$$

Note here that θ' is the variable we optimize over, and θ is the parameter of the old policy. We first compute the gradient

$$\begin{aligned}\nabla_{\theta'}\mathcal{L}(\theta';\theta)\big|_{\theta'=\theta} &= \nabla_{\theta'} E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta}\left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}A^{\pi_\theta}(s,a)\right]\Bigg|_{\theta'=\theta} \\ &= E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta}\left[\frac{\nabla_{\theta'}\pi_{\theta'}(a|s)\big|_{\theta'=\theta}}{\pi_\theta(a|s)}A^{\pi_\theta}(s,a)\right] \\ &= E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta}\left[\nabla_{\theta'}\log\pi_{\theta'}(a|s)\big|_{\theta'=\theta}A^{\pi_\theta}(s,a)\right] \qquad (6.2.2)\end{aligned}$$

Note that (7.2) is exactly the policy gradient, same as (3.1). And similar to (6.1.3), the hessian $\nabla^2_{\theta'}E_{s\sim\rho_{\pi_\theta}}\left[D_{KL}(\pi(s),\pi_{\theta'}(s))\right]$ can be estimated as $E_{s\sim\rho_{\pi_\theta},a\sim\pi_\theta}\left[\nabla_{\theta'}\log\pi_{\theta'}(a|s)\nabla_{\theta'}\log\pi_{\theta'}(a|s)^T\right]$

The relationship between policy gradient and Natural Policy Gradient is shown in Figure 1. NPG is trying to go as far as possible in PG direction while staying in the trust region. The same interpretations also carry to Natural Gradient Descent. Note $H^{-1}g$ is very similar to the newton direction, the only difference is that in Newton's direction, H is the hessian of the objective (i.e. derivative of g), or putting it in another way, the trust region is measured by the local-hessian norm.

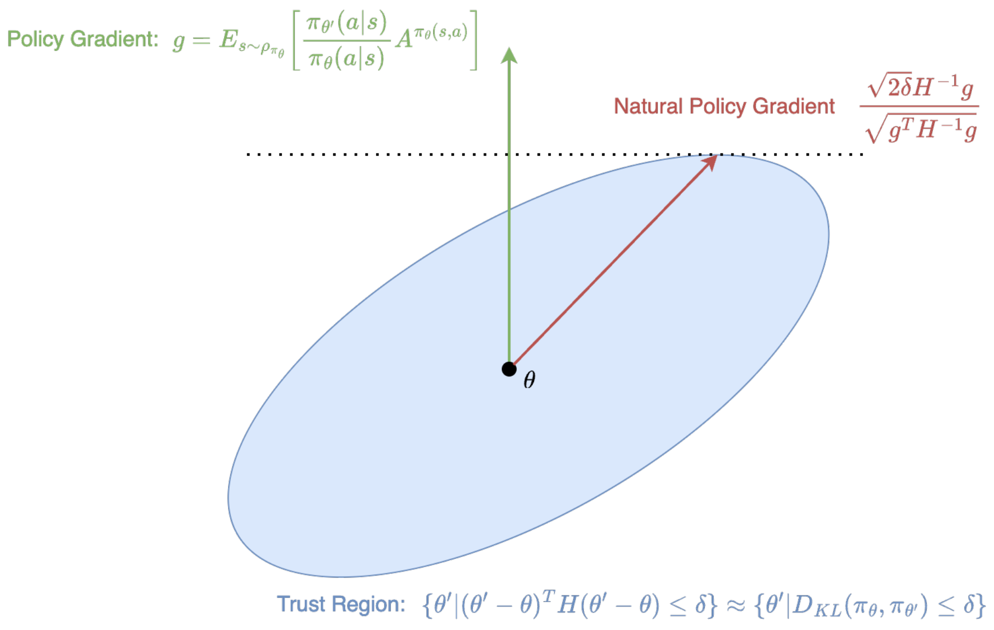
Policy Gradient: $g = E_{s\sim\rho_{\pi_\theta}}\left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}A^{\pi_\theta(s,a)}\right]$
Natural Policy Gradient $\frac{\sqrt{2\delta}H^{-1}g}{\sqrt{g^TH^{-1}g}}$
θ
Trust Region: $\{\theta'|(\theta'-\theta)^TH(\theta'-\theta)\le\delta\}\approx\{\theta'|D_{KL}(\pi_\theta,\pi_{\theta'})\le\delta\}$

Figure 1: Policy Gradient and Natural Policy Gradient

Let's outline the Natural Policy Gradient in algorithm 3 to conclude this section.

