Advanced Policy Gradients

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

Novemer 04, 2023

Overview of this Lecture

TCS Research & Innovation

Introduction

Policy Gradients : The Story So Far

Policy gradient algorithms try to solve the optimization problem

$$\max_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right]$$

by taking stochastic gradient ascent on the policy parameters θ, using the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \Psi_t \right]$$

1. $\Psi_t = \gamma^t Q^{\pi_\theta}(s_t, a_t)$, State action value function
2. $\Psi_t = \gamma^t A^{\pi_\theta}(s_t, a_t) = \gamma^t \left[Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) \right]$, Advantage function
3. $\Psi_t = \gamma^t \left[r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t) \right]$, TD residual

Limitation 1 : Sample Inefficiency

Gradient of the performance measure is given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\Psi_t\right]$$

- Policy Gradient seen thus far is **on-policy**
- Gradient update is performed using samples collected from current policy
- Above formulation does not allow to recycle old data
- If we want to use samples from other policies, then the above gradient term needs correction using **importance sampling** weights

Importance Sampling in PG : Possible Remedy ?

What if we do not samples from π_θ instead samples are from π_η(i.e. $\tau \sim \pi_\eta$) ?

We can rewrite the gradient estimate as,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\eta} \left[\frac{P(\tau|\pi_\theta)}{P(\tau|\pi_\eta)} \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\ \Psi_t \right]$$

$$\frac{P(\tau|\pi_\theta)}{P(\tau|\pi_\eta)} = \frac{\mu(s_0) \prod_{t=0}^{\infty} P(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)}{\mu(s_0) \prod_{t=0}^{\infty} P(s_{t+1}|s_t, a_t) \pi_\eta(a_t|s_t)} = \prod_{t=0}^{\infty} \frac{\pi_\theta(a_t|s_t)}{\pi_\eta(a_t|s_t)}$$

Even for policies only slightly different from each other, many small differences multiply to become a big difference and IS weights can **explode** or **vanish**

Problem : How can we efficiently use samples from old policies while avoiding the challenges posed by importance sampling ?

Limitation 2 : Updates in Parameter Space

- Policy gradient takes step in parameter space

$$\theta_{n+1} = \theta_n + \alpha \nabla_{\theta_n} J(\theta_n)$$

- **Distance** in parameter space $\neq$ Distance in policy space
- Hard to get step size right as a result
- Example of a policy space : For finite state and action case, the policy space is given by

$$\Pi = \left\{ \pi : \pi \in \mathbb{R}^{|S| \times |A|}, \sum_a \pi_{sa} = 1, 0 \leq \pi_{sa} \leq 1 \right\}$$

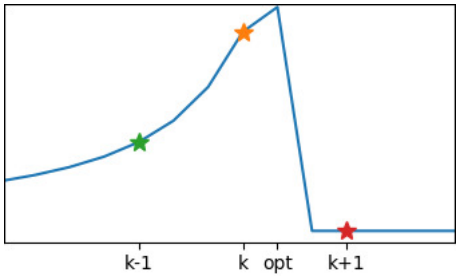
The Step Size Issue

Policy Gradient algorithms perform stochastic gradient ascent

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$$

with step $\Delta_k = \alpha_k \hat{g}_k$

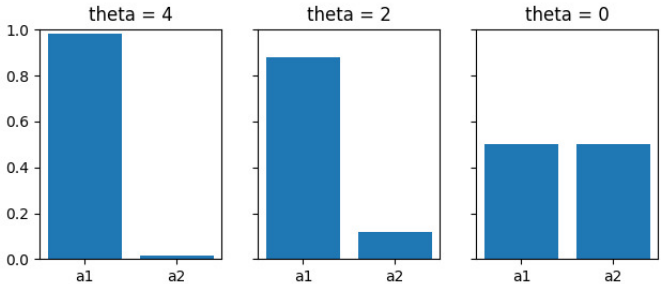
- Step size is too large, performance collapse
  - ★ Consequences in RL setting is more severe than for supervised learning
- Step size is too low, slow progress



Difference in Policy Space : Discrete Case

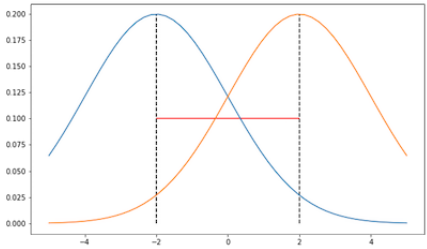
Consider a family of policies with the following parametrization

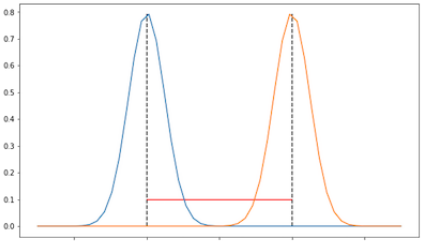
$$\pi_\theta(a) = \begin{cases} \sigma(\theta), & \text{if } a = 1 \\ 1 - \sigma(\theta), & \text{if } a = 2 \end{cases}$$



Small changes in the policy parameters can unexpectedly lead to big changes in the policy

