Reinforcement Learning

Jubayer Ibn Hamid

This document is a combination of notes from CS 234 taught by Prof. Emma Brunskill at Stanford University, CS 224R taught by Prof. Chelsea Finn at Stanford University, CS 285 taught by Prof. Sergey Levine at UC Berkeley as well as reading notes from various papers. There may be various errors throughout, both minor and major. I have also not been able to stick to the same notation throughout these notes and I apologize for that. Hopefully, the inconsistencies are not too troublesome.

Contents

1 Framework **4**
1.1 MDP 4
1.2 Values 4
1.3 State distributions 7
1.4 Oversimplified overview 8

2 Policy Gradient Methods **9**
2.1 Deriving the Policy Gradient 10
2.2 Policy Gradient Theorem 11
2.3 REINFORCE 15
2.4 REINFORCE using causality 15
2.5 REINFORCE with baseline 16
2.6 On-Policy Actor-Critic Methods 18
2.7 Making inroads to off-policy policy gradient 18
2.8 Off-Policy Actor-Critic 19
2.9 Soft Actor-Critic 20
2.10 Performance Difference Lemma 21
2.11 Covariant/natural policy gradient 24
2.12 Trust Region Policy Optimization (TRPO) 25
2.13 Proximal Policy Optimization (PPO) 30
2.14 Deterministic Policy Gradient Methods (DPG) 32

1 Framework

1.1 MDP

For the majority of these notes, we will consider the framework of a Markov Decision Process (MDP), represented as $\langle \mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma, H \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, p is the transition dynamics, R is the reward function, ρ_0 is the initial state distribution, $\gamma \in [0, 1]$ is the discount factor and H is the task horizon (which could be positive infinity).

The reward function is $r(s_t, a_t) = \mathbb{E}[r|s_t, a_t]$ i.e., the expected reward from taking an action from the current state. Here we are assuming that the reward after taking action a_t from state s_t has a distribution and $r(s_t, a_t)$ is the mean of that distribution. However, the reward could also be deterministic in which case the mean would collapse to the deterministic value. Eitherways, we will use the shorthand $r_t := r(s_t, a_t)$.

A policy refers to an agent's distribution over actions at each state i.e.,

$$\pi(a|s) := P(A_t = a|S_t = s).$$

Note that a policy can be deterministic too, in which case $\pi(a|s)$ can be written as a Dirac-delta distribution.

An episode refers to "one play of the game" i.e., our agent ends up in the terminal state or the agent has taken H actions after which the environment is reset.

Goal of reinforcement learning: Our agent seeks to maximize the expected discounted sum of rewards i.e.

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t r_t \mid \pi\right].$$

Note, the expectation is taken over the distribution of the random variables $s_t, a_t, s_{t+1}, a_{t+1}, \ldots$ induced by transition dynamics $p(s_{t'+1} \mid s_{t'}, a_{t'})$ and the policy $\pi(a_{t'}|s_{t'})$ for all $t' \in [0, H]$.

1.2 Values

The total return from a trajectory from time t ownards is

$$G_t = r_t + r_{t+1} + \cdots + r_H$$

if H is finite. When we use the *discounted* sum of rewards, one can define the total discounted return even for infinite horizon tasks

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}.$$

Then, we see that the goal of reinforcement learning is to maximize the expected total return from a trajectory. For most of these notes, I will try to be consistent and assume that H is infinite for the sake of simplicity.

Definition 1. (Optimal Policy). The optimal policy is defined as:

$$\pi^*(a|s) = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_t \gamma^t r_t \mid \pi, s\right]$$

Now, we introduce some more definitions that help express the goal of reinforcement learning - maximizing the expected (discounted) sum of rewards. To do so, we define the following:

Definition 2. Value function. Given a policy π, the value function of π refers to the expected sum of (discounted) rewards when starting from a given state s and acting according to π. For generality, suppose $H = \infty$, then this can be written as:

$$V_{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_t \Big| \pi, S_0 = s\right].$$

One can expand this to write

$$V_{\pi}(s) = \mathbb{E}_{\substack{a\sim\pi(a|s) \\ s',r\sim P(s',r|s,a)}}\left[r + \gamma V_{\pi}(s') \mid S_0 = s\right].$$

The derivation is as follows:

$$\begin{aligned}
V_{\pi}(s) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right]. \\
&= \mathbb{E}_{\pi}\left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t | s_0 = s\right]. \\
&= \mathbb{E}_{\pi}\left[r_0\right] + \mathbb{E}_{\pi}\left[\gamma \cdot \sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0 = s\right]. \\
&= \sum_a \pi(a|s_0 = s) \sum_{s',r} P(s', r|s, a)\left[r\right] + \sum_a \pi(a|s_0 = s) \sum_{s',r} P(s', r|s, a)\left[\gamma \mathbb{E}_{\pi}\left[G_{t+1}|S_1 = s'\right]\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)\left[r + \gamma V_{\pi}(s')\right] \\
&= \mathbb{E}_{\substack{a\sim\pi(a|s) \\ s',r\sim P(s',r|s,a)}}\left[r + \gamma V_{\pi}(s')\right]
\end{aligned}$$

We then see that the goal of RL is to find the optimal policy defined by $\pi(a \mid s) = \arg\max_{\pi} V^{\pi}(s)$.

Intuitively, $V^\pi(s)$ is telling us, on average, how "good" we are when we start from state s and follow policy π (here "good" refers to the expected total return).

However, sometimes we might be interestd in the following question - instead of following policy π from state s, what if we take a specific action a and *then* follow policy π. In other words, we are trying to understand the value of taking a specific action a over the trajectory induced by our policy π.

Definition 3. Action-value function/Q function. The q-value of an action from a state is defined to be

$$q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \Big| S_0 = s, A_0 = a \right].$$

Intuitively, $q_\pi(s,a)$ refers to the expected sum of discounted rewards from taking action a from state s and then following policy π from then on.

With a similar derivation as for $V_\pi(s)$, this can also be written as

$$q_\pi(s,a) = \mathbb{E}_{s',r\sim P(s',r|s,a)} \left[r + \gamma V_\pi(s') | S_0 = s, A_0 = a \right].$$

Note:

$$V_\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)} \left[Q_\pi(s,a) | s \right]$$

Definition 4. Optimal value function and optimal action-value function. The optimal value function is the expected sum of discounted rewards when starting from a given state s and acting optimally:

$$\begin{aligned} V^*(s) &= \max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{H} \gamma^t r_t | \pi, s_0 = s \right] \\ &= \max_\pi V_\pi(s). \end{aligned}$$

Similarly, we define

$$q_*(s,a) = \max_\pi q_\pi(s,a).$$

In other words, $q_*(s,a)$ is the value of taking action a and then acting optimally. One useful relation is

$$V_*(s) = \max_a q_*(s,a).$$

Definition 5. Optimal policy. We say $\pi \geq \pi'$ if and only if $V_\pi(s) \geq V_{\pi'}(s)$ for all $s \in \mathcal{S}$. The optimal policy $\pi_*(s)$ is

$$\pi_*(s) = \arg\max_\pi V_\pi(s).$$

Lemma 1. For an infinite horizon problem i.e $H = \infty$, the optimal policy is deterministic, stationary (i.e action distribution at a given state does not depend on the time) and not necessarily unique.

Note: we either require horizon H to be finite or, if $H = \infty$, we require $\gamma < 1$.

Why do we care about value functions and action-value functions? This is because knowing these would help us find the optimal policy fairly easily. Here are two ways you can do it:

1. If we have a policy π and we know $q_\pi(s, a)$, we can set the optimal policy to be

$$\pi^*(a \mid s) = 1\{a = \arg\max_{a'} q_\pi(s, a')\}. \tag{1}$$

   In other words, the optimal policy could just take the optimal action in terms of $q_\pi(s, a)$. If we define this for every state, then we could get an optimal policy regardless of what π we use in equation 1. Note that this would make a deterministic policy.

2. if policy $\pi_\theta(a \mid s)$ is parametrized by parameters θ, we could update these parameters such that $\pi_\theta(a|s)$ for any good action a is maximized. In other words, if action a is such that $q_\pi(s, a) > V_\pi(s)$, we maximize $\pi_\theta(a \mid s)$ (increase probability of that action being taken).

1.3 State distributions

We will often require some notion of the distribution of states visited or distribution of states we might start from.

