Policy Gradient In Practice

CSCI 5951-F
Ron Parr
Brown University

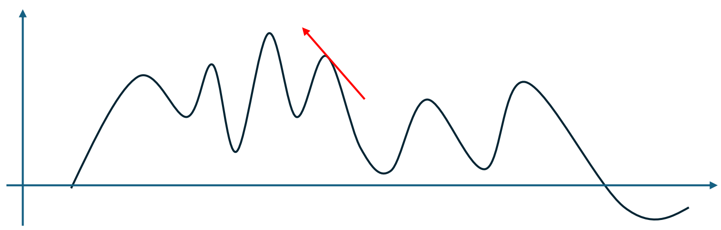
Recap

- Policy gradient allows to search directly in policy space ☺
- Variance is high ☹

- Baseline subtraction (via an advantage function, which can be computed from Q-functions) helps
  - Trades some bias for variance
  - Can still have noisy gradients

More about gradients

- Gradient descent/ascent is our most basic tool in modern ML
- Recall: First order approximation to complicated function
- Things that affect quality of first order approximation:
  - Noise
  - Smoothness of function
  - Size of region around the approximation (step size)



Policy value often is not very smooth

- Think about value of policy is through distribution over states
- Let $\rho^{(0)}$ be your initial state distribution, $\rho^{(i)}$ be distribution at time i
- $\rho^{(i+1)} = \rho^{(i)T} P_\pi$
- $\rho^{(i)} = \rho^{(0)T} P_\pi^i$
- $U(\theta) = \sum_{i=0}^{\infty} \rho^{(i)^T} R$
- P_π is parameterized by θ
- Effect is quadratic in 2 time steps, cubic in 3, etc.
- Function is very curvy, so gradient quickly becomes inaccurate

Goal steepest descent (or ascent)

- Get the most out of each step for a constant step size
- Want: Direction of descent that maximizes progress on objective fn.
- Can we be smarter about disentangling correlated effects on gradient?

What if we redefine distance?

- Idea: Warp space to compensate for interactions between parameters as well as scaling issues
- Define G to be some positive definite matrix
- Redefine distance: $|d\theta|^2 \equiv \sum_{ij} G_{ij}(\theta) d\theta_i d\theta_j = d\theta^T G(\theta) d\theta$
- Steepest descent direction is then:

$$G^{-1} \nabla U(\theta)$$

But what is a good choice of G?

- Fisher information matrix at any state tells use how parameters interact:

$$F_s(\theta) \equiv E_{\pi(a;s,\theta)}[\frac{\partial \log \pi(a;s,\theta)}{\partial \theta_i} \frac{\partial \log \pi(a;s,\theta)}{\partial \theta_j}]$$

- Total correction is weighted by visitation frequencies:

$$F(\theta) \equiv E_{\rho^{\pi}(s)}[F_s(\theta)]$$

Avoiding overstepping

- Natural gradient helps adjust the direction
- Step size is still a problem
- Want to take the largest possible step without overshooting
- Multiple approaches
  - TRPO: Uses line search and various approximations to maximize step size
  - PPO: Uses a “clamped” objective to avoid overshooting

Digression: Reproducibility

- TRO vs PPO
- PPO originally introduced as a simpler alternative to TRPO
- Was also shown to perform better in many cases
- Engstrom et al. (IMPLEMENTATION MATTERS IN DEEP POLICY GRADIENTS: A CASE STUDY ON PPO AND TRPO) investigate this:
  - Find 9 optimizations in PPO not (clearly) documented as main improvements
  - "We find that much of the PPO's observed improvement in performance comes from seemingly small modifications to the core algorithm that either can be found only in a paper's original implementation, or are described as auxiliary details and are *not* present in the corresponding TRPO baselines."
  - "Ultimately, we discover that the *PPO code-optimizations are more important in terms of final reward achieved* than the choice of general training algorithm (TRPO vs. PPO). "

Performance comparison

STEP	MUJOCO TASK WALKER2D-V2	 HOPPER-V2	 HUMANOID-V2
PPO	3292 [3157, 3426]	2513 [2391, 2632]	806 [785, 827]
PPO-M	2735 [2602, 2866]	2142 [2008, 2279]	674 [656, 695]
TRPO	2791 [2709, 2873]	2043 [1948, 2136]	586 [576, 596]
TRPO+	3050 [2976, 3126]	2466 [2381, 2549]	1030 [979, 1083]

[Engstrom et al., ICLR 19]

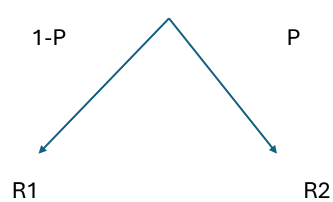
- PPO = full PPO algorithm
- PPO-M = PPO w/o 9 (seemingly secondary) optimizations
- TRPO = original TRPO algorithm
- TRPO+ = TRPO with PPO optimizations
- [,] = 95% confidence interval

Sample Efficiency

- Data reuse:
  - Algorithms like DQN use a “replay buffer” to maximize data efficiency
  - Works because Q-learning is off-policy
- Policy gradient is not inherently off-policy
- Rewards “must” be from the policy you are updating
- Is there a workaround?

Importance weights

- Simple case: Policy goes right w.p. p, left w.p. 1-p
- Adjust p using policy gradient



Re-weighting

- Suppose we have generated 100 samples using different values of p
- Need to keep sampling, or can we re-use previous experiences?
- Suppose sample was generated using policy q
- Replace p with p/q when using this sample
- Intuition: Sample was generated w.p. q, so is implicitly weighted by q, p/q re-weights to effectively sample by p
- TRPO and PPO use reweighting

Issues with importance weights

- When p ≅ q, everything is great
- As p and q get further apart, importance weights (p/q) get weird
- Combining with baseline updates also gets weird since baseline can be from an outdated policy function

Generalized Advantage Estimation

- As presented, baseline is subtracted at every step:

$$\nabla U(\theta) = \mathbb{E}_\tau\left[\sum_{k=1}^{d} \nabla_\theta \log \pi_\theta(a^{(k)} \mid s^{(k)})\gamma^{k-1} A_\theta\left(s^{(k)}, a^{(k)}\right)\right]$$

$$A(s,a) = Q(s,a) - U(s)$$

- What if baseline is stale?
- Interpolate between samples and baseline
  - Using n steps of samples
  - Use baseline at end of n steps

Brief comments about DDPG

- Many control problems *do not inherently require stochastic policies*
- Variance reduction tricks required in policy gradient/actor-critic methods can in some ways be viewed as mending a self-inflicted wound – using a stochastic policy unnecessarily introduces additional variance into the gradient estimate
- But how do we estimate the gradient for an arbitrary policy function?
- Silver et al. (2014) showed decomposition of gradient mirrors stochastic policy gradient if we have a differentiable action function as in, e.g., **a deterministic action function defined over a continuous action space**

Summary

- Large family of modern RL methods that combine aspects of policy gradient and value function approximation: Actor Critic Methods
- Simultaneously learn:
  - Continuous policy function
  - Q/Advantage functions for baseline
  - Sometimes with multiple heads on a single NN
- Originally viewed a best suited to continuous control problems
- Increasingly applied to general RL problems, even Atari