1 Lecture 5: Policy Gradient and Search

Policy gradient/ search is influential in NLP/ Proximal Policy Optimization (training GPT). The core idea:

Intuition of Gradient Search

Approximate $V^\pi(s) \approx V_W(s)$ and $Q_w(s,a) \approx Q^\pi(s,a)$ by adjusting weight w.
Policy gradient: rather than generating policy from value (ϵ-greedy), directly parametrize policy with θ, i.e.

$\pi_\theta(s,a) = \mathbb{P}[a|s;\theta]$: optimize $V(\theta)$ to find policy π

The brief classification of policy gradient is as follows:

	Value-based	Policy-based	Actor-critic
Value function	learned	not present	learned
Policy	implicit (ϵ-greedy)	learned	learned

Instead of deterministic/ ϵ-greedy policies, need to focus heavily on **stochastic** for direct policy search!

- Repeated Trials, e.g. In rock paper scissors (of many rounds), deterministic policy is easily exploited by adversary.
- Boundary Condition, e.g. In gridworld, bound to only move one direction (else get stuck/ traverse for long time for slow convergence).

In short, poliy objective functions have the following intuition:

Policy Objective Summary

- Goal: Given policy $\pi_\theta(s,a)$, find best parameter θ.
  Inherently, an optimization of $V(s_0, \theta)$.
- Purpose: Measure quality for policy π_θ with policy value at start state s_0.
- Works for: both episodic/ continuing and infinite horizons.

1.1 Gradient Free Policy Optimization

They are great simple baselines.

- Examples: Hill Climbing, Genetic Algo (evolution strategies, cross-entropy method, covariance matrix adaption)
- Known for decades but embarrassingly well: rivals standard RL techniques!
- Advantages: Flexible for any policy parameterization, easily to parallelize
  Disadvantage: Less sample efficient (ignores temporal structure)

1.2 Policy Gradient

This section focuses on gradient descent; other popular algos include conjugate gradient and quasi-newton methods.
Usually assume **Episodic MDPs** for easy extension of objectives. The method:

Define $V(\theta) = V(s_0, \theta)$, i.e. the value fucntion depending on policy parameters. Then:

- Search the local maximum in $V(s_0, \theta)$ with gradient increments:

$$\Delta\theta = \alpha\nabla_\theta V(s_0, \theta) = \alpha \begin{pmatrix} \dfrac{\partial V(s_0,\theta)}{\partial\theta_1} \\ \vdots \\ \dfrac{\partial V(s_0,\theta)}{\partial\theta_n} \end{pmatrix}$$

- Assumption: π_θ differentiable (and known gradient $\nabla_\theta \pi_\theta(s, a)$)
- We can rewrite policy value $V(s_0, \theta)$ in the following ways:
  1. **Visited States and Actions**: $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta, s_0\right]$
  2. **Weighted Average of Q-values by Actions**: $\sum_a \pi_\theta(a|s_0)Q(s_0, a, \theta)$
  3. **Trajectories Sampled using** π_θ: $\sum_\tau P(\tau|\theta)R(\tau)$

In particular, it is of interest to consider writing $V(s_0, \theta)$ in trajectory form:

To find the best policy parameter θ, we consider

$$\arg\max_\theta V(\theta) = \arg\max_\theta \sum_\tau P(\tau; \theta)R(\tau)$$

Taking gradient,

$$\begin{aligned} \nabla_\theta V(\theta) =& \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \\ & \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) (R \text{ being indep of } \theta) \\ & \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)R(\tau) \\ & \sum_\tau R(\tau)P(\tau;\theta)\nabla_\theta \log P(\tau;\theta) \qquad \text{(log-likelihood)} \end{aligned}$$

Approximate in practice using m sample trajectories under π_θ:

$$\nabla_\theta V(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)}, \theta)$$

But trajectories can be decomposed into states and actions:

$$\begin{aligned} \nabla_\theta \log P(\tau^{(i)};\theta) =& \nabla_\theta \log\left[\mu(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t)P(s_{t+1}|a_{t+1}, s_{0:t}, a_{0:t})\right] \\ =& \sum_\tau \nabla_\theta \log \pi_\theta(a_t|s_t) \end{aligned}$$

Here

- We call $\sum_\tau \nabla_\theta \log \pi_\theta(a_t|s_t)$ the **score function**.
- the initial state $\mu(s_0)$ is constant; dynamics model $P(s_{t+1}|a_{t+1}, s_{0:t}, a_{0:t})$ is invariant to θ.
- In other words, no dynamics model is required to approximate the policy parameter θ.

Questions

1. Why trajectory form is practical ("better in training")?
2. Why is log-likelihood ratio important here? What does it enable?