- $\mathbb{P}(s_0 \to s, k, \pi_\theta)$ is the probability of reaching state s from s_0 in k time-steps under policy π. We can calculate this as follows:

$$\mathbb{P}(s_0 \to s, k, \pi) = P(s_0) \prod_{t=0}^{k-2} \left(\sum_{a_t, s_{t+1}} \pi(a_t \mid s_t) P(s_{t+1}|s_t, a_t) \right) \sum_{a_{k-1}} \pi(a_{k-1} \mid s_{k-1}) P(s_k = s|s_{k-1}, a_{k-1}).$$

- $p^\pi(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(s_0 \to s, k, \pi)$ is the improper discounted state distribution.
- Let the distribution over all states induced by π_θ be:

$$d^{\pi_\theta}(s) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_0 \to s, t, \pi).$$

  The factor $(1-\gamma)$ is a normalization constant and this distribution simply discounts states visited later in time.

1.4 Oversimplified overview

With this, we can now give an overview of various methods discussed in these notes:

1. value-based methods: here we first try to estimate either the value function or the action-value function of a policy by interacting with the environment and collecting rewards. Once we have estimated the value of the policy, we update the policy and then we repeat. While this works for relatively small state and action spaces, for larger ones, we would requite value function approximation methods.
2. policy gradient methods: directly learn the optimal policy π_θ parametrized by θ. We learn this by updating θ so as to maximize the expected value. For these methods, we would still require some estimation of the value of the policy or, at the very least, returns from a policy.
3. model-based reinforcement learning: learn the transition model i.e. the transition dynamics of the environment. Once we have this transition model, we can use it for planning or improving a policy.

Generally, the algorithms we consider will be characterized as one of two things:

1. off-policy: these algorithms can collect experiences (experiences are lists like $s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots, s_{t+H}$) using a policy π_β and use that to update a *different* policy π_θ.
2. on-policy: to update any policy π, we require experiences collected using policy π. In other words, we cannot use experiences from other policies to update this policy.

While on-policy methods can give us more reliable signals to help us update our policy, off-policy methods are more sample efficient.

2 Policy Gradient Methods

In policy gradient methods, we directly parametrise the policy to be $\pi_\theta(a|s)$. Our goal is then to find π_θ that maximizes returns. In the other algorithms we have seen so far, we tried to estimate the value of a policy and *then* improved it, whereas, in the case of policy gradients, we will directly aim to learn the optimal policy. We may still learn a value function in order to help learn our parametrized optimal policy, but the value function will not be required for our action selection (in the sense that, when selecting which action to take, we will not query the value function - the value function is only used to help us update our policy π_θ, often as a baseline to reduce variance as we will soon see). Methods that incorporate both a parametrized policy and a parametrized learned value function are often called *actor-critic methods*. For policy-based methods, we have no value function, but only a learned policy.

Policy gradient methods are useful for generating stochastic policies, but they are often hard to evaluate and can have a high variance. A useful example of a parametrized policy to keep in mind is a Gaussian policy (especially for continuous action spaces) which is parametrized as $a \sim \mathcal{N}(\mu_\theta(s), \sigma^2) =: \pi_\theta(\cdot \mid s)$. However, more complex parameterizations can be used; 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.

Recall notation:

- $\mathbb{P}(s_0 \to s, k, \pi_\theta)$ is the probability of reaching state s from s_0 in k time-steps under policy π_θ.
- $p^\pi(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(s_0 \to s, k, \pi_\theta)$ is the improper discounted state distribution.
- Let the distribution over all states induced by π_θ be:

$$d^{\pi_\theta}(s) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_0 \to s, t, \pi_\theta).$$

  The factor $(1-\gamma)$ is a normalization constant and this distribution simply discounts states visited later in time.

Overview:

- **On-policy algorithms:** REINFORCE, REINFORCE with baseline and actor-critic algorithms. These algorithms require sampling trajectories with the parameters at each iteration and *then* making updates to the policy.
- **Off-policy algorithms:** Off-policy actor-critic, TRPO, PPO, SAC. These algorithms collect trajectories using a behavior policy which are then used to update potentially different policies.

2.1 Deriving the Policy Gradient

In this note, we will cover episodic tasks. Let $V(\theta) = V_{\pi_\theta}(s_0)$, where s_0 is the starting state (which we consider to be fixed for simplicity) and the dependency on θ specifies that the value depends on the policy π_θ. Policy π_θ can be any distribution. For example, we may use a Gaussian policy and write $\pi_\theta(a|s) = \mathcal{N}(f^\theta_{\text{neural network}}(s), \Sigma)$.

Now, we want to maximize $V(\theta)$:

$$\begin{aligned} V(\theta) &= V_{\pi_\theta}(s_0) \\ &= \sum_a \pi_\theta(a|s_0) Q_{\pi_\theta}(s_0, a) \\ &= \sum_\tau P_\theta(\tau) R(\tau) \end{aligned}$$

where $Q_{\pi_\theta}(s_0, a)$ is the expected reward attained under the policy π_θ after taking action a from state s_0. $P_\theta(\tau)$ is the probability of trajectory $\tau = (s_0, a_0, r_0, s_1, ..., s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ under policy π_θ and $R(\tau)$ is the total reward from trajectory τ.

Lemma 2. $\nabla_\theta V(\theta) = \sum_\tau R(\tau) P_\theta(\tau) \nabla_\theta \log(P_\theta(\tau))$

Proof.

$$\begin{aligned} \nabla_\theta V(\theta) &= \nabla_\theta \sum_\tau P_\theta(\tau) R(\tau) \\ &= \sum_\tau R(\tau) \nabla_\theta P_\theta(\tau) \\ &= \sum_\tau R(\tau) \frac{P_\theta(\tau)}{P_\theta(\tau)} \nabla_\theta P_\theta(\tau) \\ &= \sum_\tau R(\tau) \frac{P_\theta(\tau)}{P_\theta(\tau)} \nabla_\theta P_\theta(\tau) \\ &= \sum_\tau R(\tau) P_\theta(\tau) \nabla_\theta \log(P_\theta(\tau)) \end{aligned}$$

□

Now, using Lemma 2, we see that

$$\nabla_\theta V(\theta) = \mathbb{E}_{P_\theta}[R(\tau) \nabla_\theta \log(P_\theta(\tau))].$$

We can also get an approximation for our update rule:

$$\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log(P_\theta(\tau^{(i)})).$$

However, we still don't know how to compute the gradient of the log-probability. For that, we need the following lemma:

Lemma 3. $\nabla_\theta \log(P_\theta(\tau)) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Proof.

$$\begin{aligned}\nabla_\theta \log\left(P_\theta(\tau)\right. &= \nabla_\theta \log\left(\rho(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)\right) \\ &= \nabla_\theta \log\left(\rho(s_0)\right) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log(\pi_\theta(a_t|s_t))\end{aligned}$$

□

With Lemma 3, we get the following update rule:

$$\nabla_\theta V(\theta) = \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t=1}^{T} r(s_t, a_t) \right) \right].$$

2.2 Policy Gradient Theorem

We generalize this approach through the policy gradient theorem [1]. For an episodic task, let $\eta(s)$ be the number of time-steps spent, on average, in state s within a single episode. If $\mu_0(s)$ is the initial state distribution, then $\eta(s) = \mu_0(s) + \sum_{s'} \eta(s') \sum_a \gamma \cdot \pi_\theta(a|s') p(s|s', a)$. To turn this into a probability distribution, which we call the on-policy distribution i.e the fraction of time-steps spent in a state s, we get $\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}$

Theorem 4. (Policy Gradient Theorem (episodic case)) Suppose, our task is episodic. Furthermore, suppose, our start state is s_0 for all trajectories. Then, $\nabla_\theta V(\theta) \propto \sum_s \mu(s) \sum_a Q_{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a|s)$

or more simply:

$$\begin{aligned}\nabla_\theta V(\theta) &= \sum_s p^{\pi_\theta}(s) \sum_a Q_{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a|s) \\ &= \mathbb{E}_{s \sim p^{\pi_\theta}(s), a \sim \pi_\theta(a|s)} \left[Q_{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a|s) \right]\end{aligned}$$

where $p^{\pi_\theta}(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(s_0 \to s, k, \pi_\theta)$ and $\mathbb{P}(s_0 \to s, k, \pi_\theta)$ is the probability of reaching state s from s_0 in k time-steps under policy π_θ.