Difference in Policy Space : Continuous Case





Desiderata

- How to make use of data from old policy while avoiding challenges that arise from importance sampling ?
  - ★ At least use roll-outs from **most recent policy** as effectively as possible
- How to design an update rule that doesn't change the policy more than we intend to ?
  - ★ Take steps that respect notion of **distance in policy space** rather than in parameter space

Policy Performance Bounds

Relative Performance of Two Policies

Recall that the performance of a policy π is given by

$$J(\pi) = V^{\pi}(s_0)$$

where $s_0 \in \mu(s)$ is the start state of the trajectory π

Relative Performance Identity of Two Policies

For any two policies π' and π_0 we have,

$$J(\pi') - J(\pi_0) = \mathop{\mathbb{E}}_{\tau \sim \pi'} \left[\gamma^t A^{\pi_0}(s_t, a_t) \right]$$

Proof of Relative Performance Identity of Two Policies

$$
\begin{aligned}
J(\pi') - J(\pi_0)) &\stackrel{?}{=} \mathop{\mathbb{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi_0}(s_t,a_t)\right] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t\left[r_{t+1}+\gamma V^{\pi_0}(s_{t+1})-V^{\pi_0}(s_t)\right]\right] \\
&= J(\pi') + \mathop{\mathbb{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^{t+1}V^{\pi_0}(s_{t+1}) - \sum_{t=0}^{\infty}\gamma^t V^{\pi_0}(s_t)\right] \\
&= J(\pi') + \mathop{\mathbb{E}}_{\tau\sim\pi'}\left[\sum_{t=1}^{\infty}\gamma^t V^{\pi_0}(s_t) - \sum_{t=0}^{\infty}\gamma^t V^{\pi_0}(s_t)\right] \\
&= J(\pi') - \mathop{\mathbb{E}}_{\tau\sim\pi'}\left[V^{\pi_0}(s_0)\right] \\
&= J(\pi') - J(\pi_0)
\end{aligned}
$$

Policy Improvement Idea

Can we use the identity,

$$J(\pi') - J(\pi_0) = \mathop{\mathbb{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_0}(s_t, a_t) \right]$$

for policy improvement to go from **old policy** π_0 to **new policy** π' ?

$$\arg\max_{\pi'} J(\pi') = \arg\max_{\pi'} \left[J(\pi') - J(\pi_0) \right] = \arg\max_{\pi'} \left[\mathop{\mathbb{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_0}(s_t, a_t) \right] \right]$$

- Define performance of π' in terms of advantages from π_0
- Problem : Still requires trajectories from π' !!

Discounted State Distribution

Define discounted future state distribution d^π for a state s as

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}(s_t = s|\pi)$$

Consider the expectation

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

The above expectation can be rewritten using the discounted future state distribution d^π as

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

That is,

$$\mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t,a_t)\right] = \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{\substack{s\sim d^\pi\\ a\sim\pi}}\left[A^\pi(s,a)\right]$$

Performance Identity and Discounted State Distribution

$$
\begin{aligned}
J(\pi') - J(\pi_0) &= \mathop{\mathbb{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_0}(s_t, a_t) \right] \\
&= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'}} \left[A^{\pi_0}(s, a) \right] \\
&= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi_0}} \left[\frac{\pi'(a|s)}{\pi_0(a|s)} A^{\pi_0}(s, a) \right]
\end{aligned}
$$

Only problem is $s \sim d^{\pi'}$!

Approximation of Discounted State Distribution

Under what conditions is it OK to assume

$$\frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi_0}} \left[\frac{\pi'(a|s)}{\pi_0(a|s)} A^{\pi_0}(s,a) \right] \stackrel{?}{\approx} \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi_0} \\ a\sim\pi_0}} \left[\frac{\pi'(a|s)}{\pi_0(a|s)} A^{\pi_0}(s,a) \right]$$

In other words, under what conditions can we have $d^{\pi'} \approx d^{\pi_0}$?

We can have $d^{\pi'} \approx d^{\pi_0}$, if π' and π_0, are close !!

- Closeness is measured in terms of **KL divergence**
- If the policies are close in **KL divergence**, then the above approximation is good

KL Divergence Between Policies

For any two probability distributions P and Q $D_{KL}(P||Q)$ is defined as

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

KL divergence has the following properties

- $D_{KL}(P||P) = 0$
- $D_{KL}(P||Q) \geq 0$
- $D_{KL}(P||Q) \neq D_{KL}(Q||P)$

KL divergence between policies π' and π_0 is given by

$$D_{KL}(\pi'||\pi_0)[s] = \sum_{a \in \mathcal{A}} \pi'(a|s) \log \frac{\pi'(a|s)}{\pi_0(a|s)}$$

Surrogate Loss Function

If policies π' and π_0 are **'close'** in terms of their KL divergence, then,

$$\begin{aligned} J(\pi') - J(\pi_0) &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi_0}} \left[\frac{\pi'(a|s)}{\pi_0(a|s)} A^{\pi_0}(s,a) \right] \\ &\approx \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi_0} \\ a\sim\pi_0}} \left[\frac{\pi'(a|s)}{\pi_0(a|s)} A^{\pi_0}(s,a) \right] \\ &= \mathop{\mathbb{E}}_{\tau\sim\pi_0} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi_0(a_t|s_t)} A^{\pi_0}(s_t,a_t) \right] \end{aligned}$$

