ECE433/COS435 Spring 2024 Introduction to RL

Lecture 19: Advanced Policy Gradient Methods

The policy gradient method seeks to optimize the policy directly. Mathematically, the policy gradient can be represented as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q(s,a)],$$

(assuming undiscounted problems and (s,a) is a state-transition drawn randomly from the sample path). This is a simplified form of policy gradient theorem.

Actor-critic is a policy-based method that involves policy gradient updates to a policy network and value updates to a Q network simultaneously. What we hope to achieve:

- Small variance in actor/critic updates
- Smooth, stable model-free updates
- Convergence guarantees
- Invariance with respect to policy parametrization

1. Advantage Actor-Critic (A2C)

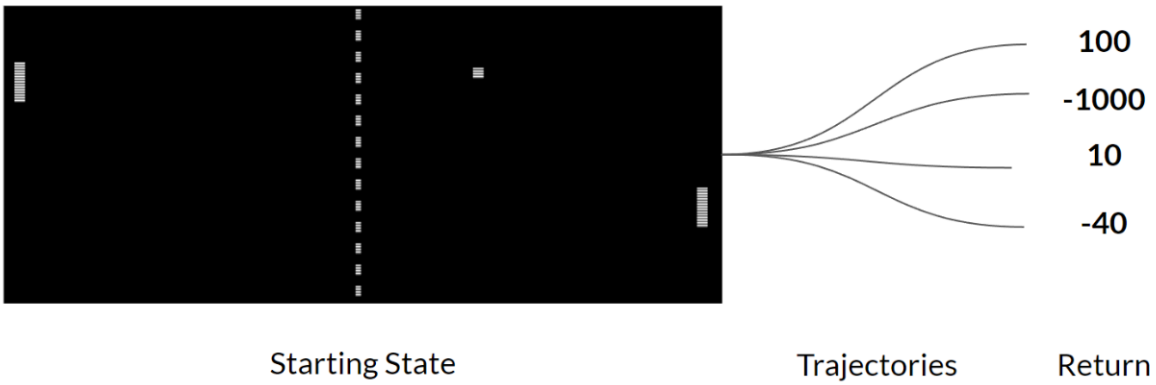
1.1. The Problem of Variance

In policy gradient method, we want to increase the probability of actions in a trajectory proportional to how high the return is.

- If $Q(s,a)$ is high, we will push up the probabilities $\pi(a|s)$.
- Else, if the value is low, it will push down $\pi(a|s)$.

In REINFORCE, this return is calculated using a Monte-Carlo sampling. But variance is very high, since trajectories can lead to different returns due to stochasticity of the environment (random events during episode) and stochasticity of the policy.

In actor-critic, this return is obtained from a critic network which is updated via some form of Q learning. This has somewhat reduced the variance by removing the stochastic of the random trajectory starting from the same (s,a). However, $Q(s,a)$ can still have high variances due to the random nature of (s,a).



1.2. Adding Baselines to PG Estimates

The solution is to mitigate the variance by introducing a **baseline**: The idea is to subtract a term $b(s_t)$ (which does not depend on the action taken) from the reward. The modified policy gradient sample looks like this:

$$\nabla_\theta \log \pi_\theta(a_t \mid s_t)(Q(s_t, a_t) - b(s_t))$$

Why It Remains Unbiased?

Mathematically, this is because the baseline $b(s_t)$ does not affect the gradient's direction since it's subtracted equally from all possible outcomes. To see this we derive

$$\begin{aligned}
E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)[Q^{\pi_\theta}(s,a) - b(s)]] &= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)] - E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) b(s)] \\
&= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)] - \sum_s d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s,a) b(s) \\
&= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)] - \sum_s d^{\pi_\theta}(s) b(s) \nabla_\theta \sum_a \pi_\theta(s,a) \\
&= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)] - \sum_s d^{\pi_\theta}(s) b(s) \nabla_\theta 1 \\
&= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)] - \sum_s d^{\pi_\theta}(s) b(s) * 0 \\
&= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)]
\end{aligned}$$

Choice of baseline

There are many options to choose a baseline function. But the state value function is regarded to be a good baseline function:

$$b(s) = V^{\pi_\theta}(s)$$

In this case, we are essentially computing policy gradients using the **advantage function**:

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

which measures the relative goodness of an action a compared to the current value function. With a good baseline, the variance of the policy gradient estimates can be significantly lowered, which leads to more stable and efficient learning.

1.3. Algorithm

The Advantage Actor-Critic (A2C) algorithm updates the actor using an estimated advantage function, which is viewed as indicating the relative quality of actions.