Proof.

$$
\begin{aligned}
\nabla_\theta V(\theta) &= \nabla_\theta \left(\sum_a \pi_\theta(a|s) q_{\pi_\theta}(s,a) \right) \\
&= \sum_a \left(\nabla_\theta \pi_\theta(a|s) \right) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \nabla_\theta q_{\pi_\theta}(a,s) \\
&= \sum_a \left(\nabla_\theta \pi_\theta(a|s) \right) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \nabla_\theta \left(\sum_{s',r} p(s',r|s,a) \left(r + \gamma V_{\pi_\theta}(s') \right) \right) \\
&= \sum_a \left(\nabla_\theta \pi_\theta(a|s) \right) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \left(\sum_{s',r} p(s',r|s,a) \gamma \cdot \nabla_\theta \left(V_{\pi_\theta}(s') \right) \right) \\
&= \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \\
&\pi_\theta(a|s) \sum_{s'} p(s'|s,a) \cdot \gamma \cdot \left(\sum_{a'} \left(\nabla_\theta \pi_\theta(a'|s') q_{\pi_\theta}(s',a') \right) + \pi_\theta(a'|s') \sum_{s''} P(s''|s',a') \gamma \cdot \nabla_\theta V_{\pi_\theta}(s'') \right) \\
&= \sum_{x \in S} \sum_{k=0}^{\infty} \mathbb{P}(s_0 \to x, k, \pi_\theta) \gamma^k \sum_a \nabla_\theta \pi_\theta(a|x) q_{\pi_\theta}(x,a)
\end{aligned}
$$

where $\mathbb{P}(s \to x, k, \pi_\theta)$ is the probability of reaching state x from x in k steps under policy π_θ.

$$
\begin{aligned}
\nabla_\theta V(\theta) &= \sum_s \sum_{k=0}^{\infty} \mathbb{P}(s_0 \to s, k, \pi_\theta) \gamma^k \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a)
\end{aligned}
$$

or more simply

$$\begin{aligned}\nabla_\theta V(\theta) &= \sum_s \sum_{k=0}^{\infty} \mathbb{P}(s_0 \to s, k, \pi_\theta)\gamma^k \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\ &= \sum_s p^\pi(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a)\end{aligned}$$

where $p^\pi(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(s_0 \to s, k, \pi_\theta)$ □

With this, we now know how to compute $\nabla_\theta V(\theta)$ using q_{π_θ}. We will now introduce algorithms that approximate q_{π_θ}, allowing us to find a full algorithm for learning the optimal policy.

We derived the policy gradient theorem for the episodic setting. Now we suppose our task is no longer episodic. Again, let us assume that the initial state is fixed s_0. In this case, we define

$$V(\theta) = \lim_{h \to \infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}_{\pi_\theta}[R_t] = \lim_{t \to \infty} \mathbb{E}_{s,a \sim \pi_\theta}[R_t] = \sum_s \mu(s) \sum_a \pi_\theta(a|s) \sum_{s',r} P(s', r|s, a) r.$$

where $\mu(s) := \lim_{t \to \infty} \mathbb{P}(s_t = s | a_{0:t} \sim \pi_\theta)$. We call $\mu(s)$ the steady-state distribution i.e. we say that as time progresses, the probability of being in a state s converges. We assume that this distribution is independent of the initial state, which is called an ergodicity assumption. In particular, if we execute actions by sampling from the policy π_θ, then we remain in this distribution:

$$\sum_s \mu(s) \sum_a \pi_\theta(a \mid s) p(s' \mid s, a) = \mu(s')$$

for all $s' \in \mathcal{S}$.

In the continuing case, we define

$$G_t := R_{t+1} - V(\theta) + R_{t+2} - V(\theta) + \cdots$$

and using this, we define, $V_{\pi_\theta}(s) := \mathbb{E}_{\pi_\theta}[G_t | s_t = s]$, $q_{\pi_\theta}(s,a) = \mathbb{E}_{\pi_\theta}[G_t | s_t = s, a_t = a]$.

We prove the following theorem, which is analogous to the episodic setting:

Theorem 5. (Policy Gradient Theorem (continuous case)) Suppose, our task is continuous. Furthermore, suppose our start state is s_0 for all trajectories. Then,

$$\nabla_\theta V(\theta) \propto \sum_s \mu(s) \sum_a q_{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a|s).$$

Proof.

$$\begin{aligned}
\nabla_\theta V_{\pi_\theta}(s) &= \nabla \left[\sum_a \pi_\theta(a|s) q_{\pi_\theta}(s,a) \right] \\
&= \sum_a \left[\nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \nabla_\theta q_{\pi_\theta}(s,a) \right] \\
&= \sum_a \left[\nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \nabla_\theta \sum_{s',r} P(s',r|s,a)(r - V(\theta) + V_{\pi_\theta}(s')) \right] \\
&= \sum_a \left[\nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) [-\nabla_\theta V(\theta) + \sum_{s'} P(s'|s,a) \nabla_\theta V_{\pi_\theta}(s')] \right] \\
\nabla_\theta V(\theta) &= \sum_a \left[\nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V_{\pi_\theta}(s') \right] - \nabla_\theta V_{\pi_\theta}(s)
\end{aligned}$$

Now, the left hand side is independent of s, so we can do a weighted sum over s (weighted by $\mu(s)$:

$$\begin{aligned}
\sum_s \mu(s) \nabla_\theta V(\theta) &= \sum_s \mu(s) \left(\sum_a \left[\nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V_{\pi_\theta}(s') \right] - \nabla_\theta V_{\pi_\theta}(s) \right) \\
\nabla_\theta V(\theta) &= \sum_s \mu(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \\
&\quad \sum_s \mu(s) \sum_a \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V_{\pi_\theta}(s') - \sum_s \mu(s) \nabla_\theta V_{\pi_\theta}(s) \\
&= \sum_s \mu(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \\
&\quad \sum_{s'} \sum_s \mu(s) \sum_a \pi_\theta(a|s) P(s'|s,a) \nabla_\theta V_{\pi_\theta}(s') - \sum_s \mu(s) \nabla_\theta V_{\pi_\theta}(s) \\
&= \sum_s \mu(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \sum_{s'} \mu(s') \nabla_\theta V_{\pi_\theta}(s') - \sum_s \mu(s) \nabla_\theta V_{\pi_\theta}(s) \\
&= \sum_s \mu(s) \sum_a q_{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a|s)
\end{aligned}$$

where in the second last line we used the fact that $\mu(s') = \sum_s \mu(s) \sum_a \pi_\theta(a|s) P(s'|s,a)$. □

I find this to be a very interesting result - the fact that the same result holds for both episoding and non-episodic tasks after the simple, understandable re-definition of G_t. In fact, the new way of seeing G_t is akin to the introduction of baselines in methods like REINFORCE.

2.3 REINFORCE

Using our derivation, we have our first policy gradient algorithm. So far, we have:

$$\nabla_\theta V(\theta) = \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t'=1}^{T} r(s_{t'}, a_{t'}) \right) \right].$$

REINFORCE is an on-policy algorithm that approximates this by sampling multiple trajectories rolled out by policy π_θ and then letting:

$$\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t'=1}^{T} r(s_{i,t'}, a_{i,t'}) \right)$$

Interpretation: One way to interpret this update rule is to note that we are maximizing the log-likelihood of each trajectory sampled - except we are weighing them by their returns. In other words, if a trajectory yields higher returns, we are increasing the log-likelihood of that with a larger weight than one that yields lower returns.

There are a couple of shortcomings of this approach:

1. Since this is an on-policy method, we require sampling multiple trajectories for each update to the policy.
2. The algorithm has high variance since the trajectory samples are often quite noisy (especially in most real-world environments). More precisely, $\sum_{t'=1}^{T} r(s_{i,t'}, a_{i,t'})$ has high variance.

We will next introduce a couple of algorithms that are aimed towards solving these issues.

2.4 REINFORCE using causality

The update rule we derived so far is

$$\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t'=1}^{T} r(s_{i,t'}, a_{i,t'}) \right).$$

Now, notice that the policy at time t cannot affect the rewards at time $t' < t$. In other words, when we maximize the log-likelihood of taking a particular action $a_{i,t}$ from a state $s_{i,t}$, should we really weigh it by the reward attained from the entire trajectory? Intuitively, it makes more sense to weigh it by the rewards attained from that time t onward in the trajectory i since only

the rewards attained *after* executing $a_{i,t}$ gives us a signal for how good the action is. With this in mind, our first modification is as follows:

$$\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right).$$

The term $\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})$ is called reward-to-go. In other words, we are using the "reward to go" from time t.

Algorithm: REINFORCE (with causality)

Initialize policy parameters θ
for iteration = 1, 2, ... **do**
  Collect a set of trajectories $\{\tau^i\}$ by running the policy $\pi_\theta(a_t|s_t)$
  for each trajectory τ^i **do**
    for each timestep t in τ^i **do**
      Compute return: $G_t^i = \sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i)$
    end for
  end for
  Update the policy parameters:
    $\nabla_\theta V(\theta) \approx \sum_i \sum_t \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) G_t^i$
    $\theta \leftarrow \theta + \alpha \nabla_\theta V(\theta)$
end for

Intuition: we are updating the policy parameters by taking a step in the direction of $\nabla_\theta V(\theta) \approx \sum_i \sum_t (\nabla_\theta \log \pi_\theta) G_t^i$. Focus on the term $\sum_i \sum_t (\nabla_\theta \log \pi_\theta)$. A step in this direction is essentially maximizing the probability of $A = a \mid s$ under π_θ, which is what maximum likelihood estimation does! Since this is scaled by G_t^i, it takes a step in a direction that is closed aligned with high G_t^i actions i.e., actions that we expect to give higher returns.