1.3 Selecting a Right Policy

1.3.1 Softmax Policy

- In softmax, **exponentially weight** quantities of linear combination of features as probabilities (that add to 1):

$$\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\sum_a e^{\phi(s,a)^T\theta}}$$

- Then the score function can be written as $\nabla_\theta(s,a) = \phi(s,a) - \mathbb{E}_{\pi_\theta[\phi(s,\cdot)]}$

1.3.2 Gaussian Policy

- A normal distribution is natural for continuous action spaces; at times used by deep NN.
- Action $a \sim N(\mu(s), \sigma^2)$. Mean $\mu(s) = \phi(s)^T\theta$ is a linear combination of state features.
- Then the score function can be written as $\nabla_\theta(s,a) = \phi(s,a) - \mathbb{E}_{\pi_\theta}\left[\phi(s,\cdot)\right]$

When to use policy gradient?

1. Differentiable reward functions
2. No dynamics required
3. Useful for both infinite horizon and episodic settings
4. Intuition: $R(\tau^{(i)})$ is replacable by other functions that *measures the wellness of sample x*
   Essentially, moving in the direction of $\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i|\theta)$ pushes up the log probability proportionally.

What are the purposes of selecting Softmax and Gaussian policies?

The generalization is as follows:

Policy Gradient Theorem

Assumption: Differentiable Policy $\pi_\theta(s,a)$, objective $J = \begin{cases} J_1(\text{Episodic}) \\ J_{avR}(\text{Avg. Reward over time}) \\ \frac{1}{1-\gamma} J_{avV}(\text{Avg. Value over time}) \end{cases}$

In any case, the policy gradient is $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)\right]$

Summary and Improvements of Policy-based RL

Criteria	Advantages	Disadvantages
Convergence	Better Properties	Typical Local Optimum
Flexibility	Effective in high-dim./ continuous action spaces Can learn stochastic policies	Inefficient, high-variance policy evaluation

Currently, use

$$\nabla_\theta V(\theta) = \frac{1}{m}\sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}, s_t^{(i)})$$

to estimate

$$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau\left[\left(\sum_{t=0}^{T-1} r_t\right) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

It is unbiased but noisy (high variance)!

On the high variance, several remedies serve as improvements:

1.3.3 Fix 1: Temporal Structure

- Focus on single reward item at once:

$$\nabla_\theta \mathbb{E}[r_{t'}] = \mathbb{E}[r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

- Sum up over t to obtain:

$$\begin{aligned} V(\theta) = \nabla_\theta \mathbb{E}[R] &= \mathbb{E}[\sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)] \\ &= \mathbb{E}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}] \quad \text{(because later decisions don't influence past rewards} \end{aligned}$$

- This can be further simplified: trajectory $\tau^{(i)}$ has return $G_t^{(i)} = \sum_{t'=t}^{T-1} r_{t'}^{(i)}$. Hence

$$\nabla_\theta \mathbb{E}[R] \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

Monte-Carlo Policy Gradient

Making use of likelihood ratio / score function and temporal structure, update param θ with

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$$

after initializing θ arbitrarily.

Q: How does Temporal structure reduce variance?

1.3.4 Fix 2: Baseline Function

As iteration costs time and computational resources, we desire quick convergence to local optima.

Baselines: Unbiasedness and Other Considerations

$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau\left[\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$

Why it works?

- Unbiased for any b if b is a function of s but not θ, because $\mathbb{E}_\tau\left[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)\right] = 0$
- A near-optimal baseline choice is the expected return $b(s_t) \approx \mathbb{E}\left[\sum_{t'=t}^{T-1} r_{t'}\right]$
- Other choices: State-value function $V^\pi(s) = \mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a)\right]$

Interpretation: Increase logprob of action at proportionally to how much returns $\sum_{t'=t}^{T-1} r_{t'}$ are better than expected

Q: Intuitively, what are the uses of baseline functions? How does variance get reduced?

Core idea: Break down t in the summation:

$$\begin{aligned}\mathrm{Var}[\nabla_\theta[R]] &= \mathrm{Var}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta)(R_t(s_t) - b(s_t))] \\ &\approx \sum_{t=0}^{T-1} \mathbb{E}_\tau\left[\mathrm{Var}[\nabla_\theta \log \pi(a_t|s_t;\theta)(R_t(s_t) - b(s_t))]\right]\end{aligned}$$

