Policy Gradient Notes

1 Prelude

1.1

Markov chain: $\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$
$\mathcal{S}$: State space
$\mathcal{T}$: Transition operator

MDP: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$
let $\mu_{t,j} = p(s_t = j)$
let $= p(a_t = k)$
let $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$
then we have:

$$\mu_{t+1,i} = \sum_{j,k} \mathcal{T}_{i,j,k} \mu_{t,j} \xi_{t,k} \tag{1}$$

reward function: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

POMDP: $\mathcal{M} = \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \varepsilon, r$
ε: emission probability $p(o_t|s_t)$

$$p_\theta(\tau) = p_\theta(s_1, a_1, ..., s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{2}$$

Objective for reinforcement learning:

$$\begin{aligned} \theta^* &= \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right] \\ &= \arg\max_\theta \sum_{t=1}^{T} E_{(s_t,a_t) \sim p_\theta(s_t,a_t)} [r(s_t, a_t)] \end{aligned}$$

we can rewrite as:

$$p((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1|s_t,a_t}) \pi_\theta(a_{t+1}|s_{t+1}) \tag{3}$$

when T is infinite, the distribution is stationary: $\mu = \mathcal{T}\mu$, on the other hand:

$$(\mathcal{T} - \mathbf{I})\mu = 0 \tag{4}$$

where $\mu = p_\theta(s, a)$ which is the stationary distribution, and μ is the eigen vector of $\mathcal{T}$ with eigenvalue 1.

2 Algorithms

There are generally 3 parts:

- fit a model:
$$J(\theta) = E_\pi \left[\sum_t r_t\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_t r_t^i \tag{5}$$
or learn f_ϕ such that $s_{t+1} \approx f_\phi(s_t, a_t)$
- improve the policy:
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{6}$$
or backprop through f_ϕ and r, train $\pi_\theta(s_t) = a_t$
- generate samples (run the policy)

2.1 Value Function

$$E_{s_1\sim p(s_1)}[E_{a_1\sim\pi(a_1|s_1)}[r(s_1,a_1) + E_{s_2\sim\pi(s_2|s_1,a_1)}[E_{a_2|\pi(a_2|s_2)}[r(s_2,a_2)+...|s_2]|s_1,a_1]|s_1]] \tag{7}$$

Define:
$$Q(s_1,a_1) = r(s_1,a_1) + E_{s_2\sim\pi(s_2|s_1,a_1)}[E_{a_2|\pi(a_2|s_2)}[r(s_2,a_2)+...|s_2]|s_1,a_1] \tag{8}$$

Then we can rewrite the original RL objective as:

$$E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right] = E_{s_1\sim p(s_1)}[E_{a_1\sim\pi(a_1|s_1)}[Q(s_1,a_1)|s_1]] \tag{9}$$

in which it is easy to modify $\pi_\theta(a_1|s_1)$ if $Q(s_1,a_1)$ is known In general: we define:

$$Q^\pi(s_t,a_t) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'},a_{t'})|s_t,a_t] \tag{10}$$

which is the total reward from taking a_t in s_t. We can also define value function:

$$V^\pi(s_t) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'},a_{t'})|s_t] \tag{11}$$

which is the total reward from s_t, and can also be written as:

$$V^\pi(s_t) = E_{a_t\sim\pi(a_t|s_t)}[Q^\pi(s_t,a_t)] \tag{12}$$

The RL objective then is: $E_{s_1\sim p(s_1)}[V^\pi(s_1)]$

2.2 Using Q-functions and value functions

- If we have policy π, and we know $Q^\pi(s,a)$, then we can improve π:
- compute gradient to increase probability of good actions a: if $Q^\pi(s,a) > V^\pi(s)$, then a is better than average

2.3 Sample efficiency

Off policy: able to improve the policy without generating new samples from that policy
On policy: each time the policy is changed, even a little bit, we need to generate new samples