Define the quantity $\mathcal{L}_{\pi_0}(\pi')$ as the **surrogate loss function**, as,

$$\mathcal{L}_{\pi_0}(\pi') = \mathop{\mathbb{E}}_{\tau\sim\pi_0} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi_0(a_t|s_t)} A^{\pi_0}(s_t,a_t) \right]$$

- $J(\pi') - J(\pi_0) \approx \mathcal{L}_{\pi_0}(\pi')$

Validity of Approximation

If policies π' and π_0 are 'close' in terms of their KL divergence, then,

$$J(\pi') - J(\pi_0) \approx \mathcal{L}_{\pi_0}(\pi')$$

We can have a **relative policy performance bound** using KL divergence as

$$\Big[J(\pi') - \big(J(\pi_0) + \mathcal{L}_{\pi_0}(\pi')\big)\Big] \leq C\sqrt{\mathop{\mathbb{E}}_{s\sim d^{\pi_0}}\left[D_{KL}(\pi'||\pi_0)[s]\right]}$$

where $C = \frac{4\varepsilon\gamma}{1-\gamma^2}\alpha^2$ with $\alpha = \max_{s\sim d^{\pi_0}}\left[D_{KL}(\pi'||\pi_0)[s]\right]$ and $\varepsilon = \max_{s,a} A^{\pi_0}(s,a)$

Surrogate Loss Function : Properties

$$\mathcal{L}_{\pi_0}(\pi') = \mathop{\mathbb{E}}_{\tau\sim\pi_0}\left[\sum_{t=0}^{\infty}\gamma^t\frac{\pi'(a_t|s_t)}{\pi_0(a_t|s_t)}A^{\pi_0}(s_t,a_t)\right]$$

- $\mathcal{L}_{\pi_0}(\pi')$ is something that we can evaluate using samples from old policy π_0
- Importance sampling is used; but weights depends only on current time-step (not preceding history); hence importance sampling weights don't vanish or explode
- Let π_{θ_k} and π_θ be two parametrized policies. Then, $\nabla_\theta\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)|_{\theta=\theta_k}$, is equal to the policy gradient

$$\begin{aligned}\nabla_\theta\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)|_{\theta=\theta_k} &= \mathop{\mathbb{E}}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{\infty}\frac{\nabla_\theta\pi_\theta(a_t|s_t)|_{\theta=\theta_k}}{\pi_{\theta_k}(a_t|s_t)}\ \gamma^t A^{\pi_{\theta_k}}(s_t,a_t)\right]\\ &= \mathop{\mathbb{E}}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{\infty}\nabla_\theta\log(\pi_{\theta_k}(a_t|s_t)|_{\theta=\theta_k}\ \gamma^t A^{\pi_{\theta_k}}(s_t,a_t)\right]\end{aligned}$$

Policy Improvement Guarantee

We can rewrite the relative policy performance bound equation as

$$\big[J(\pi') - J(\pi_0)\big] \geq \mathcal{L}_{\pi_0}(\pi') - C\sqrt{\mathop{\mathbb{E}}_{s\sim d^{\pi_0}}\big[D_{KL}(\pi'||\pi_0)[s]\big]}$$

Suppose π_{k+1} and π_k are related by

$$\pi_{k+1} = \arg\max_{\pi'}\left[\mathcal{L}_{\pi_k}(\pi') - C\sqrt{\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\big[D_{KL}(\pi'||\pi_k)[s]\big]}\right]$$

- Note π_k is a feasible point and the objective at π_k is equal to 0
  - ★ $\mathcal{L}_{\pi_k}(\pi_k) \propto \mathbb{E}[A^{\pi_k}(s_t, a_t)] = 0$
  - ★ $D_{KL}(\pi_k||\pi_k) = 0$
- $\implies$ Optimal value ≥ 0
- By the performance bound inequality, we have $J(\pi_{k+1}) - J(\pi_k) \geq 0$

A First-Cut Algorithm

1: Initialize π_0
2: **for** $k = 0, 1, 2, \cdots$ until convergence **do**
3: Sample a trajectory τ from policy π_k
4: Compute advantage function $A^{\pi_{\theta_k}}(a_t, s_t)$ for all (s_t, a_t) pairs in the trajectory τ
5: Solve the optimization problem

$$\pi_{k+1} = \arg\max_{\pi'} L_{\pi_k}(\pi') - C\sqrt{\mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[D_{KL}(\pi'||\pi_k)[s]\right]}$$

