[SOLUTION TEMPLATE] Assignment 2: Policy Gradients

Due September 25, 11:59 pm

4 Policy Gradients

- Create two graphs:
  - In the first graph, compare the learning curves (average return vs. number of environment steps) for the experiments prefixed with `cartpole`. (The small batch experiments.)

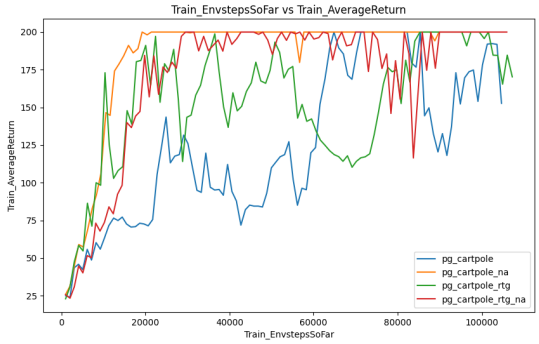


Figure 1: Small batch size (1000) exp

  - In the second graph, compare the learning curves for the experiments prefixed with `cartpole_lb`. (The large batch experiments.)

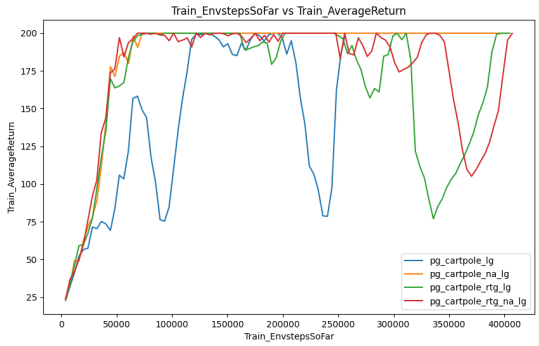


Figure 2: Large batch size (4000) exp

For all plots in this assignment, the x-axis should be number of environment steps, logged as `Train_EnvstepsSoFar` (*not* number of policy gradient iterations).

- Answer the following questions briefly:
  - Which value estimator has better performance without advantage normalization: the trajectory-centric one, or the one using reward-to-go?

    For both small and large batch sizes, reward-to-go works much better and converges faster, trajectory-centric suffers from large variance even training more steps.

  - Did advantage normalization help?

Yes, advantage normalization helps a lot. With advantage normalization, trajectory-centric converges much faster and even surpasses reward-to-go. The same effect can also be observed for reward-to-go. It makes both methods with trajectory-centric and reward-to-go more training stable.

– Did the batch size make an impact?

Yes, it makes an impact on the figures. With larger batch size, training converges faster with lower variance. The impact is more clear with advantage normalization and reward-to-go, they barely shift from their optimal action distribution.

- Provide the exact command line configurations (or `#@params` settings in Colab) you used to run your experiments, including any parameters changed from their defaults.

5 Neural Network Baseline

- Plot a learning curve for the baseline loss.

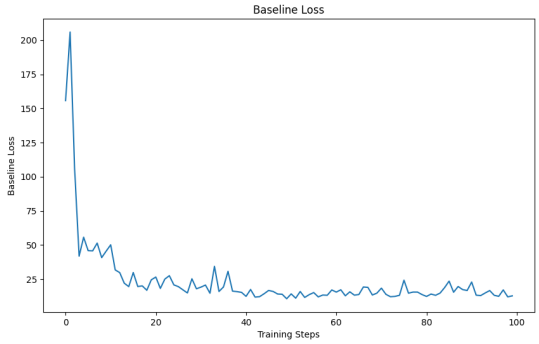


Figure 3: Learning curve for the baseline loss.

- Plot a learning curve for the eval return. You should expect to achieve an average return over 300 for the baselined version.