2.5 REINFORCE with baseline

We introduce a baseline to further reduce variance:

$$\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) - b(s_{i,t'}) \right).$$

where b is any arbitrary function as long as it does not depend on a_t. We usually select $b(s) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau) \mid s_0 = s]$. We could also use a learned state-action value function parametrized by ϕ,

$$v_\phi(s_t) =: b(s_t).$$

Note that in the policy gradient theorem, this amounts to modifying the update to be :

$$\nabla_\theta V(\theta) \propto \sum_s \mu(s) \sum_a (Q_{\pi_\theta(s,a)} - b(s_t)) \nabla_\theta \pi_\theta(a|s)$$

Proposition 6. Subtracting the baseline b is unbiased in expectation i.e.,

$$\mathbb{E}[\nabla_\theta \log P_\theta(\tau) b] = 0.$$

Proof.

$$\begin{aligned}
&\mathbb{E}[\nabla_\theta \log P_\theta(\tau) b] \\
&= \int P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) b d\tau \\
&= \int \nabla_\theta P_\theta(\tau) b d\tau \\
&= b \nabla_\theta \int P_\theta(\tau) d\tau \\
&= b \nabla_\theta (1) \\
&= b \cdot 0 \\
&= 0.
\end{aligned}$$

□

We can actually find the baseline that reduces the variance the most in a principled manner:

Proposition 7. $b = \frac{\mathbb{E}[g(\tau)^2 r(\tau)]}{\mathbb{E}[g(\tau)^2]}$ minimizes the variance:

$$\text{Var} := \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[(\nabla_\theta \log P_\theta(\tau)(r(\tau) - b))^2 \right] - \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\nabla_\theta \log P_\theta(\tau)(r(\tau) - b) \right]^2 .$$

Here $g(\tau)$ is the gradient $\nabla_\theta \log P_\theta(\tau)$.

Proof. Take the derivative with respect to the baseline b and, after a little bit of algebra, you get the desired expression. □

In other words, the optimal baseline is the expected reward but weighted by the magnitude of our gradients. In practice, we end up just using the expected reward as the baseline.

Algorithm: REINFORCE with baseline

Initialize policy parameters θ
for iteration = 1, 2, ... **do**
  Collect a set of trajectories $\{\tau^i\}_{i=1}^N$ by running the policy $\pi_\theta(a_t|s_t)$
  for each trajectory τ^i **do**
    for each timestep t in τ^i **do**
      Compute return: $G_t^i = \sum_{t'=t}^T r(s_{t'}^i, a_{t'}^i)$
    end for
  end for
  Compute $b = \frac{1}{N}\sum_{i=1}^N r(\tau^i)$
  Update the policy parameters:
    $\nabla_\theta V(\theta) \approx \sum_i \sum_t \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)(G_t^i - b)$
    $\theta \leftarrow \theta + \alpha \nabla_\theta V(\theta)$
end for

2.6 On-Policy Actor-Critic Methods

Actor-Critic methods replace the reward to go $\sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})$ with $Q_\theta(s_{i,t'}, a_{i,t'})$. As for the baseline, we use $V_\theta(s_{i,t'})$. Then, we can replace $Q_\theta(s_{i,t'}, a_{i,t'}) - V_\theta(s_{i,t'})$ with the advantage function $A^{\pi_\theta}(s_{i,t'}, a_{i,t'})$. The better the estimate of this advantage, the lower the variance.

Actor-critic methods use a learned value function $V_\phi^{\pi_\theta}(s_t)$ to approximate $V_\theta(s_t)$. This is trained using supervised regression. Suppose our training set (using the rollouts by policy π_θ) is of the form $\{(s_{i,t}, \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'}))\}$. Then, we train via minimizing

$$\frac{1}{2}\sum_i \sum_t \left|\left| V_\phi^{\pi_\theta}(s_{i,t}) - \left(\sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})\right)\right|\right|^2.$$

Alternatively, we can also train this by using a boostrapped estimate of the target:

$$\frac{1}{2}\sum_i \sum_t \left|\left| V_\phi^{\pi_\theta}(s_{i,t}) - \left(r(s_{i,t}, a_{i,t}) + V_\phi^{\pi_\theta}(s_{i,t+1})\right)\right|\right|^2.$$

2.7 Making inroads to off-policy policy gradient

Suppose, we collected trajectories using policy π_θ and we know the returns from these trajectories. Can we use them to make updates to a different policy $\pi_{\theta'}$?

Given policy $\pi_{\theta'}$, we want to maximize $V(\theta') = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}[r(\tau)]$. Using importance-samling, we can write:

$$V(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau)\right]$$
$$\nabla_\theta V(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} \nabla_{\theta'} \log(p_{\theta'}(\tau)) r(\tau)\right]$$

Then, using the full expression for the probability of any trajectory, we get

$$\begin{aligned}\nabla_\theta V(\theta) &= \mathbb{E}_{\tau \sim p_\theta}\left[\prod_{t=1}^{T} \frac{\pi_{\theta'}(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} \left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t \mid s_t)\right) r(\tau)\right] \\ &= \mathbb{E}_{\tau \sim p_\theta}\left[\prod_{t=1}^{T} \frac{\pi_{\theta'}(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} \left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t \mid s_t)\right) \left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]\end{aligned}$$

Rewrite this:

$$\begin{aligned}&\nabla_\theta V(\theta) \\ &= \mathbb{E}_{\tau \sim p_\theta}\left[\left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t \mid s_t)\right) \left(\prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'} \mid s_{t'})}{\pi_\theta(a_{t'} \mid s_{t'})}\right) \left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \left(\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(a_{t''} \mid s_{t''})}{\pi_\theta(a_{t''} \mid s_{t''})}\right)\right)\right]\end{aligned}$$

And then we get, using causality:

$$\nabla_\theta V(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t \mid s_t)\right) \left(\prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'} \mid s_{t'})}{\pi_\theta(a_{t'} \mid s_{t'})}\right) \left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\right)\right]$$

2.8 Off-Policy Actor-Critic

This section is not really new - we just use importance sampling. The derivation will assume continuous state and action spaces. We want to estimate the policy gradient *off-policy* from trajectories sampled from a distinct behaviour policy $\beta(a|s) \neq \pi_\theta(a|s)$. In this setting, the performance objective is the value function of the target policy, π_θ, averaged over the state

distribution of the behaviour policy $\beta(a|s)$:

$$J_\beta(\pi_\theta) = \int_{\mathcal{S}} p^\beta(s) V^{\pi_\theta}(s) = \int_{\mathcal{S}} \int_{\mathcal{A}} p^\beta(s) \pi_\theta(a|s) Q^{\pi_\theta}(s,a) da ds.$$

The policy gradient becomes (after approximating by dropping $\nabla_\theta Q^{\pi_\theta}(s,a)$):

$$\nabla_\theta J_\beta(\pi_\theta) = \mathbb{E}_{s\sim p^\beta, a\sim\beta}\left[\frac{\pi_\theta(a|s)}{\beta(a|s)} \nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right].$$

OffPAC (Off-Policy Actor Critic) algorithm uses the behavior policy $\beta(a|s)$ to generate trajectories. A critic estimates $V^\phi(s) \approx V^{\pi_\theta}(s)$ off-policy by gradient temporal-difference learning. Instead of the unknown $Q^{\pi_\theta}(s,a)$, the temporal-difference error $\delta_t = r_{t+1} + \gamma V^\phi(s_{t+1}) - V^\phi(s_t)$ is used.

2.9 Soft Actor-Critic

Soft Actor-Critic is an off-policy actor-critic algorithm that aims to learn a policy that maximizes rewards while also acting as stochastically as possible. In other words, if there are two actions that both achieve the same maximum rewards, our goal would be to assign nearly equal probability mass to both of them. This falls under a general framework called maximum entropy reinforcement learning where the goal is to maximize (assume a finite horizon task with discount factor $\gamma = 0$):

$$J(\theta) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta}\left[r(s_t,a_t) + \alpha H(\pi(\cdot \mid s_t))\right]$$

where $H(p)$ is the entropy of the distribution p.

We parametrize three function: $V_\psi(s_t)$, $Q_\phi(s_t,a_t)$ and $\pi_\theta(a_t|s_t)$. Although technically we do not need to have separate function approximators for V and Q, doing this improves training stability.