- As we sample from trajectories τ (MC),
- For each t, write variance $\mathrm{Var}[X]$ as $\mathbb{E}[X^2] - (\mathbb{E}[X])^2$: $\mathbb{E}\left[((\nabla_\theta \log \pi(a_t|s_t;\theta))(R_t(s_t) - b(s_t)))^2\right] - \mathbb{E}\left[((\nabla_\theta \log \pi(a_t|s_t;\theta))(R_t(s_t) - b(s_t)))\right]^2$
- Second term is not affected by choice of $b(s)$ (unbiased = same expectation). The variance equals $\arg\min_b \ \mathbb{E}\left[((\nabla_\theta \log \pi(a_t|s_t;\theta)))^2\left((G_t(s_t) - b(s_t))\right)^2\right]$ $= \arg\min_b \ \mathbb{E}_{s\sim d^\pi}\left[\mathbb{E}_{a\sim\pi(\cdot|s),G|s,a}\left[((\nabla_\theta \log \pi(a_t|s;\theta)))^2\left((G_t(s_t) - b(s_t))\right)^2\right]\right]$
- A **weighted least-squares** problem that minimizes

  $$\sum_i \sum_t |b(s_t^i) - G_t^i|^2 \qquad (G \text{ (or } A = G - b) \text{ being target})$$

  with solution (after taking zero gradient)

  $$b(s) \approx \mathbb{E}_{a\sim\pi(\cdot|s),G|s,a}[G_t(s)]$$

Q: What does it mean by "Near optimal"? When is it not?

1.3.5 Fix 3: Alternative to MC

The original G_t^i estimates expected discounted sum of returns (from single roll): unbiased but high variance. To solve this:

- Leverage bootstrapping and approximation (similar to TD vs MC and VFA) to introduce bias.
- Use "Critic" to estimate the ratio $\frac{V}{Q}$. The popular class of "Actor-critic" methods explicitly represents (and updates) policy and values.
- Essentially, replace $\sum_{t'=t}^{T-1} r_{t'} - b(s_t)$ [Vanilla MC] with Q-values $(Q(s_t, a_t; \mathbf{w}) - b(s_t))$ [This is essentially TD] or advantage function $\hat{A}^\pi(s_t, a_t)$ where $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$.

Alternative Targets to MC Estimators

With vanilla MC, the gradient is estimated by

$$\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Proposed replacements:

1. N-step estimators:
   - $\hat{R}_t^{(1)} = r_t + \gamma V(s_{t+1})$, $\hat{R}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$
   - $\hat{R}_t^{(\infty)} = r_t + \gamma r_{t+1} + \gamma r_{t+2} + \dots$
2. Advantage estimators, by *subtracting baselines of $V(s_t)$ from above*
   - $\hat{A}_t^{(1)} = \hat{R}_t^{(1)} - V(s_t) = r_t + \gamma V(s_{t+1}) - V(s_t)$ *(low variance, high bias)*
   - $\hat{R}_t^{(\infty)} - V(s_t) = r_t + \gamma r_{t+1} + \gamma r_{t+2} + \dots - V(s_t)$ *(high variance, low bias)*

What does the word "critic" here mean?

A summary of policy gradient:

Intermediate Summary of PG

- Core idea: $\nabla_\theta V(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)\right]$

Optimize $\arg\max_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ with SGD on θ:

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

- State-action pairs with higher $\hat{Q}$ increases probabilities in average
- Direction of θ dependent on gradient of $\ln \pi(S_t, A_t, \theta)$ AND Q-values/ returns
- NOT guaranteed to converge to global optima (just local!)

1.4 Limitations of Vanilla Policy Gradient

A PG algo should minimize # iterations to reach a good (probably suboptimal) policy within time. The limitations of vanilla PG:

1.4.1 Poor sample efficiency

Variance reduces slowly, because PG is an **on-policy** expectation: Data immediately discarded after just one gradient step

- Collect sample estimates from trajectories of same policy (more stable), or other policies (off-policy, less stable).
- Opportunity: Can we take multiple gradient steps from old data before new policy?

Problems of Determining Gradient Step

Problem: Difficult to handle step size (dist. in parameter space $\neq$ dist. in policy space)

- e.g. Matrices in tabular case $\Pi = \{\pi : \pi \in \mathbb{R}^{|S|\times|A|}, \sum_a \pi_{s_a} = 1, \pi_{s_a} \geq 0\}$

  VS steps of policy gradient in parameter space $\implies$ unable to map/ gauge size!
- SGD of $\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$ is subject to performance collapse with large steps!
  e.g. logistic function: small $\Delta\theta$ leads to big policy changes

The solution to restrict policy change from more than intended is through policy performance bounds:

Distance in Value to Policy

To respect distance mapping in policy space, exploit relationships between policy performance:

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

Here $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)$ is the **weighted** distribution of states. Making use,

Now, rewrite the objective:

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

Why rewrite?

1. Now, performance of π' is defined in advantages from π.
2. Requires trajectories sampled from π' (desired: from π because our tweak features $s \sim d^\pi$).

We have a useful approximation:

Relative Policy Performance Bounds

$$J(\pi') - J(\pi) \approx \mathbb{L}_\pi(\pi') \qquad \text{for close } \pi' \text{ and } \pi \ (d^{\pi'} = d^\pi)$$

Approximation quality is ensured by relative policy performance bounds:

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

But what is $\mathbb{E}_{s\sim d^\pi}\left[D_{KL}(\pi'||\pi)[s]\right]$?