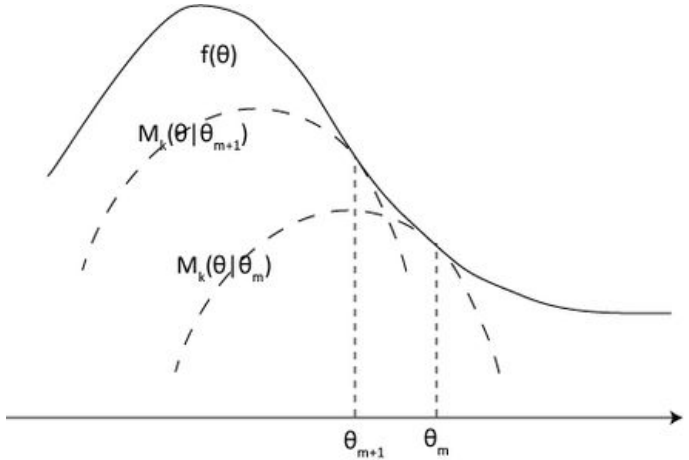
6: **end for**

Issues are :

- C is quite high when γ is close to 1 $\left(C = \frac{4\varepsilon\gamma}{1-\gamma^2}\alpha^2\right)$
- Consequently, step size becomes too small

Majorize Maximize Framework

Majorize-Maximize framework is used to solve the optimization step



Approximate Monotone Improvement

- Instead of KL penalty, use KL constraint
- Can control worst case error through constraint upper limit

$$\pi_{k+1} = \arg\max_{\pi'} \left[L_{\pi_k}(\pi') \right]$$
$$\text{such that} \quad \mathop{\mathbb{E}}_{s \sim d^{\pi_k}} D_{KL}(\pi' || \pi_k)[s] \leq \delta$$

- From the constraint, **steps respect a notion of distance in policy space**
- Above constrained optimization is basis of many algorithms, Natural Policy Gradient (NPG), truncated NPG, TRPO and PPO
- The objective and the constraint can be estimated from the roll-out of old policies – **sample efficient**
- Update is parametrization invariant

Natural Policy Gradient

Trust Region Formulation

We have the following optimization problem

$$\pi_{k+1} = \arg\max_{\pi'} \left[\mathcal{L}_{\pi_k}(\pi')\right]$$
$$\text{such that } \bar{D}_{KL}(\pi' || \pi_k) \leq \delta$$

The constraint on the optimization problem is the trust region with size δ and some guarantees on performance improvement are there within the trust region

For parametrized policies the optimization can be written as

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \left[\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)\right]$$
$$\text{such that } \bar{D}_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$

How do we solve it ?

- Linear approximation for the objective
- Quadratic approximation for the constraint

Approximation of Objective Function

Taylor series expansion for function $f(x)$ around point a is given by

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \cdots$$

- Using Taylor series expansion on objective function $\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)$ around θ_k (upto first order term) gives us

$$\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) \approx \cancelto{0}{\mathcal{L}_{\pi_{\theta_k}}(\pi_{\theta_k})} + g^T(\theta - \theta_k) \qquad \text{where } g \doteq \nabla_\theta \mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)\mid_{\theta=\theta_k}$$

- Recall that g is exactly the policy gradient (from previous lecture !)

$$\nabla_\theta \mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)|_{\theta=\theta_k} = \mathop{\mathbb{E}}_{\tau\sim\pi_{\theta_k}}\left[\sum_{t=0}^{\infty} \nabla_\theta \log(\pi_{\theta_k}(a_t|s_t)|_{\theta=\theta_k}\ \gamma^t A^{\pi_{\theta_k}}(s_t, a_t)\right]$$

- Objective function is simplified to

$$\theta_{k+1} = \arg\max_\theta g^T(\theta - \theta_k)$$

Approximation of Trust Region Constraint

Using Taylor series expansion on the constraint (around θ_k; upto second order) gives us

$$\bar{D}_{KL}(\pi_\theta||\pi_{\theta_k}) \approx \cancelto{0}{\bar{D}_{KL}(\pi_{\theta_k}||\pi_{\theta_k})} + \cancelto{0}{\nabla_\theta \bar{D}_{KL}(\pi_\theta||\pi_{\theta_k})|_{\theta=\theta_k}} + \nabla^2_\theta \bar{D}_{KL}(\pi_\theta||\pi_{\theta_k}) \ |_{\theta=\theta_k}$$

The first order term $\nabla_\theta \bar{D}_{KL}(\pi_\theta||\pi_{\theta_k})$ evaluates to zero since the expectation of the score function is zero

$$\nabla_\theta \bar{D}_{KL}(\pi_\theta \,\|\, \pi_{\theta_k}) = \nabla_\theta \mathop{\mathbb{E}}_{\pi_\theta}[\log \pi_\theta] - \nabla_\theta \mathop{\mathbb{E}}_{\pi_\theta}[\log \pi_{\theta_k}] = \mathop{\mathbb{E}}_{\pi_\theta}[\nabla_\theta \log \pi_\theta] = 0$$

Therefore, we are left only with the second order term

$$\bar{D}_{KL}(\pi_\theta||\pi_{\theta_k}) \approx \frac{1}{2}(\theta - \theta_k)^T H \ (\theta - \theta_k) \quad \text{where } H \doteq \nabla^2_\theta \bar{D}_{KL}(\pi_\theta||\pi_{\theta_k}) \ |_{\theta=\theta_k}$$