Algorithm 3 Natural Policy Gradient

```
Initialize actor $\pi_\theta$
while not converge do
    Sample a batch of $(s, a, r, s')$ tuple from $\pi_\theta$
    Compute $g = E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_{\theta'} \log \pi_{\theta'}(a|s)\big|_{\theta'=\theta} A^{\pi_\theta}(s,a)\right]$
    Compute $H = E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_{\theta'} \log \pi_{\theta'}(a|s) \nabla_{\theta'} \log \pi_{\theta'}(a|s)^T\right]$
    Update $\theta' = \theta + \frac{\sqrt{2\delta} H^{-1} g}{\sqrt{g^T H^{-1} g}}$
end while
```

7 Trust Region Policy Optimization

In Natural Policy Optimization, we try to go in the direction of policy gradient while staying in the trust region. But note that the constraint in (6.2.1) is approximated by the second-order Taylor expansion. It may be the case that the natural policy gradient can bring us out of the trust region (the constraint in (6.2.1)). To solve this problem, Appendix C of [3] proposed using line search to ensure we "improve the nonlinear objective while satisfying the nonlinear constraint". The TRPO algorithm is outlined in

Algorithm 4 Trust Region Policy Optimization

```
Initialize actor $\pi_\theta$
while not converge do
    Sample a batch of $(s, a, r, s')$ tuple from $\pi_\theta$
    Compute $g = E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_{\theta'} \log \pi_{\theta'}(a|s)\big|_{\theta'=\theta} A^{\pi_\theta}(s,a)\right]$
    Compute $H = E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_{\theta'} \log \pi_{\theta'}(a|s) \nabla_{\theta'} \log \pi_{\theta'}(a|s)^T\right]$
    Compute NPG direction $\Delta\theta = \frac{\sqrt{2\delta} H^{-1} g}{\sqrt{g^T H^{-1} g}}$
    for j=1,...,L do
        Compute proposed update $\tilde{\theta} = \theta + \beta^j \Delta\theta$
        if $\mathcal{L}(\tilde{\theta}; \theta) > 0$ and $D_{KL}(\pi_{\tilde{\theta}}, \pi_\theta) < \delta$ then
            accept the update $\theta := \theta + \beta^j \Delta\theta$
        end if
    end for
end while
```

Despite its huge impact on the further development of reinforcement learning, TRPO is nothing but Natural Policy Gradient plus backtracking line search which is widely used in optimization algorithms. Once you understand NPG, TRPO is very simple.

8 Proximal Policy Optimization

In both NPG and TRPO, we need to compute $H^{-1}g$, which can be efficiently solved with the conjugate gradient method (if you're not familiar with conjugate gradient, [6] is a good tutorial).

However, solving linear systems is still very expensive. In [7], two methods to approximately solve (6.2.1) are proposed.

8.1 Adaptive KL Penalty Coefficient

One variant of PPO is the Adaptive KL Penalty Coefficient, outlined in algorithm 5.

Algorithm 5 PPO: Adaptive KL Penalty Coefficient

```
Initialize actor $\pi_\theta$
while not converge do
    Sample a batch of $(s, a, r, s')$ tuple from $\pi_\theta$
    Take a gradient step to maximize $E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}\hat{A}(s,a) - \beta D_{KL}\big(\pi_\theta(s), \pi_{\theta'}(s)\big)\right]$
    for Every $k$ steps do
        Compute $d = E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[D_{KL}\big(\pi_\theta(s), \pi_{\theta'}(s)\big)\right]$
        if  then$d < \delta/1.5$
            $\beta = \beta/2$
        else if  then$d > \delta * 1.5$
            $\beta = \beta * 2$
        end if
    end for
end while
```

This algorithm maximizes $\mathcal{L}(\theta';\theta)$ by penalizing the KL divergence, occasionally checks the violence of the KL divergence constraint, and increases the penalizing weight if the KL constraint is violated significantly or decreases the penalizing weight if the KL constraint is satisfied by a large gap.

This type of algorithm can be seen as an approximation of the dual gradient method (if you're not familiar with this, [8] is a good resource, the "price update" is a wonderful interpretation).

8.2 Clipped Surrogate Objective

Another variant of PPO maximizes the following objective

$$E_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[\min\left(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}\hat{A}(s,a), \text{Clip}\big(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}, 1-\epsilon, 1+\epsilon\big)\hat{A}(s,a)\right)\right]$$

Intuitively, in this algorithm, even if we change the policy drastically, the change is compromised by $\text{Clip}(\cdot, 1-\epsilon, 1+\epsilon)$.

Consider this case: $\hat{A}(s,a)$ is large, we can make $\pi_{\theta'}(a|s)$ very large to maximize the objective. However, if $\pi_{\theta'}(a|s)$ is very large, $\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}\hat{A}(s,a) > \text{Clip}\big(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}, 1-\epsilon, 1+\epsilon\big)\hat{A}(s,a)$, the first term is ignored, thus the gradient flow through the first term is cut down. Also, $\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} > 1+\epsilon$, thus the clip function takes the value $1+\epsilon$, the gradient flow through $\pi_{\theta'}(a|s)$ is cut down. So when $\pi_{\theta'}(a|s)$ exceeds $(1+\epsilon)\pi_\theta(a|s)$ to achieve a good objective, the gradient flow is cut down and it's not updated (for this (s,a) pair) anymore.