KL-Divergence

Such divergence measures distance between probability distributions:

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

KL satisfies $D_{KL}(P||P) = 0, D_{KL}(P||Q) \geq 0, D_{KL}(P||Q) \neq D_{KL}(Q||P)$.
Between policies, $D_{KL}(\pi'||\pi)[s] = \sum_{a\in A}\pi'(a|s)\log\frac{\pi'(a|s)}{\pi(a|s)}$

After the approximation, we can optimize using trajectories sampled from the old policy π!

Policy Optimization under Bounded KL Approximation

Policy improvement can be estimated by sampling from old policy π!

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

1.5 Proximal Policy Optimization (PPO): Approximation in Action

PPO penalizes large policy change iterations. There are two methods:

1. regularization term on KL-divergence (Adaptive Penalty)
2. pessimistic objective on far-away policies (Clipped Objective)

PPO Iterative Algorithm - Adaptive Penalty

$$\theta_{k+1} = \arg\max_\theta L_{\theta_k}(\theta) - \beta \bar{D}_{KL}(\theta||\theta_k)$$

Here, KL-divergence is an expectation:

$$\bar{D}_{KL}(\theta||\theta_k) = \mathbb{E}_{s\ simd^{\pi_k}} D_{KL}\left(\theta_k(\cdot|s), \pi_\theta(\cdot|s)\right)$$

Penalty coefficient β_k changes between iterations:

- Initiate policy param θ_0, initial KL penalty β_0, target KL-divergence δ
- Compute policy update (iterate θ) by K steps of minibatch SGD (via Adam)
- Control KL-divergence to be around δ by adjusting penalty:
  If $\bar{D}_{KL}(\theta||\theta_k) \geq 1.5\delta$ then $\beta_{k+1} = 2\beta_k$;
  elif $\bar{D}_{KL}(\theta||\theta_k) \leq \frac{\delta}{1.5}$ then $\beta_{k+1} = \frac{1}{2}\beta_k$

This KL penalty is called "adaptive" because of how β_k changes (adapts quickly) according to KL-divergence.

An alternative approach is clipping: restrict via pessimistically treating objective value far away from θ_k.

Outline of Clipped Objective on Policy Changes

- Define relative probability change: $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$
- The new objective is

$$L^{\text{Clip}}_{\theta_k}(\theta) = \mathbb{E}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min\left(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_k}\right)\right]\right]$$

  In other words, $r_t(\theta)$ is clipped between $(1-\epsilon, 1+\epsilon)$; hyperparameter ϵ usually set at 0.2.

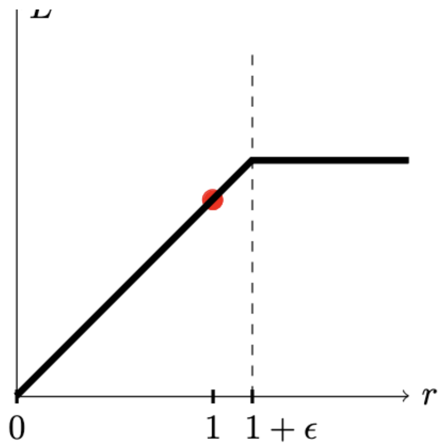
- Policy update: $\theta_{k+1} = \arg\max\ \ _\theta L^{\text{Clip}}_{\theta_k}(\theta)$

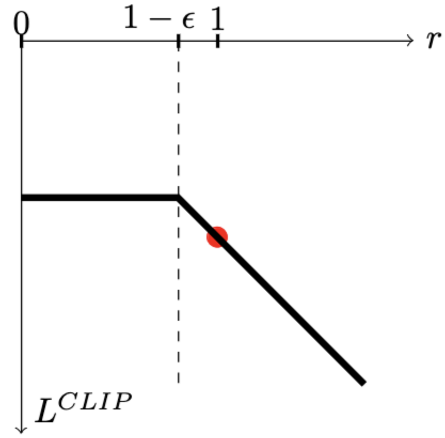
Here, Clipping discentivizes going far from θ_{k+1}:

- Left graph below: when the advantage function $A > 0$; right graph when $A < 0$.
- $L^{\text{Clip}}_{\theta_k}(\theta)$'s increase is suppressed at extreme values towards the sign of A.
- Clipping is simple to implement but works well compared to KL penalty.

PPO's performance consistently tops other algos, hence wildly popular.

- Today, it is a key component of ChatGPT (readings: OpenAI blog (2017), Publication by Schulman et. al. (2017))
- Different outcomes with reward scaling/ learning rate annealing.





Q1: Why KL-divergence applied is an expectation? Q2: How does the two methods compare?