2.4 Comparison: stability and ease of use

Supervised learning: almost always gradient descent
RL: often not gradient descent

- Q learning: fixed point iteration
- Model-based RL: model is not optimized for expected reward
- Policy gradient: is gradient descent, but also often the least efficient

2.5 Assumptions

- full observability
- episodic learnign
- continuity or smoothness

3 Policy Gradients

We can rewrite the RL objective as:

$$\begin{aligned}
\theta^* &= \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right] \\
&= \arg\max_\theta J(\theta) \\
&\approx \arg\max_\theta \frac{1}{N}\sum_i\sum_t r(s_{i,t}, a_{i,t})
\end{aligned}$$

We also write $J(\theta) = E_{\tau\sim p_\theta(\tau)}[r(\tau)] = \int p_\theta(\tau)r(\tau)d\tau$. We then have:

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_\theta(\tau)r(\tau)d\tau \tag{13}$$

There is a convenient identity to simplify this: $p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) = p_\theta(\tau)\frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)}$. The gradient then becomes:

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

We further simplify the expression by taking the log on both sides:

$$\log p_\theta(s_1, a_1, ..., s_T, a_T) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t) \tag{14}$$

and if we differentiate the right side of the equation, we would have:

$$\nabla \log p_\theta(s_1, a_1, ..., s_T, a_T) = \sum_{t=1}^{T} \nabla \log \pi_\theta(a_t|s_t) \tag{15}$$

and if we plug that back we would have:

$$\nabla_\theta J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right] \tag{16}$$

We can then evaluate the policy gradient as:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{t=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right) \tag{17}$$

We can then improve the policy by taking the gradient descent:

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta) \tag{18}$$

3.1 REINFORCE algorithm

- sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$ (run the policy)
- evaluate policy gradient: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{t=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right)$
- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

3.2 Gaussian Policies

For continuous actions, we can then use $\pi_\theta(a_t|s_t) = \mathcal{N}(f_{\text{neural network}}(s_t), \Sigma)$, and then:

$$\log \pi_\theta(a_t|s_t) = -\frac{1}{2}||f(s_t) - a_t||_\Sigma^2 + \text{const} \tag{19}$$

We can then take the derivative:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = -\frac{1}{2}\Sigma^{-1}(f(s_t) - a_t)\frac{df}{d\theta} \tag{20}$$

The policy gradient is weighted by the reward and makes good stuff more likely while making bad stuff less likely.

3.2.1 Partial observability

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{t=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right) \tag{21}$$

In which MDP is not used, and this can be used in POMDP without modification

3.3 Reducing Variance

3.3.1 Causality

policy at time t' cannot affect reward at time t when $t < t'$. As a result, we can write the above equation as:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{t=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t})\left(\sum_{t'=1}^{T} r(s_{i,t'},a_{i,t'})\right) \tag{22}$$

and further:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{t=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t})\left(\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})\right) \tag{23}$$

total sum is smaller, thus smaller variances:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{t=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t})\hat{Q}_{i,t} \tag{24}$$

3.3.2 Baseline

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log p_\theta(\tau)[r(\tau) - b] \tag{25}$$

where $b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$, and it won't change the expected value of the gradient:

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

subtracting a baseline is unbiased in expectation, but will reduce variance. Average reward is not the optimal baseline, but it works pretty well.

3.3.3 Variance

$\text{Var}[x] = E[x^2] - E[x]^2$, and we have:

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)] \tag{26}$$

and

$$\text{Var} = E_{\tau \sim p_\theta(\tau)}[(\nabla_\theta \log p_\theta(\tau)(r(\tau) - b))^2] - E_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]^2 \tag{27}$$

where the second term is unbiased in expectation. We declare $g(\tau) = \nabla_\theta \log p_\theta(\tau)$

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

in which the first term does not depend on b, while the second and third do. We then get:

$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]} \tag{28}$$

3.3.4 Off-policy learning

Importance sampling:

