ECE433/COS435 Reinforcement Learning
Assignment 2: Policy Gradient Methods
Spring 2026

Fill me in

Your name here.

March 16, 2026

Collaborators

Fill me in

Please fill in the names and NetIDs of your collaborators in this section. Make sure you each submit separately, but feel free to work together and submit the same answers as long as you put your names together here.

Instructions

Writeups should be typesetted in Latex and submitted as PDF. **Please submit the asked-for snippet and answer in the solutions box as part of your writeup. Attach code as a .zip file *with the exact structure as we've distributed the zip*, we will run the code against standardized unit tests.**

Grading. The problem set will be graded out of 50 points, and the coding assignment will also be graded out of 50 points, making the total score 100 points.

Question 0 (0 points). Feedback

How many hours did you spend on this homework? Please fill in the solution after you've done all the questions.

Solution

Your solution here...

Problem Set (50 points)

Question 1: Deriving the Policy Gradient (10 points)

The REINFORCE algorithm relies on the *policy gradient theorem*. Starting from the objective

$$J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_\theta}\big[R(\tau)\big]$$

where $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$ is the discounted return of trajectory $\tau = (s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1})$.

(a) **(7 points)** Derive the REINFORCE gradient estimator, explain where the log-derivative trick is used, and show that only future steps surive the derivation.

Solution

Your solution here...

(b) **(3 points)** Why can we not simply compute $\nabla_\theta J(\theta)$ by back-propagating through the environment dynamics? What assumption about the environment does REINFORCE avoid?

Solution

Your solution here...

Question 2: Baselines and Variance Reduction (15 points)

A central challenge with REINFORCE is **high variance**. Subtracting a *baseline* $b(s_t)$ from the returns is one of the most important variance-reduction techniques in policy gradient methods.

(a) **(5 points)** Prove that subtracting a state-dependent baseline $b(s_t)$ does *not* change the expected policy gradient.

Solution

Your solution here...

(b) **(5 points)** Consider a simple environment where the agent always receives *positive* rewards between 50 and 100 (e.g., different routes through a maze, all of which reach the goal but with different path lengths).

- Without a baseline ($b = 0$), the REINFORCE update $\theta \leftarrow \theta + \alpha\, \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t$ will *increase* the probability of *every* action taken (since $G_t > 0$ always). Explain why this is inefficient and can slow down training.
- Now suppose we subtract $b = 75$ (the average return). Some advantages $G_t - b$ are now positive (better-than-average trajectories) and some are negative (worse-than-average). Explain how this makes the gradient signal more informative for learning which actions are actually good.

Solution

Your solution here...

(c) **(5 points)** In practice, the most common baseline is the *value function* $V^\pi(s)$, giving rise to the *advantage* $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$.

- Explain intuitively what the advantage $A^\pi(s,a)$ measures and why it is more useful than the raw return G_t for deciding which actions to reinforce.
- In your coding experiments (Part 1), you will compare REINFORCE with and without a mean-return baseline. Predict what you expect to observe and explain your reasoning.

Solution

Your solution here...

Question 3: From REINFORCE to PPO (20 points)

REINFORCE is conceptually simple but has several practical limitations. This question walks you through the key ideas behind *Proximal Policy Optimization* (PPO).

(a) **(4 points)** REINFORCE uses full Monte Carlo returns G_t to weight the policy gradient. Since G_t depends on the entire sequence of future rewards (and hence future stochastic actions and transitions), its variance grows with episode length. Explain briefly why this makes REINFORCE impractical for long-horizon tasks.

Solution

Your solution here...

(b) **(4 points)** Generalized Advantage Estimation (GAE) defines

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{\ell=0}^{T-t-1} (\gamma\lambda)^\ell \, \delta_{t+\ell}, \qquad \delta_t = r_t + \gamma\, V(s_{t+1}) - V(s_t).$$

Show that:

- When $\lambda = 0$: $\hat{A}_t = \delta_t$ (one-step TD advantage, low variance but biased).
- When $\lambda = 1$: $\hat{A}_t = G_t - V(s_t)$ (Monte Carlo advantage, unbiased but high variance).

