Assignment 2: Policy Gradients

3 Policy Gradients

3.1 Learning Curves

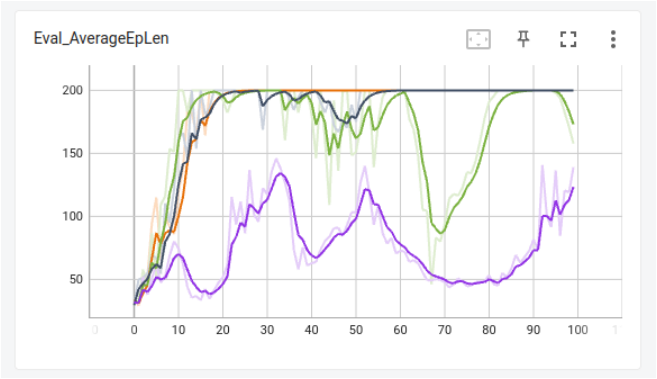


Figure 1: Small Batch (b=1000): Learning curves comparing baseline, reward-to-go (-rtg), advantage normalization (-na), and their combination.

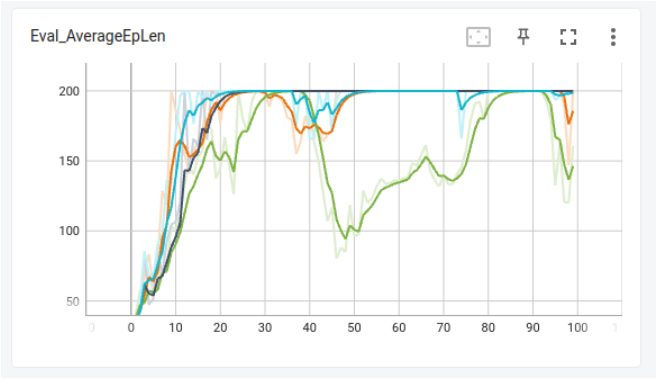


Figure 2: Large Batch (b=4000): Learning curves. For all plots in this assignment, the x-axis should be number of environment steps, logged as Train_EnvstepsSoFar (not number of policy gradient iterations)

3.2 Analysis

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

  Reward-to-go performs significantly better, converging faster with more stable performance near the maximum score of 200. The trajectory-centric estimator shows higher variance and inconsistent performance.

- **Did advantage normalization help?**

  Yes, advantage normalization reduced variance and improved stability in both batch settings, with more pronounced effects in the small batch case due to its inherent higher variance.

- **Did the batch size make an impact?**

  Yes, large batch (b=4000) produced much more stable learning curves with lower variance compared to small batch (b=1000). All large batch configurations reached and maintained the maximum score more consistently.

3.3 Commands Used

```
# Small batch experiments
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
--exp_name cartpole
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg --exp_name cartpole_rtg
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-na --exp_name cartpole_na
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg -na --exp_name cartpole_rtg_na

# Large batch experiments
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
--exp_name cartpole_lb
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
-rtg --exp_name cartpole_lb_rtg
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
-na --exp_name cartpole_lb_na
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
-rtg -na --exp_name cartpole_lb_rtg_na
```

4 Neural Network Baseline

4.1 Learning Curves

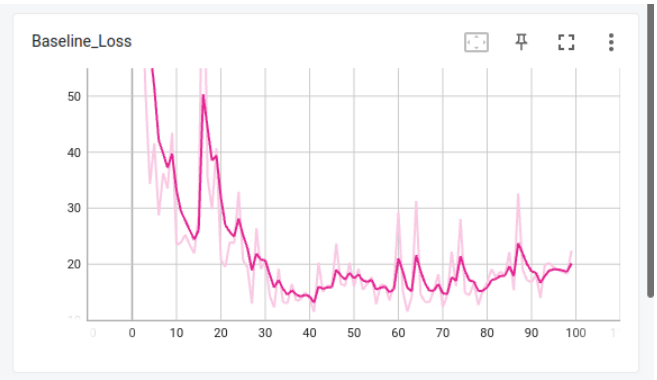


Figure 3: Baseline Loss: The neural network baseline's MSE loss decreases from ∼50 to ∼20 and stabilizes, indicating successful value function learning.