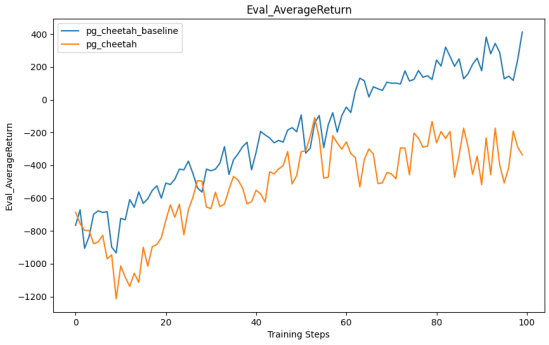


Figure 4: Average eval return for both baseline and no-baseline version

- Run another experiment with a decreased number of baseline gradient steps (`-bgs`) and/or baseline learning rate (`-blr`). How does this affect (a) the baseline learning curve and (b) the performance of the policy?

  We try to modify blr and bgs, more specifically, we decrease blr by half and increase by half, decrease to 3 and 1. As shown in figure 5 and 6. Decrease or increase blr seems to cause less impact on average eval return, it indeed affects the preliminary learning steps, especially increasing blr leads to more oscillation in the early steps. Decrease bgs leads to worse performance consistently. With more bgs decrease, the performance tends to be worse, accompanied by slower convergence.

- **Optional:** Add `-na` back to see how much it improves things. Also, set `video_log_freq 10`, then open TensorBoard and go to the "Images" tab to see some videos of your HalfCheetah walking along!

  We add normalized advantages for both methods with baseline and without baseline, as shown in figure 7 and 8. Normalizing baseline can improve the performance for both methods, the impact is more obvious without baseline, however, adding normalized advantage can still benefit version with baseline. From figure 8, we can see adding baseline leads to the almost same impact with normalizing advantage.

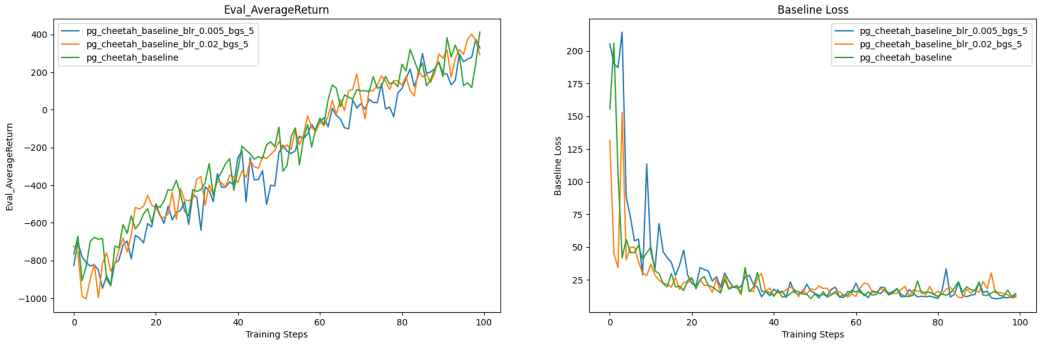


Figure 5: Average eval return and baseline loss with decreased baseline learning rate

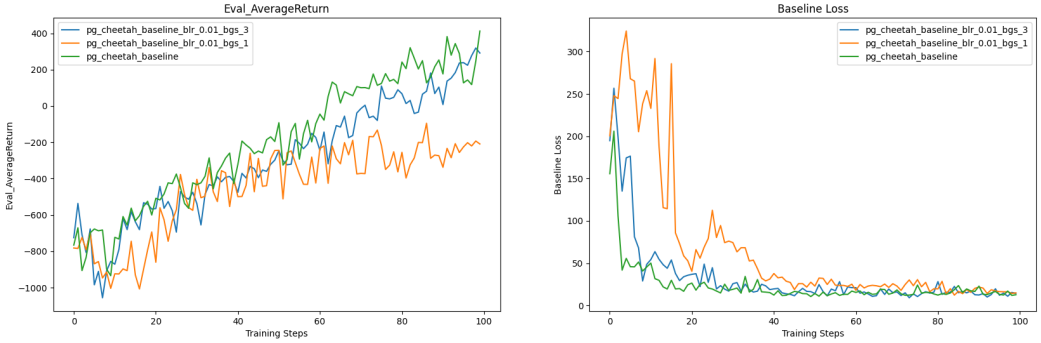


Figure 6: Average eval return and baseline loss with decreased baseline gradient steps