The soft value function is trained to minimize the following loss:

$$J_V(\psi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\psi(s_t) - \mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[Q_\phi(s_t,a_t) - \log \pi_\theta(a_t|s_t)\right]\right)^2\right].$$

Here $\mathcal{D}$ is the replay buffer. The gradient is computed as

$$\nabla_\psi J_V(\psi) = \nabla_\psi V_\psi(s_t)(V_\psi(s_t) - Q_\phi(s_t,a_t) + \log \pi_\theta(a_t|s_t))$$

where the actions are sampled $a_t \sim \pi_\theta(\cdot|s_t)$.

The soft Q-function is trained to minimize the soft Bellman residual:

$$J_Q(\phi) = \mathbb{E}_{s_t, a_t \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t) \right)^2 \right]$$

where $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} \left[V_{\bar{\psi}}(s_{t+1}) \right]$. Here $V_{\bar{\psi}}$ is our target network where $\bar{\psi}$ is an exponentially moving average of the value network weights. The gradient of this is:

$$\nabla_\phi (J_Q(\phi)) = \nabla_\phi Q_\phi(s_t, a_t)(Q_\phi(s_t, a_t) - r(s_t, a_t) - \gamma V_{\bar{\psi}}(s_{t+1})).$$

Now, to parametrize the policy $\pi_\theta(a_t|s_t)$, represent the action as $a_t = f_\theta(\epsilon_t : s_t)$ where ϵ_t is an input random noise vector sampled from a fixed distribution (like spherical Gaussian). Then, the policy is optimized by minimizing:

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{S}} \left[\log \pi_\theta(f_\theta(\epsilon_t; s_t) \mid s_t) - Q_\phi(s_t, f_\theta(\epsilon_t; s_t)) \right].$$

The unbiased gradient of this is

$$\nabla_\theta J_\pi(\theta) = \nabla_\theta \pi_\theta(a_t|s_t) + (\nabla_{a_t} \log \pi_\theta(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_\theta f_\theta(\epsilon_t; s_t).$$

2.10 Performance Difference Lemma

We want to prove a few results that are building blocks for trust regional policy optimisation.

Firstly, note that the probability of sampling any particular trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, ...)$ is $P_\theta(\tau) = \prod_{i=0}^{\infty} \pi_\theta(a_i|s_i) P(s_{i+1}|s_i, a_i)$. The probability of sampling a particular trajectory τ such that $s_t = s$ is $P_\theta(s_t = s) = \sum_{a_0} \sum_{s_1} ... \sum_{s_{t-1}} \sum_{a_{t-1}} \left(\prod_{i=1}^{T-2} \pi_\theta(a_i|s_i) P(s_{i+1}|s_i, a_i) \right) \pi_\theta(a_{t-1} \mid s_{t-1}) P(s_t = s \mid s_{t-1}, a_{t-1})$.

Let the distribution over all states induced by π_θ be:

$$d^{\pi_\theta}(s) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P_\theta(s_t = s).$$

The factor $(1 - \gamma)$ is a normalization constant and this distribution simply discounts states visited later in time.

Lemma 8. $\mathbb{E}_{\tau \sim P_\theta} \left[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \right] = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta}} \left[\mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \left[f(s, a) \right] \right]$

Proof.

$$\mathbb{E}_{\tau \sim P_\theta} \left[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \right] = \sum_{\tau} P_\theta(\tau) \sum_{t=0}^{\infty} \gamma^t f(s_t, a_t)$$

$$
\begin{aligned}
&= \sum_{\tau} \prod_{i=0}^{\infty} \pi_\theta(a_i|s_i) P(s_{i+1}|s_i, a_i) \sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \\
&= \sum_{a_0} \sum_{s_1} \sum_{a_1} \cdots \prod_{i=0}^{\infty} \pi(a_i|s_i) P(s_{i+1}|s_i, a_i) \sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \\
&= \sum_{a_0} \pi_\theta(a_0|s_0) f(s_0, a_0) + \gamma \sum_{a_0} \sum_{s_1} \sum_{a_1} \pi_\theta(a_0|s_0) P(s_1|s_0, a_0) \pi_\theta(a_1|s_1) f(s_1, a_1) + \cdots \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{a_0} \sum_{s_1} \sum_{a_1} \cdots \sum_{s_t} \sum_{a_t} \sum_{s_{t+1}} \prod_{i=0}^{t} \pi_\theta(a_i|s_i) P(s_{i+1}|s_i, a_i) f(s_t, a_t) \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1}} (\cdots) f(s_t, a_t) \\
&= \sum_{t=0}^{\infty} \gamma^t (\cdots) f(s_t, a_t) \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{s_t} \sum_{a_t} \left(\sum_{a_0} \cdots \sum_{s_{t-1}} \sum_{a_{t-1}} \prod_{i=0}^{t-1} \pi_\theta(a_i|s_i) P(s_{i+1}|s_i, a_i) \right) \pi_\theta(a_t|s_t) f(s_t, a_t) \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{s_t} \sum_{a_t} P_\theta(s_t) \pi_\theta(a_t|s_t) f(s_t, a_t) \\
&= \frac{1}{1-\gamma} (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{s_t} P_\theta(s_t) \sum_{a_t} \pi_\theta(a_t|s_t) f(s_t, a_t) \\
&= \frac{1}{1-\gamma} (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{s_t} P_\theta(s_t) \mathbb{E}_{a \sim \pi_\theta} [f(s_t, a)] \\
&= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta}} [\mathbb{E}_{a \sim \pi_\theta} [f(s_t, a)]]
\end{aligned}
$$

□

Theorem 9. (Performance Difference Lemma)

$$
V_\pi(s_0) - V_{\pi'}(s_0) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^\pi} \left[\mathbb{E}_{a \sim \pi(s)} \left[A^{\pi'}(s, a) \right] \right]
$$

where the advantage function is $A^\pi(s, a) := q^\pi(s, a) - V^\pi(s)$.

Proof.

$$
\begin{aligned}
&V_\pi(s_0) - V_{\pi'}(s_0) \\
&= \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right] + \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^{t+1}V_{\pi'}(s_{t+1})\right] - \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^{t+1}V_{\pi'}(s_{t+1})\right] - V_{\pi'}(s_0) \\
&= \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t)+\gamma V_{\pi'}(s_{t+1})) - \sum_{t=0}^{\infty}\gamma^{t+1}V_{\pi'}(s_{t+1}) - V_{\pi'}(s_0)\right]
\end{aligned}
$$

Now, we expand the first term:

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t)+\gamma V_{\pi'}(s_{t+1})\right)\right] \\
&= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\tau\sim P^\pi}\left[R(s_t,a_t)+\gamma V^{\pi'}(s_{t+1}|s_t,a_t)\right] \\
&= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s_t\sim P^\pi}\left[\mathbb{E}_{a_t\sim P^\pi}\left[\mathbb{E}_{s_{t+1}\sim P^\pi}\left[R(s,a)+\gamma V^{\pi'}(s_{t+1})|s=s_t,a=a_t\right]|a=a_t\right]|s=s_t\right] \\
&= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s_t,a_t\sim P^\pi}\left[R(s,a)+\gamma\sum_{s_{t+1}}P(s_{t+1}|s_t,a_t)V^{\pi'}(s_{t+1})|s=s_t,a=a_t\right] \\
&= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\tau\sim P^\pi}\left[Q^{\pi'}(s,a)|s=s_t,a=a_t\right] \\
&= \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^t Q^{\pi'}(s_t,a_t)\right]
\end{aligned}
$$

Therefore, we get

$$
\begin{aligned}
V^\pi(s_0) - V^{\pi'}(s_0) &= \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(Q^{\pi'}(s_t,a_t) - V^{\pi'}(s_t)\right)\right] \\
&= \mathbb{E}_{\tau\sim P^\pi}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi'}(s_t,a_t)\right] \\
&= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^\pi}\left[\mathbb{E}_{a\sim\pi}\left[A^{\pi'}(s,a)\right]\right]
\end{aligned}
$$

□

Sometimes, as in [2], this lemma has the following equivalent expression:

Proposition 10. (Performance Difference Lemma (2)) Given two policies π and π',

$$V(\pi') - V(\pi) = \mathbb{E}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right].$$

Proof. We write:

$$A_\pi(s, a) = \mathbb{E}_{s' \in P(s'|s,a)} \left[r(s, a) + \gamma V_\pi(s') - V_\pi(s) \right].$$

Then,

$$\begin{aligned}
\mathbb{E}_{\pi'} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right] &= \mathbb{E}_{\pi'} \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t) \right) \right] \\
&= \mathbb{E}_{\pi'} \left[-V_\pi(s_0) + \sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) \right) \right] \\
&= -\mathbb{E}_{s_0 \sim \rho_0}[V_\pi(s_0)] + \mathbb{E}_{\pi'}[\sum_{t=0}^{\infty} \gamma^t r_t] \\
&= -V(\pi) + V(\pi').
\end{aligned}$$

□

2.11 Covariant/natural policy gradient

So far, our update rules were aimed towards computing

$$\theta \leftarrow \theta + \alpha \nabla_\theta V(\theta)$$

to update the policy $\pi_\theta(a \mid s)$. However, controlling the learning rate to maximize $V(\theta)$ is non-trivial. Some parameters of your policy ultimately end up affecting $V(\theta)$ more than others and choosing one constant α that controls the learning rate for all the parameters is difficult. We would want to have higher learning rates for parameters that do not change the policy very much and smaller learning rates for those that do.