Natural Policy Gradient

The optimization problem is now simplified as

$$\theta_{k+1} = \arg\max_{\theta} g^T(\theta - \theta_k)$$

$$\text{such that } \frac{1}{2}(\theta - \theta_k)^T H\ (\theta - \theta_k) \leq \delta$$

Linear objective with quadratic constraint

Solution to the approximate problem obtained using Lagrange multiplier method

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

The term $H^{-1}g$ is called the Natural gradient

Algorithm : Natural Policy Gradient

Algorithm Natural Policy Gradient

1: Initialize π_0
2: **for** $k = 0, 1, 2, \cdots$ **do**
3: Collect trajectories D_k on policy $\pi_k = \pi_{\theta_k}$
4: Estimate all advantages $A^{\pi_{\theta_k}}(s_t, a_t)$
5: Form sample estimates for policy gradients $\hat{g}_k$ (using advantage estimates)
6: Form sample estimates for the Hessian of KL divergence
7: Compute the Natural Policy Gradient update

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$$

8: **end for**

Fisher Information Matrix and KL Divergence

- Let $p(x|\theta)$ be a probability distribution parameterized by θ.
- Score function of a parameterized probability distribution is given by

$$s(\theta) = \nabla_\theta \log p(x|\theta)\,,$$

- For a parameter vector θ, Fisher Information Matrix is given by,

$$\mathrm{F} = \mathop{\mathbb{E}}_{p(x|\theta)} \left[\nabla_\theta \log p(x|\theta)\, \nabla_\theta \log p(x|\theta)^{\mathrm{T}} \right] .$$

- The sample estimate of the above expectation is given by,

$$\mathrm{F} = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log p(x_i|\theta)\, \nabla_\theta \log p(x_i|\theta)^{\mathrm{T}} . \tag{1}$$

- **Claim** : Fisher Information Matrix F is the Hessian of KL-divergence between two probability distributions $p(x|\theta')$ and $p(x|\theta)$ evaluated at $\theta' = \theta$

$$\mathrm{H}_{\mathrm{KL}[p(x|\theta) \,\|\, p(x|\theta')]} = \mathrm{F}\,.$$

Properties of Natural Policy Gradient

- Natural policy gradient algorithm gives an update-rule in which updates are pre-multiplied by H^{-1}
- The Hessian of the KL-divergence is the Fischer Information Matrix given by

$$\mathrm{F} = \mathop{\mathbb{E}}_{\pi_\theta}\left[\nabla \log \pi_\theta(\cdot|s)\, \nabla \log \pi_\theta(\cdot|s)^{\mathrm{T}}\right]$$

- The NPG direction $H^{-1}g$ is **co-variant**; i.e. it points in same direction irrespective of the parametrization that is used to compute it

Relationship of Natural Gradient to Policy Gradient

Policy Gradient Formulation

Consider the following optimization problem

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \left[\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) \right]$$
$$\text{such that } \left\| \theta - \theta_k) \right\|^2 \leq \delta$$

After **linearising** the objective, the optimization problem is now,

$$\theta_{k+1} = \arg\max_{\theta} g^T(\theta - \theta_k) \text{ such that } (\theta - \theta_k)^2 \leq \delta$$

This is the original policy gradient problem !!

We move a small distance in parameter space in the direction of the gradient

Natural Gradient Formulation

Natural policy gradient problem is given by,

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \left[\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) \right]$$
$$\text{such that } \bar{D}_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$

After **linearising** the objective and **quadratifying** the constraint, fhe optimization problem is then given by,

$$\theta_{k+1} = \arg\max_{\theta} g^T(\theta - \theta_k) \text{ such that } \frac{1}{2}(\theta - \theta_k)^T F \ (\theta - \theta_k) \leq \delta$$

Relationship between Formulations

- Vanilla policy gradient has the right objective but *"incorrect"* constraint (Euclidean penalty instead of KL penalty)
- Recall that, policy iteration (from MDP lectures) obtain policy improvement with no constraint

Other Algorithms

Truncated Natural Policy Gradient

- **Problem** : For neural networks, the dimensionality of parameter θ are high. High computational cost in inverting the matrix H
- **Solution** : Use the **conjugate gradient algorithm** to compute $H^{-1}g$ without inverting H
- Resultant algorithm : Truncated Natural Policy Gradient
- ACTKR algorithm uses KFAC technique to solve the inverse Hessian computation problem

Problems with Natural Policy Gradient Update

- Another problem with NPG update is that - might not be robust to trust region size δ
  - ★ δ may be too large in some iterations and can degrade the performance
- Because of quadratic approximation, the KL-divergence constraint may be violated
- Monotonic improvement may not occur in all iterations

TRPO : Line Search Algorithm

- Enforce improvement in surrogate (i.e. $\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) \geq 0$)
- Enforce KL constraint
- How ? Backtracking line search with exponential decay

Algorithm Line Search for TRPO

Compute the proposed policy step $\Delta_k = \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$
for $j = 0, 1, 2, \cdots N$ **do**
  Compute proposed update $\theta = \theta_k + \alpha_j \Delta_k$
  If $\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) \geq 0$ and $\bar{D}_{KL}(\theta || \theta_k) \leq \delta$
    Accept the update $\theta = \theta_k + \alpha_j \Delta_k$
  Else
    Find another α_j (Reduce α_j)
end for