6 Generalized Advantage Estimation

- Provide a single plot with the learning curves for the `LunarLander-v2` experiments that you tried. Describe in words how λ affected task performance. The run with the best performance should achieve an average score close to 200 (180+).

  Figure 9 shows the results.

  Note: The average eval return did NOT get the expected score. I don't know where the bug is cause my gae implementation works fine for other environments (half cheetah and cart-pole).

- Consider the parameter λ. What does $\lambda = 0$ correspond to? What about $\lambda = 1$? Relate this to the task performance in `LunarLander-v2` in one or two sentences.

  $\lambda = 0$ corresponds to TD(0), which cancels the future rewards and only cares about the most recent reward. $\lambda = 1$ corresponds to Monte Carlo method, we use the rewards from entire trajectory and take them equally. TD(0) leads to lower variance at the cost of higher bias, Monte Carlo leads to the opposite. Take the λ between 0 and 1 as a method to make variance lower but keep the bias in a tolerable range.

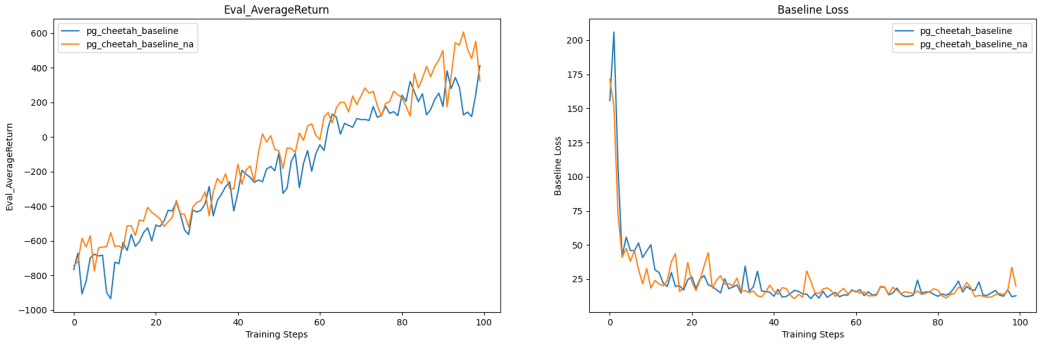


Figure 7: Average eval return and baseline loss with baseline and normalized advantage.

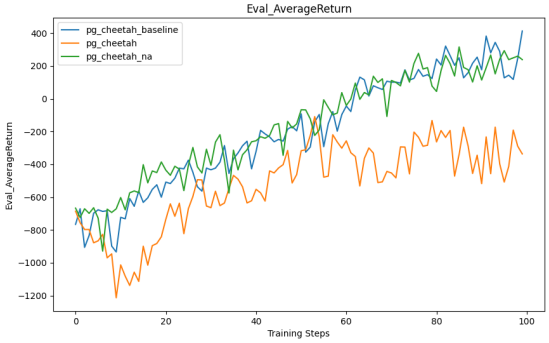


Figure 8: Average eval return with baseline and normalized advantage. Comparison with adding baseline and adding normalized advantage.

7 Hyperparameter Tuning

1. Provide a set of hyperparameters that achieve high return on `InvertedPendulum-v4` in as few environment steps as possible.
2. Show learning curves for the average returns with your hyperparameters and with the default settings, with environment steps on the x-axis. Returns should be averaged over 5 seeds.