Now, notice that using a first-order Taylor expansion, we can write:

$$\arg\max_{\theta'} V(\theta') \approx \arg\max_{\theta'} V(\theta) + (\theta' - \theta)^T \nabla_\theta V(\theta).$$

So, we aim to solve $\arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta V(\theta)$ such that $||\theta' - \theta|| \leq \epsilon$ (so that the Taylor approximation is valid). We can reframe this problem in the policy space: we aim to solve $\arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta V(\theta)$ such that $D(\pi_{\theta'} \,||\, \pi_\theta) \leq \epsilon$ where D is a divergence-measure [3].

We can choose D to be the KL-divergence. In this case, we can approximate (using the second-order Taylor approximation) $D_{KL}(\pi_{\theta'} \mid\mid \pi_\theta) \approx (\theta' - \theta)^T F(\theta' - \theta)$ where F is the Fisher information matrix, i.e. $F = \mathbb{E}_{\pi_\theta}[(\nabla_\theta \log \pi_\theta(a \mid s)) (\nabla_\theta \log \pi_\theta(a \mid s))^T]$. Then, the problem becomes: $\arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta V(\theta)$ such that $||\theta' - \theta||_F^2 \leq \epsilon$. Then, the update rule becomes

$$\theta \leftarrow \theta + \alpha F^{-1} \nabla_\theta V(\theta)$$

2.12 Trust Region Policy Optimization (TRPO)

Let $V(\pi) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_t]$. Recall: $Q_\pi(s_t, a_t) = \mathbb{E}_\pi[\sum_{l=0}^\infty \gamma^l r_{t+l} \mid s_t, a_t]$, $V_\pi(s_t) = \mathbb{E}_\pi[\sum_{l=0}^\infty \gamma^l r_{t+l} \mid s_t]$ and $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$ where $a_t \sim \pi(a_t \mid s_t)$.

Recall the improper discounted state distribution $p^\pi(s) = \sum_{k=0}^\infty \gamma^k \mathbb{P}(s_0 \to s, k, \pi)$. Using this, we can write the performance difference lemma as:

$$\begin{aligned} V(\pi') - V(\pi) &= \sum_{t=0}^\infty \sum_s P(s_t = s \mid \pi') \sum_a \pi'(a \mid s) \gamma^t A_\pi(s, a) \\ &= \sum_s \sum_{t=0}^\infty \gamma^t P(s_t = s \mid \pi') \sum_a \pi'(a \mid s) A_\pi(s, a) \\ &= \sum_s p^{\pi'}(s) \sum_a \pi'(a \mid s) A_\pi(s, a) \end{aligned}$$

Note: $V(\pi') - V(\pi) \geq 0$ if, at every state s, we have that $\sum_a \pi'(a \mid s) A_\pi(s, a) \geq 0$.

Now, the first building block of TRPO is the local approximation of this as

$$L_\pi(\pi') = V(\pi) + \sum_s p^\pi(s) \sum_a \pi'(a \mid s) A_\pi(s, a)$$

where we have replaced $p^{\pi'}$ with $p^{\pi'}$. Note that when our policy $\pi_\theta(a \mid s)$ is differentiable, then, for the parameters θ_0, we have that $L_{\pi_{\theta_0}}(\pi_{\theta_0}) = V(\pi_{\theta_0})$ and $\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)|_{\theta=\theta_0} = \nabla_\theta V(\pi_\theta)|_{\theta=\theta_0}$. Therefore, if the update $\pi_{\theta_0} \to \pi'$ is small enough such that $L_{\pi_{\theta_0}}(\pi_{\theta_0})$ improves, then we see an improvement in the value V as well. However, controlling the learning rate that ensures this is difficult. TRPO aims to solve this issue.

The guiding principal comes from the following theorem [2]:

Theorem 11. Let $\alpha = D_{\text{TV}}^{\max}(\pi_{\text{old}}, \pi_{\text{new}}) = \max_s D_{\text{TV}}(\pi_{\text{old}}(\cdot \mid s) || \pi_{\text{new}}(\cdot \mid s))$. Then,

$$V(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} \alpha^2$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

The proof can be found in [2]. The main building blocks are the following two definitions and lemma: To

Definition 6. (α-coupled policy pair). We call (π, π') an α-coupled policy pair if the joint distribution $(a, a') \mid s$ is such that $\mathbb{P}(a \neq a' \mid s) \leq \alpha$ for all s.

We also define

$$\bar{A}(s) = \mathbb{E}_{a \sim \pi'(\cdot|s)} \left[A_\pi(s, a) \right].$$

Then, we have the following lemma:

Lemma 12. Let (π, π') be an α-coupled policy pair. Then, for all states s,

$$|\bar{A}(s)| \leq 2\alpha \max_{s,a} |A_\pi(s, a)|.$$

Proof.

$$\begin{aligned}
\bar{A}(s) &= \mathbb{E}_{a' \sim \pi'}[A_\pi(s, a')] \\
&= \mathbb{E}_{(a,a') \sim (\pi, \pi')}[A_\pi(s, a') - A_\pi(s, a)]
\end{aligned}$$

as $\mathbb{E}_{a \sim \pi}[A_\pi(s, a)] = 0$. Then, continuing:

$$\begin{aligned}
\bar{A}(s) &= \mathbb{E}_{(a,a') \sim (\pi, \pi')}[A_\pi(s, a') - A_\pi(s, a)] \\
&= \sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)(A_\pi(s, a') - A_\pi(s, a)) \\
|\bar{A}(s)| &= |\sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)(A_\pi(s, a') - A_\pi(s, a))| \\
&\leq |\sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)(A_\pi(s, a'))| + |\sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)(A_\pi(s, a))| \\
&\leq |\sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)(\max_{s,a} A_\pi(s, a))| + |\sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)(\max_{s,a} A_\pi(s, a))| \\
&= |\max_{s,a} A_\pi(s, a)| 2 |\sum_{a'} \pi'(a' \mid s) \sum_{a \neq a'} \pi(a \mid s)| \\
&= 2\alpha |\max_{s,a} A_\pi(s, a)| \\
&\leq 2\alpha \max_{s,a} |A_\pi(s, a)|
\end{aligned}$$

□

Lemma 13. Let (π, π') be an α-coupled policy pair. Then,

$$|\mathbb{E}_{s_t \sim \pi'} \left[\bar{A}(s_t) \right] - \mathbb{E}_{s_t \sim \pi} \left[\bar{A}(s_t) \right]| \leq 2\alpha \max_s |\bar{A}(s)| \leq 4\alpha(1 - (1 - \alpha)^t) \max_s |A_\pi(s, a)|.$$

Proof. Let n_t be the number of time steps such that $a_i \neq a_i'$ for $i < t$ and $a_i \sim \pi, a_i' \sim \pi'$. This denotes the number of times π and π' take different actions before time step t. Then,

$$\mathbb{E}_{s_t\sim\pi'}\left[\bar{A}(s_t)\right] = P(n_t = 0)\mathbb{E}_{s_t\sim\pi'|n_t=0}\left[\bar{A}(s_t)\right] + P(n_t > 0)\mathbb{E}_{s_t\sim\pi'|n_t>0}\left[\bar{A}(s_t)\right].$$

Similarly,

$$\mathbb{E}_{s_t\sim\pi}\left[\bar{A}(s_t)\right] = P(n_t = 0)\mathbb{E}_{s_t\sim\pi|n_t=0}\left[\bar{A}(s_t)\right] + P(n_t > 0)\mathbb{E}_{s_t\sim\pi|n_t>0}\left[\bar{A}(s_t)\right].$$

Now,

$$\mathbb{E}_{s_t\sim\pi'|n_t=0}\left[\bar{A}(s_t)\right] = \mathbb{E}_{s_t\sim\pi|n_t=0}\left[\bar{A}(s_t)\right]$$

since $n_t = 0 \implies \pi$ and π' executed the same actions on all time steps less than t. Then,

$$\begin{aligned}
&\mathbb{E}_{s_t\sim\pi'}\left[\bar{A}(s_t)\right] - \mathbb{E}_{s_t\sim\pi}\left[\bar{A}(s_t)\right] \\
&= P(n_t > 0)\left(\mathbb{E}_{s_t\sim\pi'|n_t>0}\left[\bar{A}(s_t)\right] - \mathbb{E}_{s_t\sim\pi|n_t>0}\left[\bar{A}(s_t)\right]\right).
\end{aligned}$$