Algorithm : Trust Region Policy Optimization

Algorithm Trust Region Policy Optimization

1: Initialize π_0
2: **for** $k = 0, 1, 2, \cdots$ **do**
3: Collect trajectories D_k on policy $\pi_k = \pi_{\theta_k}$
4: Estimate all advantages $A^{\pi_{\theta_k}}(s_t, a_t)$
5: Form sample estimates for policy gradients $\hat{g}_k$ (using advantage estimates)
6: Form sample estimates for the Hessian of KL divergence / FIM
7: Use conjugate gradient to obtain FIM estimate H^{-1}
8: Estimate step size α using backtracking line search to enforce KL constraint and monotonic improvement
9: Compute the Natural Policy Gradient update

$$\theta_{k+1} = \theta_k + \alpha \, \Delta_k$$

10: **end for**

Proximal Policy Optimization

Proximal Policy Optimization is a family of methods that approximately enforce without actually computing the natural gradient

- **Adaptive KL Penalty**

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \left[\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) - \beta \bar{D}_{KL}(\pi_\theta || \pi_{\theta_k}) \right]$$

Penalty co-efficient β is changed between iterations to approximately enforce KL constraint

- **Clipped Objective** (Simpler to implement, no need to check KL constraint; works well)

$$\mathcal{L}^{CLIP}_{\pi_{\theta_k}}(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_k}} \left[\sum_{t=0}^{T} \min \left(r_t(\theta) A_t^{\pi_{\theta_k}}, clip(1-\varepsilon, 1+\varepsilon) A_t^{\pi_{\theta_k}} \right) \right]$$

where $r_t(\theta)$ is the importance sampling ratio between target policy π_θ and behaviour policy π_{θ_k} and policy update takes place as

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \mathcal{L}^{CLIP}_{\pi_{\theta_k}}(\pi_\theta)$$

PPO : KL Constraint

Algorithm Proximal Policy Optimization : KL Constraint

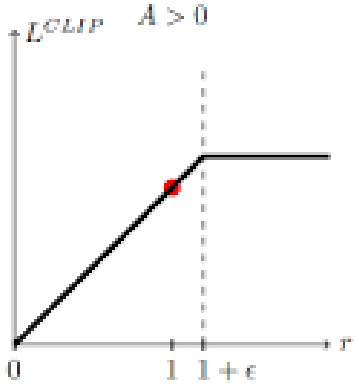
Initialize π_0 initial KL penalty β_0 and target KL divergence δ
for $k = 0, 1, 2, \cdots$ **do**
  Collect set of trajectories on policy $\pi_k = \pi_{\theta_k}$
  Estimate advantages $A^{\pi_{\theta_k}}(s_t, a_t)$
  Compute policy update by solving (using SGD)

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \left[\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta) - \beta_k \bar{D}_{KL}(\pi_\theta || \pi_{\theta_k}) \right]$$

  If $\bar{D}_{KL}(\pi_{\theta_{k+1}} || \pi_{\theta_k}) \geq 1.5\delta$ then
    $\beta_{k+1} = 2\beta_k$
  Else if $\bar{D}_{KL}(\pi_{\theta_{k+1}} || \pi_{\theta_k}) \leq \delta / 1.5$
    $\beta_{k+1} = \beta_k / 2$
end for

- Initial KL penalty not that important; it adapts quickly
- Some iterations may violate KL constraint, but most don't

Clipped Objective : Rationale



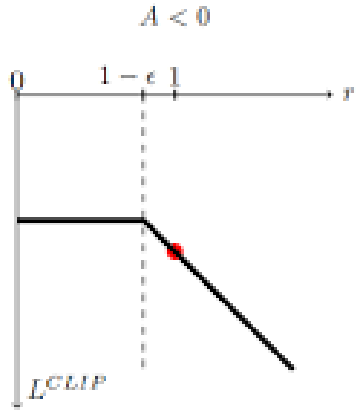


Figure Source: https://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe

PPO : Clipped Objective

Algorithm Proximal Policy Optimization : Clipped Objective

1: Initialize π_0
2: **for** $k = 0, 1, 2, \cdots$ **do**
3: Collect set of trajectories on policy $\pi_k = \pi_{\theta_k}$
4: Estimate advantages $A^{\pi_{\theta_k}}(s_t, a_t)$
5: Compute policy update by solving (using SGD)

$$\pi_{\theta_{k+1}} = \arg\max_{\pi_\theta} \mathcal{L}^{CLIP}_{\pi_{\theta_k}}(\pi_\theta)$$

where

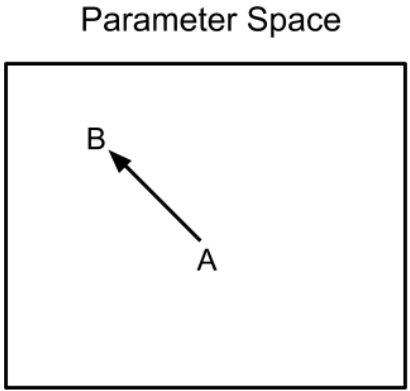
$$\mathcal{L}^{CLIP}_{\pi_{\theta_k}}(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_k}} \left[\sum_{t=0}^{T} \min \left(r_t(\theta) A_t^{\pi_{\theta_k}}, clip(1-\varepsilon, 1+\varepsilon) A_t^{\pi_{\theta_k}} \right) \right]$$