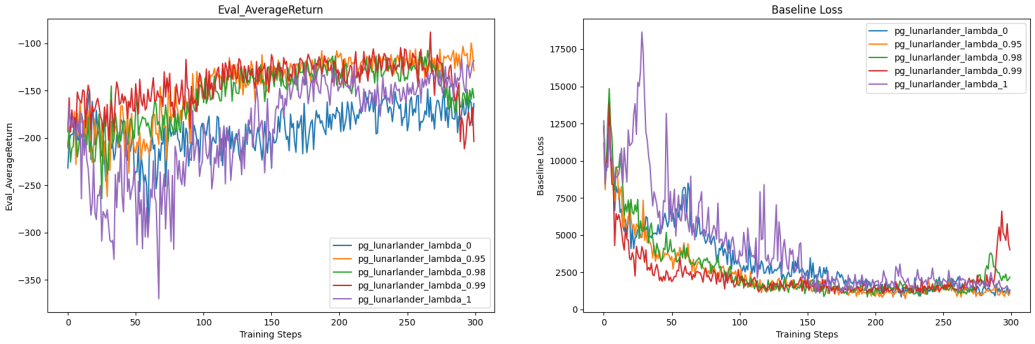


Figure 9: Average eval return and baseline loss using gae with different λ, These value are averaged over 10 runs respectively.

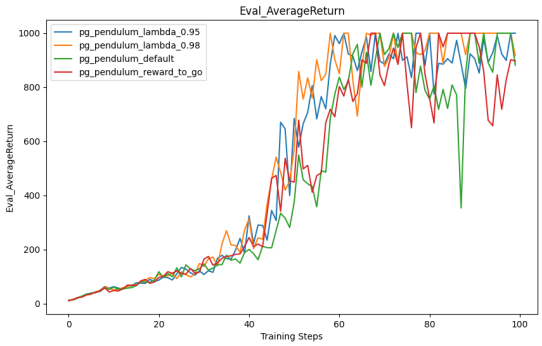


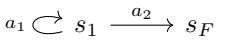
Figure 10: Pendulum average eval return under different settings. GAE with $\lambda = 0.95$ and $\lambda = 0.98$, reward to go, default setting are used.

8 (Extra Credit) Humanoid

1. Plot a learning curve for the Humanoid-v4 environment. You should expect to achieve an average return of at least 600 by the end of training. Discuss what changes, if any, you made to complete this problem (for example: optimizations to the original code, hyperparameter changes, algorithmic changes).

9 Analysis

Consider the following infinite-horizon MDP:



At each step, the agent stays in state s_1 and receives reward 1 if it takes action a_1, and receives reward 0 and terminates the episode otherwise. Parametrize the policy as stationary (not dependent on time) with a single parameter:

$$\pi_\theta(a_1|s_1) = \theta, \pi_\theta(a_2|s_1) = 1 - \theta$$

1. Applying policy gradients

   (a) Use policy gradients to compute the gradient of the expected return $R(\tau)$ with respect to the parameter θ. **Do not use discounting.**

   Hint: to compute $\sum_{k=1}^{\infty} k\alpha^{k-1}$, you can write:

   $$\sum_{k=1}^{\infty} k\alpha^{k-1} = \sum_{k=1}^{\infty} \frac{d}{d\alpha}\alpha^k = \frac{d}{d\alpha}\sum_{k=1}^{\infty}\alpha^k$$

   Solution: As we know $J(\theta) = \mathbb{E}_{\pi_\theta} R(\tau)$, we can rewrite it as:

   $$J(\theta) = \sum_{k=0}^{\infty} \theta^k(1-\theta)k$$

   Then the gradient of $J(\theta)$ w.r.t the parameter θ is:

   $$\nabla_\theta J(\theta) = \frac{d}{d_\theta}\sum_{k=0}^{\infty}\theta^k(1-\theta)k = (1-\theta)\frac{d}{d_\theta}\sum_{k=1}^{\infty}k\theta^k - \sum_{k=1}^{\infty}k\theta^k$$

