Lecture #6: Advanced Policy Gradients

Chong Yihui, Shen Ting, Dongyang
Department of Computer Science
National University of Singapore
Singapore, S117417
{ychong, STUDENT2, etc.}@u.nus.edu

1 Policy gradient as policy iteration

First, we try to show that we can use the advantage of the previous policy to get a better policy and the result method looks like policy gradient.

To do so, we try to maximise the difference in the returns: $J(\theta') - J(\theta)$

We claim the following equivalence:

$$\text{claim:} J(\theta') - J(\theta) = E_{\underbrace{\tau\sim p_{\theta'}(\tau)}_{\text{New policy}}}\left[\sum_t \gamma^t \underbrace{A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)}_{\text{Advantage under old policy}}\right]$$

Recall that $J(\theta) = E_{s_0 \sim p(s_1)}[V^{\pi_\theta}(s_0)]$

The following derivation tricks are used:

1. Initial state distribution is the same for all policy.

$$\begin{aligned} J(\theta') - J(\theta) &= J(\theta') - E_{\mathbf{s}_0\sim p(\mathbf{s}_0)}\left[V^{\pi_\theta}(\mathbf{s}_0)\right] \\ &= J(\theta') - E_{\tau\sim p_{\theta'}(\tau)}\left[V^{\pi_\theta}(\mathbf{s}_0)\right] \end{aligned}$$

We can do this since the distribution of trajectories is the same for the initial state

2. Expanding out $V^{\pi_\theta}(s_0)$ to a telescoping sum

$V^{\pi_\theta}(s_0) = \sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(s_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(s_t)$

$= V^{\pi_\theta}(s_0) + \gamma^1 V^{\pi_\theta}(s_1) + \gamma^2 V^{\pi_\theta}(s_2) ... - [\gamma^1 V^{\pi_\theta}(s_1) + \gamma^2 V^{\pi_\theta}(s_2) + ...]$

3. Correction: t=0 instead of t=1 for $J(\theta)$

$J(\theta') = E_{\tau\sim p_{\theta'}(\tau)}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$

instead of

$E_{\tau\sim p_{\theta'}(\tau)}[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)]]$

After using the above trick, we would have proofed our claim. Next, we instead of taking expectation over our new policy(which we do not have yet and is what we are trying to find), we would like to take the expectation over the old policy.

Exploration in Computer Science Research: Deep Reinforcement Learning (CS 6101, 2019), National University of Singapore.

2 Ignoring distribution mismatch

$E_{\tau \sim p_{\theta'}(\tau)}[\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)] = \sum_t E s_t \sim p_{\theta'}(s_t)[E a_t \sim \pi_{\theta'}(a_t|s_t)[\gamma^t A^{\pi_\theta}(s_t, a_t)]]$

Using importance sampling we are able to switch the inside expectation to get the following equation:
$= \sum_t E s_t \sim p_{\theta'}(s_t)[E a_t \sim \pi_\theta(a_t|s_t)[\frac{\pi_{\theta'}(a_t,s_t)}{\pi_\theta(a_t,s_t)}\gamma^t A^{\pi_\theta}(s_t, a_t)]]$

However, we are not able to do this for the outer expectation as $\frac{\pi_{\theta'}(a_t,s_t)}{\pi_\theta(a_t,s_t)} < 1$ and the probability of state is a series of multiplication. Therefore, the importance sampling weight decays to 0 as the time horizon gets longer.

Main Takeaway: we can use our existing policy to approx our cost if they are similar (ignore the distribution mismatch) Next, under what conditions are they similar?

3 Bounding the distribution change

Important notation: $|p_{\theta'}(s_t) - p_\theta(s_t)| = \sum_x(|p(x) - q(x)|)$

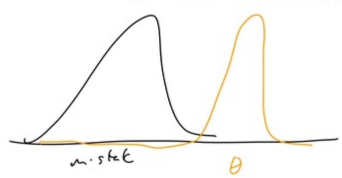
Also note that $|p_{\theta'}(s_t) - p_\theta(s_t)| <= 2$.

A possible worst case: Consider 2 different points x_1, x_2 such that $p(x_1) = 1$ and $q(x_1) = 0$

$p(x_2) = 1$ and $q(x_2) = 0$

$|p_{\theta'}(s_t) - p_\theta(s_t)| = |1 - 0| + |0 - 1| = 2$

no overlap, so difference = 2.



When can we change $p_{\theta'}$ to p_θ?

[Case 1: Assume policy is deterministic] Claim: $p_\theta(s_t)$ is close to $p'_\theta(s_t)$ when π_θ is close to π'_θ

Definition of close for deterministic distribution:

$$\pi_{\theta'} \text{ is } \textit{close} \text{ to } \pi_\theta \text{ if } \pi_{\theta'}(\mathbf{a}_t \neq \pi_\theta(\mathbf{s}_t)|\mathbf{s}_t) \leq \epsilon$$

If the above bound holds,

$|p_{\theta'}(s_t) - p_\theta(s_t)| \leq 2\epsilon t$

[Case 2: Assume policy is stochastic]

Definition of close for general case:

$\pi_{\theta'}$ is close to π_θ if $|\pi_{\theta'}(at|s_t) - \pi_\theta(at|s_t)| \leq \epsilon$ for all s_t

If the above bound holds and using the useful lemma,

$|p_{\theta'}(s_t) - p_\theta(s_t)| \leq 2\epsilon t$

4 Bounding with KL-divergence

Instead of using ϵ in the above bound, we can also bound the policy using KL-divergence

$|\pi_{\theta'}(at|s_t) - \pi_\theta(at|s_t)| \leq \sqrt{\frac{1}{2}D_{KL}(\pi_{\theta'(a_t|s_t)}||\pi_\theta(a_t|s_t))}$

With this, we get the following objective function

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

$$\text{such that } D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) \leq \epsilon$$

To optimize the above objective,

[Method 1] Enforce the constraint on the KL divergence use Lagrangian method.

$$\mathcal{L}(\theta', \lambda) = \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] - \lambda(D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) - \epsilon)$$

1. Maximize $\mathcal{L}(\theta', \lambda)$ with respect to θ' ← **can do this incompletely (for a few grad steps)**
2. $\lambda \leftarrow \lambda + \alpha(D_{\mathrm{KL}}(\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)) - \epsilon)$

Intuition: raise λ if constraint violated too much, else lower it

an instance of *dual gradient descent* (more on this later!)

This is an example of dual gradient descent (on both θ and λ)

[Method 2] Use natural policy gradient (for further reading).

[Method 3] Use natural policy gradient with learning rate (Trust region policy optimization)

$$\alpha = \sqrt{\frac{2\epsilon}{\nabla_\theta J(\theta)^T \mathbf{F} \nabla_\theta J(\theta)}}$$

5 Further Readings

Going towards the trust region policy gradient algorithm:

`https://medium.com/@jonathan_hui/rl-natural-policy-gradient-actor-critic-using-kronecker-factore`

Trust Region Policy Optimization (TRPO) using Adam

`https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-part-2-f51e3b2e373a`

Natural Gradient and Fisher Information:

`https://wiseodd.github.io/techblog/2018/03/11/fisher-information/`

`https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/`

Natural Gradient Intuition

`http://kvfrans.com/a-intuitive-explanation-of-natural-gradient-descent/`

Intuitive explanation of Policy Gradients

`http://karpathy.github.io/2016/05/31/rl/`