- **Actor Update:** The actor's policy parameters are updated using the PG estimate:
$$\Delta\theta = \alpha \nabla_\theta \log \pi(a|s;\theta) A(s,a), \tag{1}$$
where α is a learning rate, $A(s,a)$ is the estimated advantage function.
- **Critic Update:** The critic updates its value function parameters ϕ by:
$$\Delta\phi = -\beta \nabla_\phi (R - V(s;\phi))^2, \tag{2}$$
where β is the critic's learning rate, R is the observed return, and $V(s;\phi)$ is the estimated value function.
- **Advantage Function:** The advantage function is approximated using the **TD error**:
$$A(s,a) = R + \gamma V(s';\phi) - V(s;\phi), \tag{3}$$
where s' is the next state and γ is the discount factor.

A2C updates the policy according to the extra return/loss beyond the expected value of that state.

- If $A(s,a) > 0$: our actor update is pushed in the direction to increase $\pi(a|s)$.
- If $A(s,a) < 0$, the update is pushed in the opposite direction.

2. Convergence Theory of PG Method

Policy optimization: Parameter space vs distribution space

When we think about policy optimization, we usually mean optimizing the return in the policy parameter space for θ:

$$\max_{\theta} J(\theta)$$

What we are really interested in is optimizing in distribution space for λ^{π}. Let λ^{π} be the cumulative state-action distribution of a policy π:

$$\lambda^{\pi}_{sa} := \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{P}(s_t = s, a_t = a \mid \pi, s_0 \sim \xi).$$

The distribution optimization problem is actually a large-scale linear program:

$$\max_{\theta} J(\theta) = \sum_{s,a} \lambda^{\pi_\theta}_{sa} \cdot r(s,a)$$
$$\text{subject to } \lambda^{\pi} \text{ is a valid distribution}$$

Note that optimization in θ is equivalent to optimization in λ^{π_θ}, as long as the policy network is sufficiently expressive to parametrize all possible policies.

Convergence theory of policy gradient method

Consider the policy gradient method of the form $\theta \leftarrow \theta + \alpha \nabla J(\theta)$ for maximizing $J(\theta)$. This optimization problem is highly nonconvex and involves complexity neural networks.
[1,2] showed that policy gradient converges to the optimal policy in the cases of no parametrization and softmax parametrization, respectively.

In [3], we showed that policy gradient method converges globally to the optimal policy, regardless of the policy network architecture (as long as its sufficiently expressive). This analysis exploits the hidden convexity phenomenon: gradient updates in θ can be mapped to gradient flow in λ for a linear optimization problem.

3. Natural Policy Gradient

Gradient redefined

Steepest ascent direction of a function $h(\theta)$ is $\nabla h(\theta)$:

- It yields the most reduction in h per unit of change in θ (parameter space)
- Change is measured using the standard Euclidean norm $\|\cdot\|$

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\min_{d:\|d\| \leq \epsilon} \{h(\theta + d)\}$$

However, if we know θ belongs to some manifold, steepest descent defined using the Euclidean norm might no longer work well.

- Can we use an alternative definition of (local) distance?
- As suggested by [Amari, 1998] it is better to define a metric based not on the choice of the coordinates but rather on the manifold these coordinates parametrize! For example, the geodesic distance of that manifold.

Let's try to find the ``natural policy gradient"

Policy optimization is equal to optimizing the distribution. Consider the optimizing a general objective function $h(\theta)$. Let's choose KL divergence as the "metric", and we redefine the gradient by considering the KL divergence instead of Euclidean ball, for $\epsilon \approx 0$,

$$\begin{aligned}\widetilde{\nabla} h(\theta) = \arg\max_{\Delta\theta} & h(\theta + \Delta\theta) \\ \text{s.t. } & D_{KL}(p(x \mid \theta) \| p(x \mid \theta + \Delta\theta)) \leq \epsilon\end{aligned}$$

To approximate the objective, we use Taylor's series to expand the objective and the expected KL-divergency constraint up to the second-order.

- $h(\theta + \Delta\theta) \approx h(\theta) + \Delta\theta^T \nabla h(\theta)$
- The Kullback Leibler divergence can be approximated by the Fisher information matrix (2nd order Taylor approximation)

$$D_{KL}(p(x \mid \theta) \| p(x \mid \theta + \Delta\theta)) = \Delta\theta^T F(\theta) \Delta\theta + O\left(\Delta\theta^3\right)$$

where $F(\theta)$ is the Fisher Information Matrix (FIM)

$$F(\theta) = \mathop{\mathbb{E}}_{x \sim p(\cdot \mid \theta)} \left[\nabla \log p(x \mid \theta) \nabla \log p(x \mid \theta)^T \right]$$

The Fisher information matrix captures information on how a parameter influences the distribution.