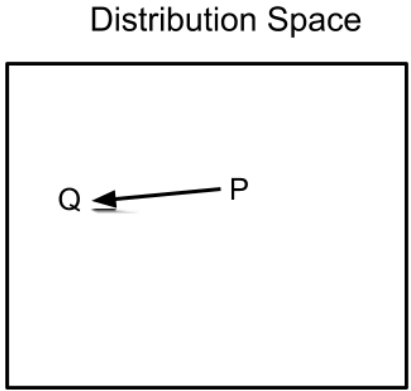
6: **end for**

- Clipping prevents the new policy $\pi_{\theta_{k+1}}$ to go far away from π_{θ_k}
- Clipping is simpler to implement and works well in practice

Insights into Natural Gradients

Parameter and Distribution Space





Natural Gradients

- Natural gradient is not the gradient with respect to model parameters; rather, it is the gradient on the manifold of probabilistic models
- Traditional gradients considers how perturbations of the model parameters affect the loss function; Natural gradients considers how the loss function changes when the probabilistic model moves a little bit on the manifold of model family
- Because the model family does not change with respect to parameterizations, the natural gradient will also remain the same.
- Natural gradients are invariant to parameterization. That is, Natural gradients under two different parameterizations correspond to the same "gradient entity" on the manifold

Probabilistic Models and Parameterization

Consider the following model that captures the joint distribution between two random variables x and y

$$p_\theta(x, y) = p(x)\mathcal{N}(y \mid \theta x + b, \sigma^2), \theta \in \mathbb{R},$$

where θ is a parameter and b and σ are constants.

Let $\theta = 2\mu$. We can rewrite the above model in terms of new parameter μ as

$$p_\mu(x, y) = p(x)\mathcal{N}(y \mid 2\mu x + b, \sigma^2), \mu \in \mathbb{R}.$$

Note that even if $p_\theta(x, y)$ and $p_\mu(x, y)$ have different analytical forms, they represent the same (family) distribution. Specifically,

$$p_{\theta=a}(x, y) \equiv p_{\mu=a/2}(x, y), \text{ for any } a \in \mathbb{R}$$

Construct a negative log-likelihood as loss function for the distribution $p(x, y)$ as

$$\begin{aligned}
\mathcal{L}(p(x,y)) &= -[\log p(x,y)] \\
&= \left[\frac{1}{2\sigma^2}(\theta x + b - y)^2 + \log\sqrt{2\pi}\sigma\right] \quad (\text{for } p_\theta(x,y)) \\
&= \left[\frac{1}{2\sigma^2}(2\mu x + b - y)^2 + \log\sqrt{2\pi}\sigma\right] \quad (\text{for } p_\mu(x,y))
\end{aligned}$$

Gradient Descent and Parameterization

- For $p_\theta(x, y)$ the update rule is given by $\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y))$ where α is the step size with

$$\nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y)) = \left[\frac{x}{\sigma^2} (\theta_t x + b - y) \right]$$

- For $p_\mu(x, y)$ the update rule is given by $\mu_{t+1} = \mu_t - \alpha \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y))$ where α is the step size with

$$\nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y)) = \left[\frac{2x}{\sigma^2} (2\mu_t x + b - y) \right]$$

Gradient Descent and Parameterization

- Let $\theta_t = 2\mu_t = a$ for some $a \in \mathbb{R}$
- That is, at the t-th step of the gradient descent, the parametric probabilistic models $p_{\theta=a}(x, y)$ and $p_{\mu=a/2}(x, y)$ represent the same distribution
- Because

$$\nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y))|_{\theta_t=a} = \frac{1}{2} \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y))|_{\mu_t=a/2}$$

  we will have,

$$\begin{aligned} \theta_{t+1} &= \theta_t - \alpha \nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y)) \\ &= 2\mu_t - \frac{\alpha}{2} \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y)) \\ &\neq 2\mu_t - 2\alpha \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y)) \\ &= 2\mu_{t+1} \end{aligned}$$

- Hence the $t+1$-th optimization step will result in different probabilistic models $p_{\theta_{t+1}}(x, y) \not\equiv p_{\mu_{t+1}}(x, y)$

Gradient Descent and Parameterization

- One step of gradient descent will result in different models depending on which parameterization is used
- Hence, requires carefully choosing the parameterization to avoid hindering optimization
- Not all optimization procedures are parameterization dependent. For example, the Newton-Raphson method is invariant to affine transformations of model parameters
- Natural gradient methods are invariant to arbitrary differentiable transformations of model parameters when the learning rate is small enough