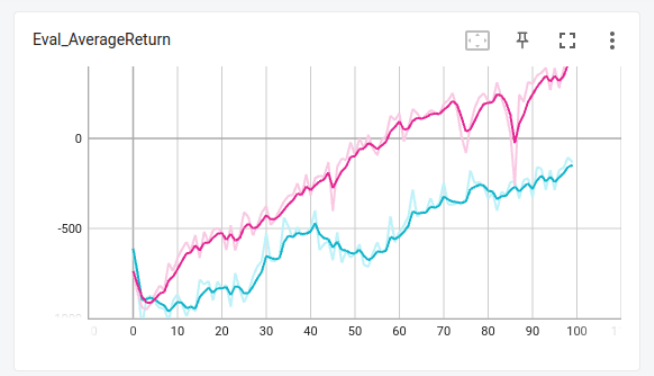


Figure 4: Eval Return: Pink curve (with baseline) reaches 300+ average return, while cyan curve (no baseline) remains near -100 in given iterations, demonstrating the baseline's significant performance improvement.

4.2 Questions

- Plot a learning curve for the baseline loss. **Answer:** See Figure 3 above.
- Plot a learning curve for the eval return. You should expect to achieve an average return over 300 for the baselined version. **Answer:** See Figure 4 above.
- 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?

  Answer: We varied baseline learning rate with blr $\in \{0.001, 0.005, 0.01\}$ (keeping bgs=5) and baseline gradient steps with bgs $\in \{1, 2, 3, 5\}$ (keeping blr=0.01).

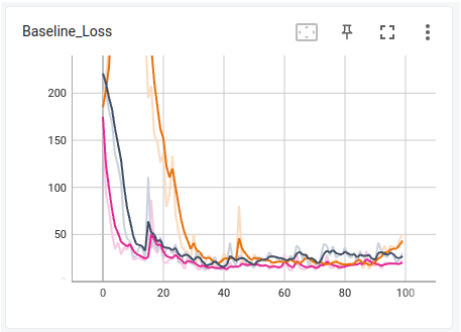


Figure 5: Baseline Loss Comparison: Orange (blr=0.001) converges slowest to ∼40, gray (blr=0.005) to ∼25, pink (blr=0.01) fastest to ∼20.

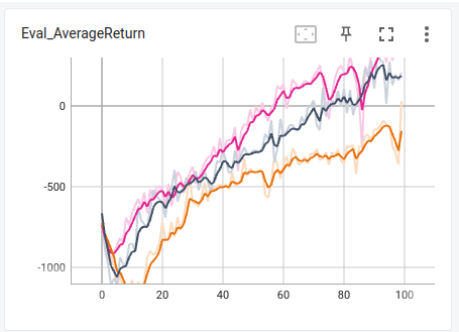


Figure 6: Policy Performance: Pink (blr=0.01) reaches 500+, gray (blr=0.005) ∼200, orange (blr=0.001) near 30.

We also varied baseline gradient steps with bgs $\in \{1, 2, 3, 5\}$ while keeping blr=0.01.

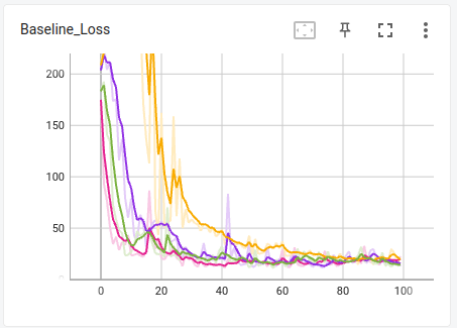


Figure 7: Baseline Loss (varying bgs): Orange (bgs=1) converges slowest, all others (bgs=2,3,5) converge similarly to ∼20.

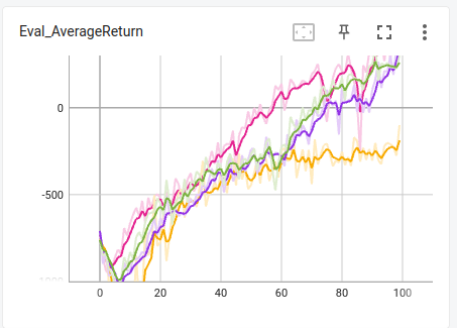


Figure 8: Policy Performance (varying bgs): Pink (bgs=5) and purple (bgs=3) reach 300+, others lower.

Lower blr and fewer bgs both result in slower convergence and higher final loss.

- **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!