9 Generalized Advantage Estimation

In [7], they used generalized advantage estimation, which is, similar to λ-return, an important technique to balance the bias-variance trade-off in TD learning. The generalized advantage estimation $\hat{A}(s,a)$ is computed as

$$
\begin{aligned}
\hat{A}^{(1)}(s_0,a_0) &= r(s_0,a_0) + \gamma\hat{V}(s_1,a_1) - \hat{V}(s_0) \\
\hat{A}^{(2)}(s_0,a_0) &= r(s_0,a_0) + \gamma r(s_1,a_1) + \gamma^2\hat{V}(s_2) - \hat{V}(s_0) \\
&\vdots \\
\hat{A}(s_0,a_0) &= (1-\lambda)\big(\hat{A}^{(1)}(s_0,a_0) + \lambda\hat{A}^{(2)}(s_0,a_0) + \lambda^2\hat{A}^{(3)}(s_0,a_0) + \ldots\big) \\
&= \quad (1-\lambda)\lambda^0\big(r(s_0,a_0) + \gamma\hat{V}(s_1) - \hat{V}(s_0)\big) \\
&\quad + (1-\lambda)\lambda^1\big(r(s_0,a_0) + \gamma r(s_1,a_1) + \gamma^2\hat{V}(s_2) - \hat{V}(s_0)\big) \\
&\quad + (1-\lambda)\lambda^1\big(r(s_0,a_0) + \gamma r(s_1,a_1) + \gamma^2 r(s_2,a_2) + \gamma^3\hat{V}(s_3) - \hat{V}(s_0)\big) \\
&\qquad \vdots \\
&= -\hat{V}(s_0) \\
&\qquad + (1-\lambda)\lambda^0\big(r(s_0,a_0) + \gamma\hat{V}(s_1)\big) \\
&\qquad + (1-\lambda)\lambda^1\big(r(s_0,a_0) + \gamma r(s_1,a_1) + \gamma^2\hat{V}(s_2)\big) \\
&\qquad + (1-\lambda)\lambda^2\big(r(s_0,a_0) + \gamma r(s_1,a_1) + \gamma^2 r(s_2,a_2) + \gamma^3\hat{V}(s_3)\big) \\
&\qquad \vdots \ \text{group the term in same color} \\
&= -\hat{V}(s_0) \\
&\qquad + (\gamma\lambda)^0\big(r(s_0,a_0) + \gamma(1-\lambda)\hat{V}(s_1)\big) \\
&\qquad + (\gamma\lambda)^1\big(r(s_1,a_1) + \gamma(1-\lambda)\hat{V}(s_2)\big) \\
&\qquad + (\gamma\lambda)^2\big(r(s_2,a_2) + \gamma(1-\lambda)\hat{V}(s_3)\big) \\
&\qquad \vdots \\
&= -\hat{V}(s_0) \\
&\qquad + (\gamma\lambda)^0\big(r(s_0,a_0) + \gamma\hat{V}(s_1) - \gamma\lambda\hat{V}(s_1)\big) \\
&\qquad + (\gamma\lambda)^1\big(r(s_1,a_1) + \gamma\hat{V}(s_2) - \gamma\lambda\hat{V}(s_2)\big) \\
&\qquad + (\gamma\lambda)^2\big(r(s_2,a_2) + \gamma\hat{V}(s_3) - \gamma\lambda\hat{V}(s_3)\big) \\
&\qquad \vdots \ \text{shift red terms down by 1} \\
&= \quad (\gamma\lambda)^0\big(r(s_0,a_0) + \gamma\hat{V}(s_1) - \hat{V}(s_0)\big) \\
&\quad + (\gamma\lambda)^1\big(r(s_1,a_1) + \gamma\hat{V}(s_2) - \hat{V}(s_1)\big) \\
&\quad + (\gamma\lambda)^2\big(r(s_2,a_2) + \gamma\hat{V}(s_3) - \hat{V}(s_2)\big) \\
&\quad \vdots \\
&= (\gamma\lambda)^0\delta_0 + (\gamma\lambda)^1\delta_1 + (\gamma\lambda)^2\delta_2 + \ldots
\end{aligned}
$$

References

[1] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[2] David Silver. Reinforcement learning, ucl. `https://www.youtube.com/watch?v=mpbWQbkl8_g`.

[3] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[4] Total variation and coupling. `https://jerryzli.github.io/robust-ml-fall19/lec2.pdf`.

[5] Cs294-112, uc berkeley. `https://www.youtube.com/watch?v=ycCtmp4hcUs&list=PLkFD6_40KJIznC9CDbVTjAF2oyt8_VAe3&index=14`, 10 2017.

[6] conjugate gradeint method. `http://lovinglavigne.com/ConjugateGradient/cg.pdf`, 2021.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[8] Ee364b, lec 4. `https://www.youtube.com/watch?v=kE3wtUaQzpA&list=PL7DD2F5ED3D1A1514&index=4`, 2021.