Problem Set

Due Friday, Aug 1st, 2025

Turn in your assignment by 23:59 (midnight) this Friday, Aug 1st, 2025. Your submission should be typed, preferably in latex or markdown.

Emails: KartikNagpal@berkeley.edu JohnViljoen@berkeley.edu

The goal of this assignment is to get familiarity with single and multi-agent reinforcement learning (RL) and its application to transportation settings. You will be loading and training on a mujoco ant environment. All starter code can be found in the class Google drive folder

https://drive.google.com/drive/folders/15IDBS-DW4ZWyL0hGJMjucLbXkK4QcfJg

You have the option of running the code either on Google Colab or on your own machine. Note: The Colab is only used as a source of GPU compute, so you will be editing the same code regardless of what option you choose. If you are running locally we strongly recommend you use Conda to manage your Python environment and dependencies. Here are instructions for installing Conda and an environment setup guide.

1 Generalized Advantage Estimation

Consider the reinforcement learning problem where we are training an agent policy π, and you have access to two learned functions:

- $V^\pi(s)$: this **Value Function** estimates the expected return under policy π if start at state s,
- $Q^\pi(s,a)$: the estimated value of taking action a in state s, and then following policy π.

Now consider the quantity:

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

A. What does this difference represent, intuitively? Why might $A^{\pi_\theta}(s,a)$ be called the "Advantage Function"?

B. Suppose you sample from a policy π_θ, and find that $A^\pi(s,a) \geq 0$. What does this tell you about the action a relative to your current policy? What about when $A^\pi(s,a) < 0$?

Proximal Policy Optimization (PPO) uses a Generalized Advantage Estimation (GAE) estimate of our policy's advantage function. To compute this advantage estimate for a given (s_t, a_t) at timestep t we construct estimators $\hat{A}_t^{(k)} = \hat{A}^{(k)}(s_t, a_t)$:

$$\begin{aligned}
\hat{A}_t^{\pi,(1)} &= -V^\pi(s_t) + r_t + \gamma V^\pi(s_{t+1}), \\
\hat{A}_t^{\pi,(2)} &= -V^\pi(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V^\pi(s_{t+2}), \\
\hat{A}_t^{\pi,(3)} &= -V^\pi(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V^\pi(s_{t+3}), \\
\hat{A}_t^{\pi,(k)} &:= -V^\pi(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V^\pi(s_{t+k})
\end{aligned}$$

and we take the weighted average of these estimators

$$\hat{A}_t^\pi = A_t^{G\hat{A}E} = \frac{\sum_k w_k \hat{A}_t^{(k)}}{\sum_k w_k}$$

This Generalized Advantage Estimation (GAE) is the basis for our PPO loss! For further simplification, we choose a hyper-parameter λ and set $w_k = \lambda^{k-1}$, which greatly simplifies this weighted average to

$$\hat{A}_t^\pi = \delta_t + \gamma\lambda\hat{A}_{t+1}^\pi \tag{1}$$

$$\text{where } \delta_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) \tag{2}$$

C. How does the bias and variance of the $\hat{A}_t^{\pi,(1)}$ estimate compare with the bias and variance of the $\hat{A}_t^{\pi,(\infty)}$ estimate?

D. Implement the formula for the GAE calulation seen in Eqn. 1 as python code inside the

```
_get_advantages(gae_and_next_value, transition)
```

function found in our starter code. Please attach a copy of your function implementation to your submission.

2 Calculating Actor Loss

As seen during lecture, we can use our GAE estimate of the advantage function to define the actor loss for the our policy π_θ as

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a),\ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right),$$

$$\text{where } g(\epsilon, A) = \begin{cases} (1+\epsilon)A & \text{if } A \geq 0 \\ (1-\epsilon)A & \text{if } A < 0 \end{cases}$$

Note that computationally it is difficult for us to represent policies directly, and so instead compute log probabilities (i.e. $\log(\pi_\theta(a|s))$), and so we use a simple exponentiation trick to rewrite our loss as

$$\begin{aligned} L(s, a, \theta_k, \theta) &= \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a),\ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right), \\ &= \min\left(\exp\left(\log\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\right)\right) A^{\pi_{\theta_k}}(s, a),\ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right), \\ &= \min\left(\exp\left(\log(\pi_\theta(a \mid s)) - \log(\pi_{\theta_k}(a \mid s))\right) A^{\pi_{\theta_k}}(s, a),\ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right). \end{aligned} \tag{3}$$

A. Note that it is common to normalize our GAE estimates before performing this loss calculation via

$$A_t^\pi = \frac{\hat{A}_t^\pi - \text{mean}(\hat{A}_t^\pi)}{\text{std\_dev}(\hat{A}_t^\pi)}.$$

Why would this help our loss calculations?

B. Implement the actor loss shown in Eqn. 3 within the

```
_loss_actor(log_prob, log_prob_k, normalized_gae, clip_epsilon)
```

function in our starter code. Please attach a copy of your function implementation to your submission.

3 Training

Now that you have implemented GAE calulation and actor loss, let us try training our PPO and IPPO implementations! Much of the environment setup has been done automatically for you in the starter code.

A. Now train your IPPO set of policies! **Remember to download the `data` folder after training!**

B. Use tensorboard to generate plots of your results and attach them below.

C. Discuss your results briefly.

4 Next Steps...

You have completed the assignment! Everything past this point is just for fun! You can use the

```
generate_rollout(env, policy, rng, jit=True, num_timesteps=100)
```

function to generate rollout html files under the `data/rollouts/` folder which can be downloaded and launched using any internet browser or html-viewer. These visualizations will show you what your policy has learned to do!

Furthermore, you can now mess with your `config` dictionary in the "Perform IPPO Training" section to identify what changing these hyper-parameters does to your training.