4.3 Commands Used

```
# Section 4: Using a Neural Network Baseline
# Experiment 2: HalfCheetah-v4

# No baseline
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 \
-n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 --exp_name cheetah --which_gpu 0

# Baseline
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.01 -bgs 5 --exp_name cheetah_baseline --which_gpu 0
```

```
# Section 4: Grid search over baseline learning rate (blr)
# Keep baseline gradient steps fixed at 5
# Vary blr to see effect on baseline learning and policy performance

# Baseline learning rate = 0.001 (very low)
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.001 -bgs 5 --exp_name cheetah_baseline_blr0.001

# Baseline learning rate = 0.005
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.005 -bgs 5 --exp_name cheetah_baseline_blr0.005
```

```
# Baseline gradient steps = 1 (very low)
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.01 -bgs 1 --exp_name cheetah_baseline_bgs1

# Baseline gradient steps = 2
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.01 -bgs 2 --exp_name cheetah_baseline_bgs2

# Baseline gradient steps = 3
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.01 -bgs 3 --exp_name cheetah_baseline_bgs3
```

```
# Optional: Baseline with advantage normalization and video logging
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 -n 100 -b 5000 -rtg --discount 0.95 -lr 0.01 \
--use_baseline -blr 0.01 -bgs 5 -na --video_log_freq 10 --exp_name cheetah_baseline_na_video
```

5 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+).
- 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.

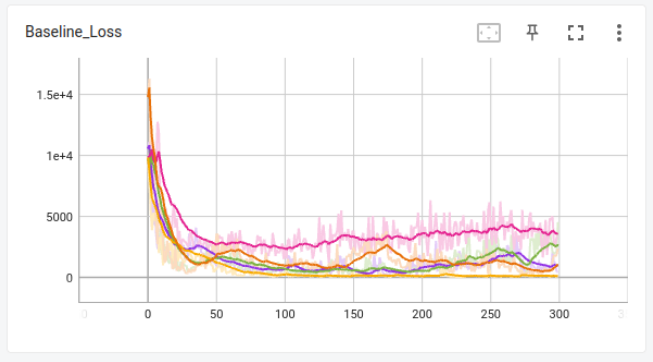


Figure 9: Baseline Loss: All configurations converge.

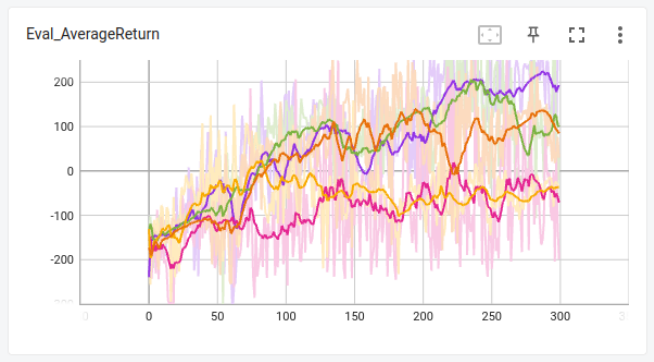


Figure 10: Eval Return: Higher λ values ($\lambda = 0.98$, $\lambda = 0.99$, $\lambda = 1$) achieve 180-200+ average return. Lower λ values show degraded performance

Answer: The parameter λ critically affects performance through the bias-variance tradeoff. Performance ranking from best to worst: $\lambda = 0.98$-0.99 (blue, green: 180-200+) $>$ $\lambda = 1$ (orange: 100-150) $>$ $\lambda = 0.95$ (light orange) $>$ $\lambda = 0$ (pink: -100).

The optimal performance at intermediate values $\lambda \in \{0.98, 0.99\}$ demonstrates the **key insight of GAE**. Both extremes are suboptimal: $\lambda = 0$ suffers from high bias due to relying entirely on bootstrapped value estimates and provides poor credit assignment in this sparse-reward environment (only looks one step ahead). Conversely, $\lambda = 1$ (pure Monte Carlo) has excessive variance from long trajectory rollouts, leading to noisy gradients and unstable learning. The sweet spot at $\lambda = 0.98$-0.99 captures long-horizon returns for proper credit assignment while incorporating slight TD bootstrapping to reduce variance, achieving the best bias-variance tradeoff.

$\lambda = 0$ corresponds to the one-step TD advantage estimator (high bias, low variance), while $\lambda = 1$ corresponds to pure Monte Carlo using full trajectory returns (low bias, high variance).

6 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.