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

What if we don't have samples from $p_\theta(\tau)$, but from $\bar{p}_(\tau)$ instead?

$$J(\theta) = E_{x \sim q(x)}[\frac{p_\theta(\tau)}{\bar{p}(\tau)} r(\tau)] \tag{29}$$

Expanding the terms, we have:

$$\frac{p_\theta(\tau)}{\bar{p}_\tau} = \frac{p(s_1) \prod_{t=1}^T \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)}{p(s_1) \prod_{t=1}^T \bar{\pi}(a_t|s_t) p(s_{t+1}|s_t, a_t)} = \frac{\prod_{t=1}^T \pi_\theta(a_t|s_t)}{\prod_{t=1}^T \bar{\pi}(a_t|s_t)} \tag{30}$$

We can estimate the value of some new parameters θ':

$$J(\theta') = E_{\tau \sim p_\theta(\tau)} \left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau) \right] \tag{31}$$

differentiate that we have:

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

If we estimate locally, at $\theta = \theta'$, then we would have the original policy gradient equation.

4 Off-policy policy gradient

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

if we ignore the last term, we get a policy iteration algorithm. The second term can get diminished quickly if the importance weights are less than 1. We can write the objective for off-policy gradient descent a bit differently:

$$
\begin{aligned}
\nabla_{\theta'} J(\theta') &\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(s_{i,t}, a_{i,t})}{\pi_\theta(s_{i,t}, a_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t}|s_(i,t)) \hat{Q}_{i,t} \\
&= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(s_{i,t}, a_{i,t})}{\pi_\theta(s_{i,t}, a_{i,t})} \frac{\pi_{\theta'}(s_{i,t}, a_{i,t})}{\pi_\theta(s_{i,t}, a_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t}|s_(i,t)) \hat{Q}_{i,t}
\end{aligned}
$$

5 Implementing policy gradients

$$
\nabla_\theta \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \tag{33}
$$

The calculation inside the summation is inefficient, we need a graph such that its gradient is the policy gradient.
Maximum likelihood: $\nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{T} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})$ which is the gradient of $J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{T} \sum_{t=1}^{T} \log \pi_\theta(a_{i,t}|s_{i,t})$ We can implement "pseudo-loss" as a weighted maximum likelihood:

$$
\tilde{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{T} \sum_{t=1^N} \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \tag{34}
$$

log term here will have cross-entropy loss for discrete actions or squared error is the outut is Gaussian.

6 Advanced policy gradients

first-order gradient descent:

$$
\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \tag{35}
$$

s.t. $||\theta' - \theta||^2 \leq \epsilon$ and α can be viewed as Lagrange multiplier for the gradient descent. We can rescale the gradient by constraining in the policy space instead of the parameter space:

$$
\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \tag{36}
$$

s.t. $D(\pi_{\theta'}, \pi_\theta) \leq \epsilon$ which is a divergence measure. Usually KL-divergence: $D_{\text{KL}}(\pi_{\theta'}||\pi_\theta) = E_{\pi_{\theta'}}[\log \pi_\theta - \log \pi_{\theta'}]$, which can be approximated as:

$$
D_{\text{KL}}(\pi_{\theta'}||\pi_\theta) \approx (\theta' - \theta)^T \mathbf{F} (\theta' - \theta) \tag{37}
$$

where $\mathbf{F}$ is the Fisher-information matrix: $E_{\pi_\theta}[\nabla_{\theta \log_{\pi_\theta}}(a|s) \nabla_{\theta \log_{\pi_\theta}}(a|s)^T]$ We can then formalize the problem as covariant/natural policy gradient:

$$
\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \tag{38}
$$

s.t. $||\theta' - \theta||_F^2 \leq \epsilon$

We would then have:

$$\theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_\theta J(\theta) \tag{39}$$

can solve optimal α while solving $\mathbf{F}^{-1}\nabla_\theta J(\theta)$. conjugate gradient works well on that. Natural gradient picks α while TRPO picks ϵ.