Sanity Check : Natural Gradients

Recall that the natural gradient of a loss function is given by,

$$\widetilde{\nabla}_\theta \mathcal{L}(p_\theta(x,y)) := \mathbf{F}_\theta^{-1} \nabla_\theta \mathcal{L}(p_\theta(x,y)),$$

where

$$[\mathbf{F}_\theta]_{i,j} := \mathbb{E}_{p_\theta(x,y)} \left[\left(\frac{\partial}{\partial \theta_i} \log p_\theta(x,y) \right) \left(\frac{\partial}{\partial \theta_j} \log p_\theta(x,y) \right) \right],$$

For different parameterizations of Gaussian conditional distributions, the derivatives of the log densities are respectively

$$\begin{aligned} \frac{\partial}{\partial \theta} \log p_\theta(x,y) &= \frac{(y-\theta x - b)x}{\sigma^2} \\ \frac{\partial}{\partial \mu} \log p_\mu(x,y) &= \frac{2(y-2\mu x - b)x}{\sigma^2} \end{aligned}$$

With $\theta_t = 2\mu_t$, we can conclude that $\mathbf{F}_{\mu=\mu_t} = 4\mathbf{F}_{\theta=\theta_t}$. Using

$$\widetilde{\nabla}_\mu \mathcal{L}(p_\mu(x,y))|_{\mu=\mu_t} = \frac{1}{2} \widetilde{\nabla}_\theta \mathcal{L}(p_\theta(x,y))|_{\theta=\theta_t}$$

we can conclude that, $\theta_{t+1} = 2\mu_{t+1}$

Slide Credit: https://ermongroup.github.io/blog/geo/

Metric Tensor (Distance Function)

In a Reimannian space, the distance between points v and $v + \delta v$ can be expressed as

$$dist^2(v + \delta v, v) = \delta v^T \mathcal{G}(v) \delta v$$

The $\mathcal{G}(v)$ is the **metric tensor** (or the distance function) in the space and it depends on the co-ordinate system that represents vector v

Example

The Euclidean space $\mathbb{R}^2$ can be represented in Cartesian coordinates or polar coordinates. If $v = (x, y)$ in Cartesian coordinates (metric tensor is identity matrix)

$$dist^2(v + \delta v, v) = \delta x^2 + \delta y^2 = \delta v^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \delta v$$

In polar coordinates,

$$dist^2(v + \delta v, v) = (\delta r, \delta \theta)^T \underbrace{diag(1, r^2)}(\delta r, \delta \theta)$$

The Fischer Information matrix F is the metric tensor in policy space.

Covariance of Natural Policy Gradient

Consider the same vector and vector difference in two coordinate systems

- In system 1 (v), we write $(v; \delta v)$, and the metric tensor is written as $\mathcal{G}_v$
- In system 2 (w), we write $(w; \delta w)$, and the metric tensor is written as $\mathcal{G}_w$
- Note that $v = w$ (same vector in different parametrizations).

Because the deltas are also equal. they are related by,

$$\delta v_i = \sum_j \frac{\partial v_i}{\partial w_j} \partial w_j \implies \delta v = A^T \delta w, \text{ where } A_{ji} = \frac{\partial v_i}{\partial w_j}$$

The distances must be the same in both, so metrics are related as follows:

$$\begin{aligned} dist^2(v, v+\delta v) &= dist^2(w, w+\delta w) \\ dist^2(v, v+\delta v) &= \delta v^T \mathcal{G}_v \delta v = \delta w^T A \mathcal{G}_v A^T \delta w \\ dist^2(w, w+\delta w) &= \delta w^T \mathcal{G}_w \delta w \implies \mathcal{G}_w = A \mathcal{G}_v A^T \end{aligned}$$

Gradients are related by chain rule:

$$[g_w]_j = \frac{\partial f}{\partial w_j} = \sum_i \frac{\partial v_i}{\partial w_j} \frac{\partial f}{\partial v_i} \implies g_w = A g_v$$

Slide Credit: Joshua Aicham : OpenAI

Covariance of Natural Policy Gradient

Now we ask the question if $\Delta v = \mathcal{G}_v^{-1} g_v$ and $\Delta w = \mathcal{G}_w^{-1} g_w$ are same vectors

$$\begin{aligned} \Delta w &= \mathcal{G}_w^{-1} g_w \\ &= (A \mathcal{G}_v A^T)^{-1} A g_v \\ &= (A^T)^{-1} \mathcal{G}_v^{-1} A^{-1} A g_v \\ &= (A^T)^{-1} \Delta v \end{aligned}$$

$$\implies A^T \Delta w = \Delta v$$

They are the same vectors and the natural gradient $H^{-1} g$ is invariant to parameterization

Recommended Reading

- "Why Natural Gradient?" S. Amari and S. C. Douglas, 1998
- "Natural Policy Gradient," Sham Kakade, 2001
- "Reinforcement Learning of Motor Skills with Policy Gradients," Jan Peters and Stefan Schaal, 2008
- "Trust Region Policy Optimization," Schulman et al. 2015
- "Benchmarking Deep Reinforcement Learning for Continuous control," Duan et al. 2016
- "Proximal Policy Optimization Algorithms," Schulman et al. 2017