First, train a policy for the inverted pendulum problem without GAE and with otherwise default settings.

```
for seed in $(seq 1 5); do
    python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v4 -n 100 \
    --exp_name pendulum_default_s$seed \
    -rtg --use_baseline -na \
    --batch_size 5000 \
    --seed $seed
done
```

During training, we have to choose many hyperparameters and settings:

1. **Discount factor** (γ): Controls the horizon of future rewards considered.
2. **Network architecture**: Number of layers and hidden units per layer.
3. **Batch size**: Small batches introduce high variance but enable frequent updates; large batches waste samples as the entire batch must be recollected for each gradient step.
4. **Learning rate**: Step size for policy parameter updates.
5. **Return-to-go**: Whether to use causal advantage estimation by computing returns from the current timestep onward.
6. **Advantage normalization**: Whether to normalize advantages to unit variance for stable training.
7. **GAE**: Whether to use Generalized Advantage Estimation, and if so, what λ value to balance bias-variance tradeoff.

The detailed hyperparameter tuning process is included in the section 6 scripts, and the final selected hyperparameters are the uncommented ones in `section6_tuned.sh`:

Different tuned parameters gave the plots below:

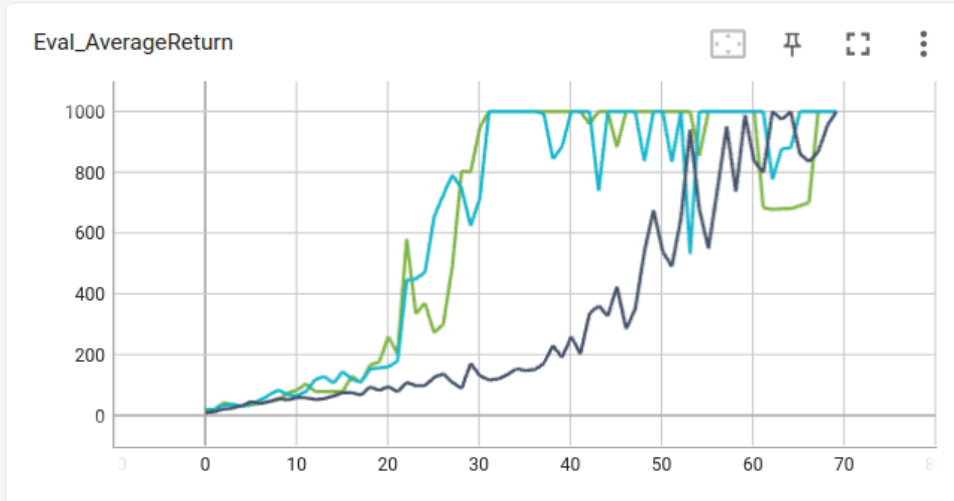


Figure 11: Hyperparameter Tuning Results: Comparison of different configurations on InvertedPendulum-v4. Cyan and green curves (tuned configurations) converge to maximum return around step 25-30 and maintain stability, while the gray curve (default configuration) requires 60 steps to converge.

7 (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).

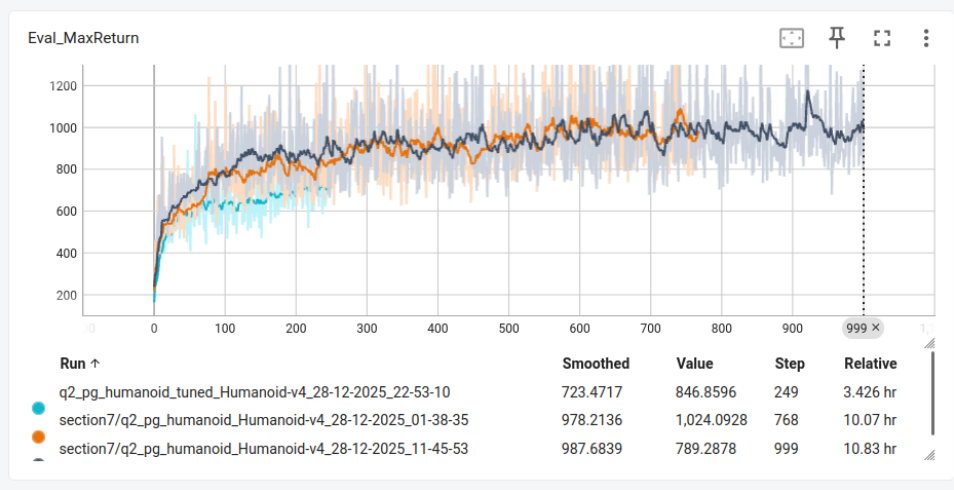
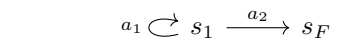


Figure 12: Humanoid-v4 Training Results: Learning curve showing evaluation return over training steps. The policy achieves an average return of 600+ by the end of training.