Now, since π and π' are α-coupled, then, $\mathbb{P}(a = a' \mid s) \geq 1 - \alpha$, so $P(n_t = 0) \geq (1-\alpha)^t$ and

$$P(n_t > 0) \leq 1 - (1-\alpha)^t.$$

On the other hand, using triangle inequality,

$$\begin{aligned}
&|\mathbb{E}_{s_t\sim\pi'|n_t>0}\left[\bar{A}(s_t)\right] - \mathbb{E}_{s_t\sim\pi|n_t>0}\left[\bar{A}(s_t)\right]| && (2)\\
&\leq |\mathbb{E}_{s_t\sim\pi'|n_t>0}\left[\bar{A}(s_t)\right]| + |\mathbb{E}_{s_t\sim\pi|n_t>0}\left[\bar{A}(s_t)\right]| && (3)\\
&\leq 2\max_s |\bar{A}(s)| && (4)\\
&\leq 4\alpha \max_{s,a} |A_\pi(s,a)| && (5)\\
& && (6)
\end{aligned}$$

Then,

$$\begin{aligned}
&|\mathbb{E}_{s_t\sim\pi'}\left[\bar{A}(s_t)\right] - \mathbb{E}_{s_t\sim\pi}\left[\bar{A}(s_t)\right]| \\
&= |P(n_t > 0)\left(\mathbb{E}_{s_t\sim\pi'|n_t>0}\left[\bar{A}(s_t)\right] - \mathbb{E}_{s_t\sim\pi|n_t>0}\left[\bar{A}(s_t)\right]\right)| \\
&\leq 4\alpha(1-(1-\alpha)^t)\max_{s,a}|A_\pi(s,a)|.
\end{aligned}$$

□

Now we prove theorem 11:

Proof. Denote $\pi = \pi_{\text{old}}$ and $\pi' = \pi_{\text{new}}$. Let $\epsilon = \max_{s,a} |A_\pi(s,a)|$. Then, using performance difference lemma and the definition of $\bar{A}(s)$:

$$\begin{aligned} V(\pi') - V(\pi) &= \mathbb{E}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A_\pi(s_t,a_t)\right] \\ &= \mathbb{E}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t \bar{A}(s_t)\right] \end{aligned}$$

On the other hand,

$$\begin{aligned} L_\pi(\pi') &= V(\pi) + \mathbb{E}_{s\sim p^\pi, a\sim\pi'}[A_\pi(s,a)] \\ &= V(\pi) + \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t \bar{A}(s_t)\right]. \end{aligned}$$

Combining:

$$\begin{aligned} |V(\pi') - L_\pi(\pi')| &\le \sum_{t=0}^{\infty}\gamma^t \left|\mathbb{E}_{\tau\sim\pi'}[\bar{A}(s_t)] - \mathbb{E}_{\tau\sim\pi}[\bar{A}(s_t)]\right| \\ &\sum_{t=0}^{\infty}\gamma^t 4\epsilon\alpha(1-(1-\alpha)^t) \\ &= 4\epsilon\alpha\left(\frac{1}{1-\gamma} - \frac{1}{1-\gamma(1-\alpha)}\right) \\ &= \frac{4\alpha^2\gamma\epsilon}{(1-\gamma)(1-\gamma(1-\alpha))} \\ &= \frac{4\alpha^2\gamma\epsilon}{(1-\gamma)(1-\gamma(1-\alpha))} \\ &\le \frac{4\alpha^2\gamma\epsilon}{(1-\gamma)^2}. \end{aligned}$$

Now, if we have two policies π and π' such that $\max_s D_{\text{TV}}(\pi(\cdot \mid s)||\pi'(\cdot \mid s)) \le \alpha$, then we can define an α-coupled policy with the appropriate marginals. Then, take $\alpha = \max_s D_{\text{TV}}(\pi(\cdot \mid s)||\pi'(\cdot \mid s))$, plug this into $\frac{4\alpha^2\gamma\epsilon}{(1-\gamma)^2}$ to conclude. □

From theorem 11, by noting that $D_{\text{TV}}(p||q)^2 \le D_{\text{KL}}(p||q)$, we get that

$$V(\pi') \ge L_\pi(\pi') - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{\text{KL}}^{\max}(\pi,\pi').$$

Using this, we can find a preliminary algorithm:

Algorithm: Policy iteration guaranteeing non-decreasing expected return V

1: **Initialize** policy π_0
2: **for** $i = 0, 1, 2, \ldots$ until convergence **do**
3: Compute all advantage values $A_{\pi_i}(s, a)$
4: Solve the constrained optimization problem:
$\pi_{i+1} = \arg\max_\pi \left[L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right]$
where $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$
and $L_{\pi_i}(\pi) = V(\pi_i) + \sum_s p^{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$
5: **end for**

This algorithm has guaranteed monotonic improvement. We can now use it to derive TRPO. Let $V(\theta) := V(\pi_\theta)$, $L_\theta(\theta') := L_{\pi_\theta}(\pi_{\theta'})$, and $D_{\mathrm{KL}}(\theta \,\|\, \theta') := D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\theta'})$. We saw that:

$$V(\theta) \geq L_{\theta_{\mathrm{old}}}(\theta) - C D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta)$$

with equality at $\theta = \theta_{\mathrm{old}}$. Thus, we can improve $V(\theta)$ by solving the following optimization problem:

$$\max_\theta \; L_{\theta_{\mathrm{old}}}(\theta) - C D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta)$$

However, choosing the step size here is tricky - if we use $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$, then the step sizes become small. Instead, we solve the following with a trust region constraint:

$$\max_\theta \; L_{\theta_{\mathrm{old}}}(\theta) \tag{7}$$

$$\text{subject to } D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta) \leq \delta \tag{8}$$

In practice, we use the average KL divergence: $\bar{D}_{\mathrm{KL}}^{p^{\theta_{\mathrm{old}}}}(\theta_{\mathrm{old}}, \theta) = \mathbb{E}_{s \sim p^{\theta_{\mathrm{old}}}}[D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(\cdot \mid s) || \pi_\theta(\cdot \mid s))]$

Lastly, we show how to estimate the objective and constraint functions using Monte Carlo simulation. We replace the objective $\sum_s p^{\theta_{\mathrm{old}}}(s) \sum_a \pi_\theta(a \mid s) A_{\theta_{\mathrm{old}}}(s, a)$ with $\frac{1}{1-\gamma} \mathbb{E}_{s \sim \theta_{\mathrm{old}}, a \sim \pi_\theta(\cdot|s)}[A_{\theta_{\mathrm{old}}}(s, a)]]$. Next, we replace $A_{\theta_{\mathrm{old}}}(s, a)$ with $Q_{\theta_{\mathrm{old}}}(s, a)$. Lastly, we use importance sampling. Altogether, we have:

$$\max_\theta \mathbb{E}_{s \sim \rho_{\theta_{\mathrm{old}}}, a \sim q} \left[\frac{\pi_\theta(a \mid s)}{q(a \mid s)} Q_{\theta_{\mathrm{old}}}(s, a) \right] \tag{9}$$

$$\text{subject to } \mathbb{E}_{s \sim \rho_{\theta_{\mathrm{old}}}} \left[D_{\mathrm{KL}} \left(\pi_{\theta_{\mathrm{old}}}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s) \right) \right] \leq \delta \tag{10}$$

2.13 Proximal Policy Optimization (PPO)

There are two major issues with vanilla policy gradient methods. Firstly, it is difficult to optimize in the sense that it is difficult to find the right step size to use in gradient descent. The input data distribution is non-stationary - you sample trajectories using a learned policy, then you use those samples to update your policy, then you use this updated policy to sample new trajectories. However, if, at any point in time, you use a set of bad samples and therefore, your optimisation step is wrong, this could lead to performance collapse - with the bad samples, you take a "wrong step" to get a poor policy, with which you sample new trajectories which are also poor which you then use to optimise again. The second issue is that the algorithm is sample inefficient - with any particular set of sampled trajectories, we carry out one step of gradient descent and then throw those samples out. For future optimisation steps, we sample *new* trajectories. Although we have made some modifications to the basic vanilla PG algorithm like we found the actor-critic methods, they are still insufficient in completely curbing these issues.