Explain in one sentence what λ controls.

Solution

Your solution here...

(c) **(7 points)** PPO optimizes the *clipped surrogate objective*:

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \mathrm{clip}\big(r_t(\theta),\,1{-}\epsilon,\,1{+}\epsilon\big)\hat{A}_t\Big)\Big], \quad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}.$$

Let $\epsilon = 0.2$. Work through the following two scenarios:

Scenario 1: $\hat{A}_t = +3.0$ and $r_t(\theta) = 1.5$ (the new policy is 50% more likely to take this action).

- Compute $\text{surr}_1 = r_t \cdot \hat{A}_t$.
- Compute $\text{surr}_2 = \text{clip}(r_t, 0.8, 1.2) \cdot \hat{A}_t$.
- What is L^{CLIP} for this sample? Is the policy encouraged to keep increasing r_t beyond 1.2?

Scenario 2: $\hat{A}_t = -2.0$ and $r_t(\theta) = 0.6$ (the new policy has reduced the probability of this bad action).

- Compute surr_1 and surr_2.
- What is L^{CLIP} for this sample? Is the policy encouraged to reduce r_t further below 0.8?

Using these examples, explain in 2–3 sentences why the clipping mechanism acts as a "trust region" that prevents destructively large policy updates.

Solution

Your solution here...

(d) **(5 points)** REINFORCE discards data after a single gradient step. PPO reuses each batch for multiple gradient steps (`num_epochs` > 1).

- Why would it be invalid to do multiple gradient steps with vanilla REINFORCE?
- What mechanism in PPO makes data reuse safe?
- What would you expect to happen if `num_epochs` were set very large (e.g., 100)?

Solution

Your solution here...

Coding Assignment (50 points)

See the coding portion `hw2.py`. Implement every function marked `TODO` and get rewards increasing for REINFORCE and PPO. REINFORCE should be able to get close to solving them but it'll be noisy. Don't worry about this as much as getting PPO. PPO should reliably solve both environments.

Part 1: REINFORCE on LunarLander-v3 (15 points)

Implement the missing components in hw2.py needed to get the script running for the LunarLander environment with REINFORCE.

Part 2: PPO on LunarLander-v3 (15 points)

Implement the missing components in hw2.py needed to get the script running for the LunarLander environment with PPO, including `ValueNetwork`, `collect_batch` (with GAE advantage computation), `ppo_update`, `train_ppo`.

Note: Your `collect_batch` implementation should handle both discrete and continuous action spaces (see Part 3). Detect the action space type using `isinstance(env.action_space, gym.spaces.Discrete)`.

Part 3: Continuous Control — PPO/REINFORCE on Pendulum-v1 (5 points)

So far, both environments have used *discrete* actions. Many real-world control problems require *continuous* actions (e.g., torques, velocities, steering angles).

Implement `GaussianPolicy`, which outputs a Gaussian (Normal) distribution over continuous actions:

- The network outputs the *mean* of the action distribution.
- A learnable `log_std` parameter (one per action dimension) controls the standard deviation.
- Actions are sampled from $\mathcal{N}(\mu, \sigma^2)$ and clamped to the environment's valid range.

This is necessary for getting the Pendulum-v1 environment working.

Part 4: Discussion and Plots (15 points)

Now run hw2.py and try to make sure that PPO and REINFORCE are learning by looking at the output plots. Don't worry if REINFORCE is not performing well. In particular, look at the REINFORCE versus PPO plots, what do you see?

Solution

Embed all the created plots here. Discuss why PPO is more stable than REINFORCE. What was most annoying to get working and why?

Now tune the hyperparameters for PPO to get the best performance. Report the hyperparameters you used and the results you achieved. Include the relevant plots you created for different hyperparameters. What was most effective and why? HINT: try increasing the number of epochs and learning rate for PPO. Don't worry so much about REINFORCE, it is possible you won't be able to get it to work beyond some minimal learning.

Solution

Embed all the created plots here. Discuss why PPO is more stable than REINFORCE. What was most annoying to get working and why?