8 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:

   Using the policy gradient theorem:

   $$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\nabla_\theta \log \pi_\theta(\tau)\right]$$

   For a trajectory with k steps (taking a_1 exactly k times then a_2), we have $\pi_\theta(\tau) = \theta^k(1-\theta)$ and $R(\tau) = k$.

   $$\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta(k\log\theta + \log(1-\theta)) = \frac{k}{\theta} - \frac{1}{1-\theta}$$
   $$\nabla_\theta \log \pi_\theta(\tau)\cdot R(\tau) = \left(\frac{k}{\theta} - \frac{1}{1-\theta}\right)k = \frac{k^2}{\theta} - \frac{k}{1-\theta}$$

   $$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}\left[\frac{k^2}{\theta} - \frac{k}{1-\theta}\right] = \mathbb{E}\left[\frac{k^2}{\theta}\right] - \mathbb{E}\left[\frac{k}{1-\theta}\right] = \frac{\mathbb{E}[k^2]}{\theta} - \frac{\mathbb{E}[k]}{1-\theta}$$

   From the next part (b), we have $\mathbb{E}[k] = \frac{\theta}{1-\theta}$. Now we compute $\mathbb{E}[k^2]$:

   $$\begin{aligned}\mathbb{E}[k^2] &= \sum_{k=1}^{\infty} k^2\theta^k(1-\theta) = (1-\theta)\theta\sum_{k=1}^{\infty} k^2\theta^{k-1} \\ &= (1-\theta)\theta\frac{d}{d\theta}\sum_{k=1}^{\infty} k\theta^k \quad \text{(from part (b))} = (1-\theta)\theta\frac{d}{d\theta}\frac{\theta}{(1-\theta)^2} \quad \left(\text{since } \sum_{k=1}^{\infty} k\theta^k = \frac{\theta}{(1-\theta)^2}\right) \\ &= (1-\theta)\theta\cdot\frac{(1-\theta)^2 - \theta\cdot 2(1-\theta)(-1)}{(1-\theta)^4} = (1-\theta)\theta\cdot\frac{1+\theta}{(1-\theta)^3} = \frac{\theta(1+\theta)}{(1-\theta)^2}\end{aligned}$$

   Substituting back:

   $$\begin{aligned}\nabla_\theta \mathbb{E}[R(\tau)] &= \frac{\mathbb{E}[k^2]}{\theta} - \frac{\mathbb{E}[k]}{1-\theta} \\ &= \frac{1}{\theta}\cdot\frac{\theta(1+\theta)}{(1-\theta)^2} - \frac{1}{1-\theta}\cdot\frac{\theta}{1-\theta} \\ &= \frac{1+\theta}{(1-\theta)^2} - \frac{\theta}{(1-\theta)^2} = \frac{1}{(1-\theta)^2}\end{aligned}$$

(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:

$$\begin{aligned}\mathbb{E}_{\tau\sim\pi_\theta} R(\tau) &= \sum_{k=0}^{\infty} P(\text{trajectory with } k \text{ steps}) \cdot R(\tau) = \sum_{k=0}^{\infty} \theta^k (1-\theta)\cdot k = (1-\theta)\sum_{k=1}^{\infty} k\theta^k = (1-\theta)\cdot\theta\sum_{k=1}^{\infty} k\theta^{k-1} \\ &= (1-\theta)\theta\frac{d}{d\theta}\sum_{k=1}^{\infty}\theta^k \\ &= (1-\theta)\theta\frac{d}{d\theta}\frac{\theta}{1-\theta} = (1-\theta)\theta\cdot\frac{1}{(1-\theta)^2} = \frac{\theta}{1-\theta} \\ \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta} R(\tau) &= \frac{d}{d\theta}\frac{\theta}{1-\theta} = \frac{1}{(1-\theta)^2}\end{aligned}$$

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:

From Question 1, the policy gradient estimator for a single trajectory with k steps is:

$$\hat{g}(\theta) = \nabla_\theta \log \pi_\theta(\tau)\cdot R(\tau) = \frac{k^2}{\theta} - \frac{k}{1-\theta}$$

We already computed $\mathbb{E}[\hat{g}(\theta)] = \frac{1}{(1-\theta)^2}$ in Question 1.