We now derive the building blocks of Trust Region Policy Optimisation (TRPO): We already saw the performance difference lemme:

$$V_{\pi'} - V_{\pi} = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}}\left[\mathbb{E}_{a\sim\pi'}\left[A^{\pi}(s,a)\right]\right]$$

Now, suppose, our current policy is π. In our next step, we essentially want to maximize the difference between $V_{\pi'} - V_{\pi}$. Therefore,

$$\begin{aligned}
\arg\max_{\pi'} V_{\pi'} - V_{\pi} &= \arg\max_{\pi'} \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}}\left[\mathbb{E}_{a\sim\pi'}\left[A^{\pi}(s,a)\right]\right] \\
&= \arg\max_{\pi'} \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}}\left[\mathbb{E}_{a\sim\pi}\left[\frac{\pi'(a|s)}{\pi(a|s)}A^{\pi}(s,a)\right]\right] \\
&\approx \arg\max_{\pi'} \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi}}\left[\mathbb{E}_{a\sim\pi}\left[\frac{\pi'(a|s)}{\pi(a|s)}A^{\pi}(s,a)\right]\right] \\
&=: \arg\max_{\pi'} \mathcal{L}_{\pi}(\pi')
\end{aligned}$$

where, in the second last line, we made the approximation $d^{\pi'} \approx d^{\pi}$. This approximation only holds true if

$$||V_{\pi'} - V_{\pi} - \mathcal{L}_{\pi}(\pi')|| \leq C\sqrt{\mathbb{E}_{s_t\sim\pi}\left[D_{KL}(\pi(\cdot|s_t)||\pi'(\cdot|s_t))\right]}$$

With this, TRPO maximises $\mathcal{L}_{\pi}(\pi')$ subject to $\mathbb{E}_{s\sim\pi}\left[D_{KL}(\pi(\cdot|s)||\pi'(\cdot|s))\right] \leq \delta$. Note that, in actual implementation, we use a learned approximation for the advantage function.

Also note that we get monotonic improvement since the KL divergence is zero when $\pi' = \pi$ whereas $\mathcal{L}_{\pi}(\pi) = 0$ too, therefore, the performance of π' is at least as good as π.

PPO slightly modifies this - instead of placing a harsh constraint in the optimization process (which requires conjugate gradient descent otherwise), instead PPO brings in 2 variants. The first is to maximize $\mathbb{E}_{s_t\sim d^\pi, a_t\sim\pi}\left[\frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)}A^\pi(s_t,a_t) - \beta\cdot D_{KL}(\pi'(\cdot|s_t)||\pi(\cdot|s_t))\right]$. If the KL-divergence is too high, we adaptively increase β and if it is small, then we decrease β. The other variant is as follows - define $r_t(\theta) := \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$. Then, maximize

$$\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\left[\min(r_t(\theta)A^{\pi_\theta}(s_t,a_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A^{\pi_\theta}(s_t,a_t))\right]\right]$$

.

In both variants, the algorithm uses an advantage estimator $\hat{A}^\pi(s_t,a_t)$. PPO uses Generalized Advantage Estimator (GAE).

First, we define N-step advantage estimators:

$$\begin{aligned}
\hat{A}_t^{(1)} &= r_t + \gamma V(s_{t+1}) - V(s_t)\\
\hat{A}_t^{(2)} &= r_t + \gamma r_{t+1} + \gamma V(s_{t+2}) - V(s_t)\\
\hat{A}_t^{(\infty)} &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots - V(s_t).
\end{aligned}$$

If we define

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t),$$

then, these become:

$$\begin{aligned}
\hat{A}_t^{(1)} &= \delta_t^V,\\
\hat{A}_t^{(2)} &= \delta_t^V + \gamma\delta_{t+1}^V,\\
\hat{A}_t^{(k)} &= \sum_{l=0}^{k-1}\gamma^l\delta_{t+l}^V.
\end{aligned}$$

Thus, generally,

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1}\gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t).$$

GAE is an exponentially-weighted average of k-step estimators:

$$\begin{aligned}
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \dots\right)\\
&= (1-\lambda)\left(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots\right)\\
&= (1-\lambda)\left(\delta_t^V(1+\lambda+\lambda^2+\dots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\dots) + \dots\right)\\
&= (1-\lambda)\left(\delta_t^V\frac{1}{1-\lambda} + \gamma\delta_{t+1}^V\frac{\lambda}{1-\lambda} + \gamma^2\delta_{t+2}^V\frac{\lambda^2}{1-\lambda} + \dots\right)\\
&= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V.
\end{aligned}$$

PPO uses a truncated version of a GAE:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}^V$$

2.14 Deterministic Policy Gradient Methods (DPG)

Notation: We denote $r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$. Then, $V^\pi(s) = \mathbb{E}[r_1^\gamma | S_1 = s; \pi]$ and $Q^\pi(s,a) = \mathbb{E}[r_1^\gamma | S_1 = s, A_1 = a; \pi]$. The density at state s' after transitioning for t timesteps from state s is $p(s \to s', t, \pi)$. The improper, discounted state distribution is $p^\pi(s') := \int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s) p(s \to s', t, \pi) ds$.

Suppose, $\mathcal{A} = \mathbb{R}^m$ and $\mathcal{S} = \mathbb{R}^d$.

Goal: Learn a policy which maximizes $J(\pi) := \mathbb{E}[r_1^\gamma | \pi]$. With our notation, this becomes:

$$J(\pi_\theta) = \int_{\mathcal{S}} p^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) r(s,a) da ds = \mathbb{E}_{s \sim p^\pi, a \sim \pi_\theta}[r(s,a)].$$

Intuition behind the deterministic policy gradient theorem:

Most model-free RL algorithms use policy evaluation and policy improvement together. Evaluation approximates $Q^\pi(s,a)$ and then improvement updates the policy, most often through $\pi^{k+1}(s) = \arg\max_a Q^{\pi^k}(s,a)$. However, in continuous action spaces, this is difficult since the $\arg\max_a$ requires a global maximisation at each step and our action space is very large (because it is continuous). Instead, the idea is to move the policy in the direction of the gradient of Q^{π^k} (instead of maximizing it altogether). Then

$$\theta^{k+1} = \theta^k + \alpha \mathbb{E}_{s \sim p^{\theta_k}}[\nabla_\theta Q^{\pi^{\theta^k}}(s, \pi^{\theta^k}(s)].$$

By chain rule this becomes

$$\theta^{k+1} = \theta^k + \alpha \mathbb{E}_{s \sim p^{\theta_k}}[\nabla_\theta \pi^\theta(s) \nabla_a Q^{\pi^{\theta^k}}(s,a)|_{a=\pi^\theta(s)}].$$

Notation: To distinguish between stochastic and deterministic policy, we will use $\mu_\theta(s)$ as our deterministic policy.

More formally, our performance objective is

$$J(\mu_\theta) = \int_{\mathcal{S}} p^{\mu_\theta}(s) r(s, \mu_\theta(s)) ds = \mathbb{E}_{s \sim p^{\mu_\theta}}[r(s, \mu_\theta(s))]$$

.

Then,

$$\nabla_\theta J(\mu_\theta) = \int_{\mathcal{S}} p^{\mu_\theta}(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)} ds = \mathbb{E}_{s\sim p^{\mu_\theta}} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)} \right] \tag{11}$$

On-policy algorithm: We have a critic that estimates the action-value function while the actor updates the policy by ascending the gradient of the action-value function (using equation 11). The critic, $Q^w(s,a)$ approximates $Q^\mu(s,a)$. The update rules are :

$$\begin{aligned} \delta_t &= r_t + \gamma Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t) \\ w_{t+1} &= w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t) \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu_\theta(s)} \end{aligned}$$

Off-policy algorithm: Suppose we have trajectories generated by behavior policy $\pi(s,a)$. The new objective becomes:

$$J_\pi(\mu_\theta) = \int_{\mathcal{S}} p^\pi(s) V^\mu(s) ds = \int_{\mathcal{S}} p^\pi(s) Q^\mu(s, \mu_\theta(s)) ds$$

and the update rule becomes

$$\nabla_\theta J_\pi(\mu(\theta)) \approx \int_{\mathcal{S}} p^\pi(s) \nabla_\theta \mu_\theta(a|s) Q^\mu(s,a) ds = \mathbb{E}_{s\sim p^\pi} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} \right].$$

Now we can develop an actor-critic algorithm similar to the on-policy case:our critic is $Q^w(s,a)$:

$$\begin{aligned} \delta_t &= r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t) \\ w_{t+1} &= w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t) \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu_\theta(s)} \end{aligned}$$

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[2] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2017. [Online]. Available: https://arxiv.org/abs/1502.05477

[3] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomput.*, vol. 71, no. 7–9, p. 1180–1190, Mar. 2008. [Online]. Available: https://doi.org/10.1016/j.neucom.2007.11.026

[4] D. Silver, G. Lever, N. M. O. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:13928442