By using the Taylor expansion argument, the redefined gradient, namely the **natural gradient** solves the following quadratic optimization problem.

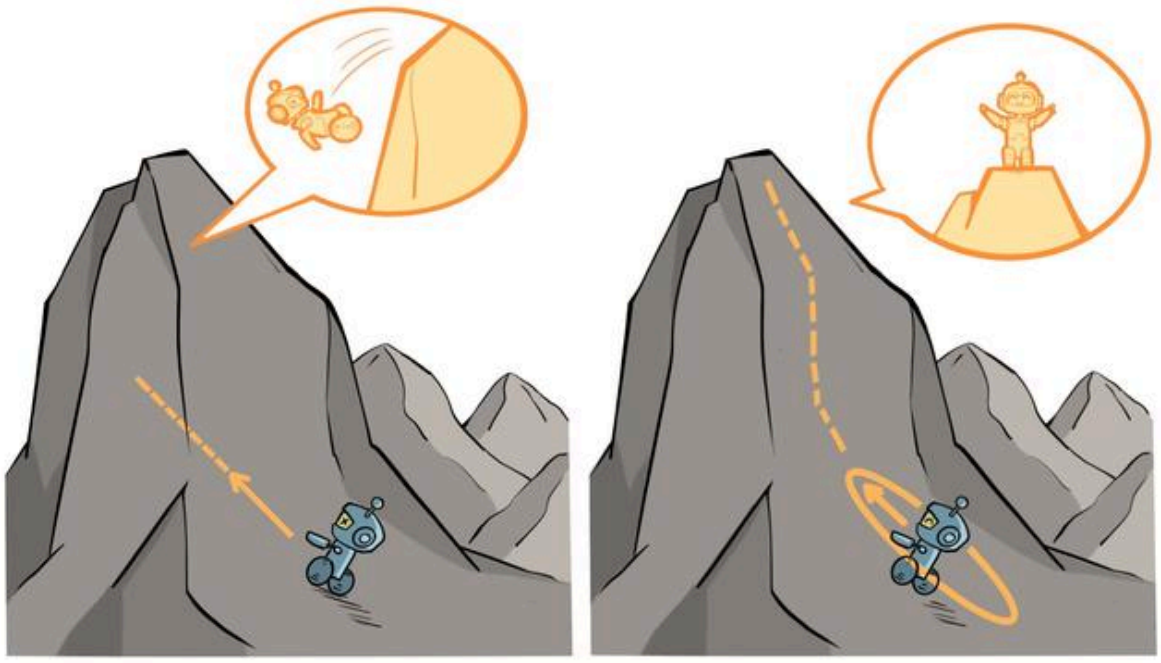
$$\tilde{\nabla} h(\theta) = \arg\max_{\Delta\Theta} \Delta\theta^T \nabla h(\theta) \qquad \text{s.t. } \Delta\theta^T F(\theta) \Delta\Theta \leq \epsilon$$

Solving the preceding quadratic optimization problem, we get the natural policy gradient, given as

$$\tilde{\nabla} h(\theta) \propto F^{-1} \nabla h(\theta)$$

The natural policy gradient can be viewed as a conditioned version of the vanilla policy gradient, which captures both the steepest ascent direction and the local curvature of the distribution space.

4. Trust Region Policy Optimization (TRPO)



Even with the natural policy gradient, choosing stepsize for policy optimization can be tricky:

- Stepsize too large → terrible policy → terrible data
- Need to update policies in a smooth and stable way, but not clear how to backtrack if stepsize gets too far

TRPO take advantages of the constraint formulation and optimizes the policy within a trust region to ensure each update improves the policy with high confidence.

In particular, TRPO uses a surrogate loss and solves the KL-constrained optimization problem for each policy update:

$$\text{surrogate loss} \quad \max_{\theta} \mathbb{E}_{\theta_{old}}\left[\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_{old})}A(s,a)\right], \tag{4}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_{\theta_{old}}[KL[\pi(\cdot|s;\theta_{old})||\pi(\cdot|s;\theta)]] \leq \delta, \tag{5}$$

where A is the current estimated advantage, δ is a small positive constant, $\mathbb{E}_{\theta_{old}}$ denotes taking average of recently generated transition samples.

