Midterm + HW4 Recitation

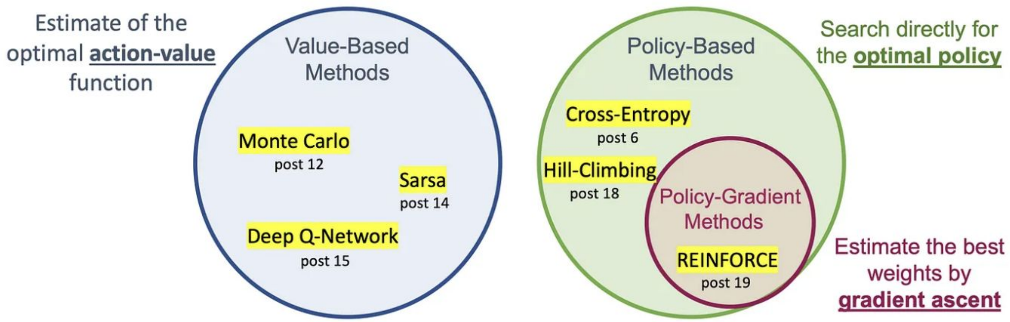
Nidhi Hegde, Lawrence Jang

Table of Contents

Topics to Know for Midterm

- Everything covered till the lecture on Sep 29th is fair game
- Value-Based Methods (Recitation 3)
- Policy Gradients and Actor Critic Methods (Recitation 2)
- We will cover Advanced PG today

Policy-Gradient Methods



- Value-based methods: learn a value function (an optimal value function leads to an optimal policy)
  - Goal: minimize the loss between the predicted and target value
  - Policy is implicit as it is generated directly from the value function (e.g. eps-greedy from Q-function)
  - Examples: DQN, SARSA
- Policy-based methods: learn to approximate optimal policy directly (without learning a value function)
  - Parameterize the policy, e.g. using a neural network
  - Policy outputs a probability distribution over actions (stochastic policy)
  - Goal: maximize the performance of the parameterized policy using gradient ascent

REINFORCE (Monte-Carlo PG) - Algorithm

REINFORCE, or Monte Carlo policy-gradient, uses an estimated return from an entire episode to update the policy parameter θ.

0. Initialize policy parameters θ

1. Sample trajectories $\{\tau_i = \{s_t^i, a_t^i\}_{t=0}^{T}\}$ by deploying the current policy $\pi_\theta(a_t \mid s_t)$.

2. Compute gradient vector $\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)}) G_t^{(i)}$

3. $\theta \leftarrow \theta + \alpha \nabla_\theta U(\theta)$

Approximate the gradient with the empirical estimate from N sampled trajectories

Gradient Estimator expected to be unbiased (~N samples) and have low variance

What does the term Σ_t $\log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})$ resemble? Think in terms of MLE: “log-likelihood of actions under the policy“

Multiplying by G reweights the likelihood (“reward-weighted maximum likelihood”)

REINFORCE - Baseline: Algorithm

Initialize policy parameter θ, baseline b

for iteration$=1, 2, \cdots$ **do**

Collect a set of trajectories by executing the current policy

At each timestep t in each trajectory τ^i, compute

Return $G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$, and

Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t ||b(s_t) - G_t^i||^2$,

Update the policy, using a policy gradient estimate $\hat{g}$,

Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.

Expectation of the baseline term is 0, so the estimator is still unbiased

Advantage Actor Critic: Algorithm

Train a value network to estimate the baseline

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize S (first state of episode)
    $I \leftarrow 1$
    Loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Policy-based methods, pros and cons

Pros

- We can estimate the policy directly without storing additional data
- Policy-gradient methods can learn a stochastic policy
    - We don’t need to implement an exploration/exploitation trade-off by hand
- More effective in high-dimensional action spaces and continuous action spaces
- Better convergence properties

Cons

- Converges to a local maximum sometimes
- Slower, step-by-step: it can take longer to train (inefficient)
- Gradient estimate is very noisy: there is a possibility that the collected trajectory may not be representative of the policy

Big Picture Table

Method	On/Off Policy?	Bootstraps?
Monte Carlo Methods	On*	N
SARSA	On*	Y
Expected SARSA	Either	Y
Q-Learning	Off	Y
REINFORCE	On*	N
Actor Critic, A2C	On*	Y

* can be made off policy with importance sampling

Value Based vs Policy Gradients

Value-based methods (e.g. Q-learning) struggle with:

- Large/continuous action spaces
- Instability in function approximation

Policy Gradients offer:

- Direct optimization of the policy.
- Can learn stochastic Policies
- Naturally handle continuous actions (think how?)

Problems:

- High-variance (e.g. REINFORCE)
- Sample Efficiency (on-policy constraints)

Advanced Policy Gradients

Policy Difference Lemma

$$J(\pi) - \underbrace{J(\mu)}_{\text{data collection policy}} = \mathbb{E}_{\tau \sim P_\pi(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t \underbrace{A^\mu(s_t, a_t)}_{\text{estimated via a critic}} \right]$$

- 2 policies: π and μ
- J(.) -> expected return from a policy
- A(s,a) -> Advantage under μ
(also learned via critic network)

$$J(\pi) - J(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^\pi, a \sim \pi} [A^\mu(s, a)].$$

Distribution over Trajectories => State Visitation frequencies

Approximation: assuming state distribution under the 2 policies are similar (can ignore the importance weight)

Generic optimization problem:

$$\max_{\pi} \quad \mathbb{E}_{s\sim d^{\mu}(s)}\left[\mathbb{E}_{a\sim \pi(\cdot|s)}\left[A^{\mu}(s,a)\right]\right]$$

$$- \alpha \, D(\pi, \mu)$$

what is this precisely?