   We can deal with $\sum_{k=1}^{\infty} k\theta^k$ by:

   $$\sum_{k=1}^{\infty}k\theta^k = \theta\sum_{k=1}^{\infty}k\theta^{k-1} = \theta\frac{d}{d_\theta}\sum_{k=1}^{\infty}\theta^k = \frac{\theta}{1-\theta^2}$$

   Thus, we can get:

   $$\nabla_\theta J(\theta) = (1-\theta)\frac{d}{d_\theta}\frac{\theta}{(1-\theta)^2} - \frac{\theta}{(1-\theta)^2} = \frac{1}{(1-\theta)^2}$$

   As $\theta \in [0, 1)$, it indicates that the gradient will increase when θ increases. If we always choose action a_1, our expected return $J(\theta)$ goes to infinity.

   (b) Compute the expected return of the policy $\mathbb{E}_{\tau\sim\pi_\theta} R(\tau)$ directly. Compute the gradient of this expression with respect to θ and verify that this matches the policy gradient.

   Solution: As the process is MDP, we can write

   $$\mathbb{E}_{\tau\sim\pi_\theta} R(\tau) = (1-\theta)\cdot 0 + \theta\cdot(1 + \mathbb{E}_{\tau\sim\pi_\theta} R(\tau))$$

We get $\mathbb{E}_{\tau\sim\pi_\theta} R(\tau) = \frac{\theta}{1-\theta}$. Take the gradient w.r.t θ:

$$\nabla_\theta J(\theta) = \frac{d}{d_\theta}(\frac{\theta}{1-\theta}) = \frac{1}{(1-\theta)^2}$$

which matches the gradient we get from the policy gradient.

2. Compute the variance of the policy gradient in closed form and describe the properties of the variance with respect to θ. For what value(s) of θ is variance minimal? Maximal? (Once you have an exact expression for the variance you can eyeball the min/max).

   Hint: Once you have it expressed as a sum of terms $P(\theta)/Q(\theta)$ where P and Q are polynomials, you can use a symbolic computing program (Mathematica, SymPy, etc) to simplify to a single rational expression.

   Solution:

3. Apply return-to-go as an advantage estimator.

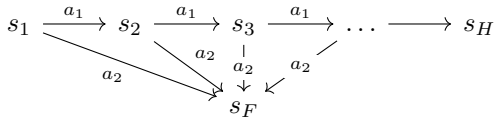
   (a) Write the modified policy gradient and confirm that it is unbiased.

   Solution:

   (b) Compute the variance of the return-to-go policy gradient and plot it on $[0, 1]$ alongside the variance of the original estimator.

   Solution:

4. Consider a finite-horizon H-step MDP with sparse reward:



The agent receives reward $R_{\max}$ if it arrives at s_H and reward 0 if it arrives at s_F (a terminal state). In other words, the return for a trajectory τ is given by:

$$R(\tau) = \begin{cases} 1 & \tau \text{ ends at } s_H \\ 0 & \tau \text{ ends at } s_F \end{cases}$$

Using the same policy parametrization as above, consider off-policy policy gradients via importance sampling. Assume we want to compute policy gradients for a policy π_θ with samples drawn from $\pi_{\theta'}$.

(a) Write the policy gradient with importance sampling.

(b) Compute its variance.

10 Survey

Please estimate, in minutes, for each problem, how much time you spent (a) writing code and (b) waiting for the results. This will help us calibrate the difficulty for future homeworks.

- **Policy Gradients:**
- **Neural Network Baseline:**
- **Generalized Advantage Estimation:**
- **Hyperparameters and Sample Efficiency:**
- **Humanoid:**
- **Humanoid:**
- **Analysis – applying policy gradients:**
- **Analysis – PG variance:**
- **Analysis – return-to-go:**
- **Analysis – importance sampling:**