- **Surrogate loss:** Recall that the standard loss for actor update can be reformulated into another expectation

$$\nabla \mathbb{E}_{\theta}[\pi(a|s;\theta)A] = \mathbb{E}_{\theta_{old}}\left[\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_{old})}A\right],$$

where we used a *change of probability trick*.
When $\theta = \theta_{old}$, the gradient of surrogate loss is an unbiased estimator of the true policy gradient:

$$\nabla \mathbb{E}_{\theta_{old}}\left[\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_{old})}A\right] = \mathbb{E}_{\theta_{old}}\left[\frac{\nabla\pi(a|s;\theta)}{\pi(a|s;\theta_{old})}A\right] = \mathbb{E}_{\theta}\left[\frac{\nabla\pi(a|s;\theta)}{\pi(a|s;\theta)}A\right] = \nabla J(\theta)$$

where the second equality uses $\theta = \theta_{old}$.

The surrogate loss allows us to reuse data generated by recent policies θ_{old} for better sample efficiency. When $\theta \neq \theta_{old}$, we hope $\theta \approx \theta_{old}$ so that the surrogate loss would generate a good approximation to the policy gradient.

- **KL constraints:** We approximate the KL divergence using its second-order Taylor expansion, similar to analysis in the previous section. We also approximate the surrogate loss using its first-order Taylor expansion. Let $\theta_k = \theta_{old}$. The optimization problem simplifies to

$$\begin{aligned}\theta_{k+1} = \arg\max_{\theta} & g^T(\theta - \theta_k) \\ \text{s.t. } & \frac{1}{2}(\theta - \theta_k)^T F(\theta - \theta_k) \leq \delta\end{aligned}$$

where the Fisher information matrix is

$$F = \mathbb{E}_{\pi_{\theta_k}}\left[\nabla_\theta \log \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s)^T\right]$$

and it can be estimated using Monte Carlo sampling.

This objective can be solved analytically:

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

To implement TRPO in practice, we need to estimate F and take matrix inversion to compute the natural policy gradient estimate.

- Efficiency: Computing F exactly can be computationally intensive. Various approximation methods, such as using conjugate gradient algorithm to compute the step direction without explicitly forming F, are often used in practice.
- Sampling Variance: The approximation of F can be noisy due to the finite sample size. Techniques like adding a damping coefficient can help stabilize the updates.

Pros and Cons:

+ Leverages the geometry of distribution space
+ Not sensitive to policy network parametrization because the KL divergence is invariance with respect to the parametrization
+ Adaptive stepsize to make sure smooth, stable updates in a local confidence region
− Constrained optimization is generally harder than unconstrained optimization
− Matrix inversion for natural gradient implementation is complex
− Would be good to harness good first-order optimizers for deep neural nets, but TRPO do not take advantages of Adam, RMSProp, etc.

5. Proximal Policy Optimization (PPO)

To further scale up TRPO, we want to get rid of matrix inversion in TRPO but still keep the new policy proximal to the original one, to improve the stability and efficiency of policy updates.

PPO achieves this goal by restricting updates to a small region around the current policy. PPO uses a (heuristic) clipped objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_{\theta_{old}}[\min(r(\theta)A, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A)], \tag{6}$$

where $r(\theta)$ is the probability ratio $\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a_t|s_t)}$, indicating how the new policy π_θ deviates from the old policy $\pi_{\theta_{old}}$ in terms of action probabilities. Here, A is an estimator of the advantage at the current (s, a), and ϵ is a small value that controls the extent to which the policy is allowed to change (often set around 0.1 to 0.2).

The clip function ensures that the ratio stays within the range $[1-\epsilon, 1+\epsilon]$, limiting the update step's impact. This clipped objective will become more **pessimistic** when θ gets far away from θ_{old}, so that it keeps the new θ close to θ_{old}.

PPO is easy to implement and relatively simple to tune. It turns out to be a most popular policy-gradient-based algorithm for Deep RL.

Summary

Policy gradient methods are an and powerful class of RL methods. Yes we make actor critic more efficient using a combination of technologies:

- Reduce Variance: Advantage estimation and A2C
- Leverage the Geometry: Nature Policy Gradient and TRPO
- Scale Up Further: Proximal Policy Optimization
- Theory: Global convergence in the distribution space

[1] On the theory of policy gradient methods: Optimality, approximation, and distribution shift A Agarwal, SM Kakade, JD Lee, G Mahajan
Journal of Machine Learning Research, 2021 (Unparameterized polic)

[2] On the global convergence rates of softmax policy gradient methods
J Mei, C Xiao, C Szepesvari, D Schuurmans
International conference on machine learning, 2020 proceedings.mlr.press (Softmax policy)

[3] Variational policy gradient method for reinforcement learning with general utilities
J Zhang, A Koppel, AS Bedi, C Szepesvari, M Wang
Advances in Neural Information Processing Systems 2020 (General policy networks)