TV distance is upper bounded by sqrt(KL Divergence/2) (Pinsker's Inequality)

KL is not symmetric

KL (π || μ) -> AWR

KL (μ || π) -> TRPO, PPO

Advantage Weighted Regression

Actions with higher advantage get exponentially more probability mass. β is like a temperature controlling how aggressive the update is

$$\pi(a|s) \quad \propto \quad \mu(a|s) \, \exp\left(\frac{1}{\beta} A^{\mu}(s,a)\right).$$

Think: is AWR off-policy or on-policy?

- **Input:** replay buffer of transitions (s, a, r, s').
- **Step 1:** Estimate value function $V^{\pi}(s)$ (e.g., via TD learning or fitted V).
- **Step 2:** Compute advantage estimates:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s).$$

- **Step 3:** Weight each action by $w(s,a) = \exp(\frac{1}{\beta} A^{\pi}(s,a))$.
- **Step 4:** Update the policy network π_θ with weighted log-likelihood loss.

Proximal Policy Optimization

In PG methods, too large updates -> instability

Idea: prevent the policy from changing too much without doing explicitly KL-constrained optimization but by clipping

$$\mathbb{E}_{s\sim d_\pi, a\sim \pi}[A^\mu(s,a)] = \mathbb{E}_{s\sim d_\mu, a\sim \mu}\left[\frac{d_\pi(s)}{d_\mu(s)}\frac{\pi(a|s)}{\mu(a|s)}A^\mu(s,a)\right].$$

PPO Clipping Objective

1. Collect trajectories using current policy $\mu = \pi_{\text{old}}$.
2. Estimate advantages $A^\mu(s, a)$ (e.g., GAE).
3. Optimize clipped surrogate objective $L^{\text{clip}}(\pi)$.
4. Update policy, set $\mu \leftarrow \pi$.
5. Repeat.

$$L^{\text{clip}}(\pi) = \mathbb{E}_{s,a\sim\mu}\Big[\min\big(r(s,a)A^\mu(s,a),\ \text{clip}(r(s,a), 1-\epsilon, 1+\epsilon)A^\mu(s,a)\big)\Big].$$

$$r = \frac{\pi(a|s)}{\mu(a|s)}$$

1. A > 0, r < 1 - ϵ -> good action, but new policy is undersampling it
2. A > 0, r > 1 + ϵ -> good action, new policy already favoring it (clip)
3. A < 0, r < 1 - ϵ -> bad action, new policy already avoids it (clip)
4. A < 0, r >1 + ϵ -> bad action, new policy favors it, so bring it down

Idea of PPO: allow “corrective updates”, but don’t correct aggressively!

Exercises

Q1: Given a discrete action space and a feature vector $\phi(s,a)$ for each state-action pair, how would you parameterize a policy for a policy gradient algorithm?

Q2: If training a robot arm with continuous actions -- DQN, REINFORCE, or PPO? Why?

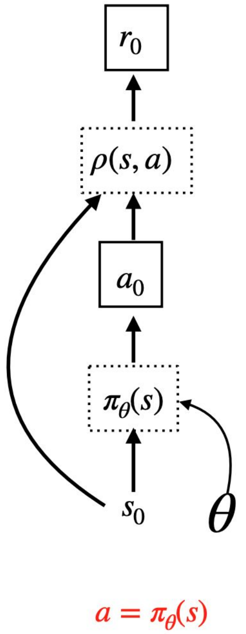
DDPG (Deep Q-Learning + Policy Gradients) - 2015

Off-policy AC method, designed for continuous action spaces.

Learns deterministic policy

Exploration: by adding noise to actions

$$\mathbb{E} \sum_{t} \frac{dQ(s_t, a_t)}{d\theta} = \mathbb{E} \sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t} \frac{da_t}{d\theta}$$



DDPG algorithm

Target Values: Soft updates to both actor and critic networks

Critic update (Bellman error):

$$L(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\big(Q_\phi(s,a) - y\big)^2\Big]$$

where

$$y = r + \gamma Q_{\phi'}\big(s', \pi_{\theta'}(s')\big).$$

Actor update (Deterministic Policy Gradient):

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\mathcal{D}}\Big[\nabla_\theta \pi_\theta(s)\, \nabla_a Q_\phi(s,a)\big|_{a=\pi_\theta(s)}\Big].$$

Target networks:

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi', \quad \theta' \leftarrow \tau\theta + (1-\tau)\theta'.$$

Problem: unstable, overestimates Q, exploration hard

TD3- Twin Delayed Deep Deterministic Policy Gradient (2018)

Aims to fix training instability and overestimation bias in DDPG.

Idea:

1. Train 2 critic networks, and define the target to use the minimum of the two
2. Delayed Policy Update- let the critic stabilize before the actor chases (update every d steps)
3. Smoothen critic targets by adding clipped noise

Soft Actor-Critic (SAC) - 2018

Stochastic Policy: Gaussian (natural exploration)

$$J(\pi) = \mathbb{E}\Big[\sum_t r(s_t, a_t) + \alpha\,\mathcal{H}(\pi(\cdot|s_t))\Big]$$

Adds Entropy Regularization – Max Entropy RL
(α is the temperature parameter that determines the relative importance of the entropy term versus the reward, controlling the stochasticity of the optimal policy)

Comparison

Algorithm	On/Off Policy	Regularizer	Exploration	Stability	Use Case
AWR	Off-policy	KL (reverse, exponential reweight)	Low	High	Imitation, offline RL
PPO	On-policy	KL (clipping)	Medium	High	Robust baseline
TRPO	On-policy	KL (constraint)	Medium	Very High	Theory-focused
SAC	Off-policy	Entropy + Q	High	High	Continuous control
DDPG	Off-policy	None	Low	Low	Baseline, simple tasks