To compute the variance, we use $\mathrm{Var}[\hat{g}] = \mathbb{E}[\hat{g}^2] - (\mathbb{E}[\hat{g}])^2$.

Compute $\mathbb{E}[\hat{g}^2]$:

$$\hat{g}^2 = \left(\frac{k^2}{\theta} - \frac{k}{1-\theta}\right)^2 = \frac{k^4}{\theta^2} - \frac{2k^3}{\theta(1-\theta)} + \frac{k^2}{(1-\theta)^2}$$

Taking expectation:

$$\mathbb{E}[\hat{g}^2] = \frac{\mathbb{E}[k^4]}{\theta^2} - \frac{2\mathbb{E}[k^3]}{\theta(1-\theta)} + \frac{\mathbb{E}[k^2]}{(1-\theta)^2}$$

Using the moments from Question 3 in the next part:

$$\begin{aligned}\mathbb{E}[\hat{g}^2] &= \frac{1}{\theta^2}\cdot\frac{\theta(1+11\theta+11\theta^2+\theta^3)}{(1-\theta)^4} - \frac{2}{\theta(1-\theta)}\cdot\frac{\theta(1+4\theta+\theta^2)}{(1-\theta)^3} + \frac{1}{(1-\theta)^2}\cdot\frac{\theta(1+\theta)}{(1-\theta)^2} \\ &= \frac{1+11\theta+11\theta^2+\theta^3}{\theta(1-\theta)^4} - \frac{2\theta(1+4\theta+\theta^2)}{\theta(1-\theta)^4} + \frac{\theta(1+\theta)}{(1-\theta)^4} \\ &= \frac{1+11\theta+11\theta^2+\theta^3-2\theta-8\theta^2-2\theta^3+\theta^2+\theta^3}{\theta(1-\theta)^4} = \frac{1+9\theta+4\theta^2}{\theta(1-\theta)^4}\end{aligned}$$

$$\begin{aligned}\mathrm{Var}[\hat{g}(\theta)] &= \mathbb{E}[\hat{g}^2] - (\mathbb{E}[\hat{g}])^2 \\ &= \frac{1+9\theta+4\theta^2}{\theta(1-\theta)^4} - \frac{1}{(1-\theta)^4} = \frac{1+8\theta+4\theta^2}{\theta(1-\theta)^4}\end{aligned}$$

- Variance is **minimal** at $\theta \approx 0.107$ (found numerically by setting $\frac{d}{d\theta}\mathrm{Var}[\hat{g}] = 0$)
- Variance is **maximal** as $\theta \to 0^+$ or $\theta \to 1^-$ (blows up at the boundaries)

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

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

   Solution:

   Before: $\nabla_\theta \log \pi_\theta(\tau) \cdot R(\tau) = \left(\frac{k}{\theta} - \frac{1}{1-\theta}\right) \cdot k.$

   With return-to-go, instead of multiplying each action's log probability by the total return $R(\tau) = k$, we multiply it by the return from that timestep onward. At timestep t (where $t \in 1, 2, ..., k$), the return-to-go is $(k - t + 1)$ (number of remaining rewards of 1).

   The original policy gradient multiplies each action's log probability by the total trajectory return, crediting actions for rewards that occurred before they were taken. Return-to-go only credits each action for **rewards from that timestep onward**, respecting **causality** since an action at time t cannot influence past rewards.

   Return-to-go exists to provide an unbiased estimate of the future cumulative reward and removes mis-crediting bias per action

   (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: Use the variance formula: $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$

$$\hat{g}_{\text{rtg}}(\theta) = \sum_{t=1}^{k} \frac{1}{\theta} \cdot (k - t + 1) = \frac{1}{\theta} \sum_{t=1}^{k} (k - t + 1) = \frac{1}{\theta} \sum_{j=1}^{k} j = \frac{1}{\theta} \frac{k(k+1)}{2} = \frac{k(k+1)}{2\theta}$$

$$\mathbb{E}[\hat{g}_{\text{rtg}}] = \frac{1}{2\theta} \mathbb{E}[k^2 + k] = \frac{1}{2\theta} \left(\frac{\theta(1+\theta)}{(1-\theta)^2} + \frac{\theta}{1-\theta} \right) = \frac{1}{(1-\theta)^2}$$

$$\mathbb{E}[\hat{g}_{\text{rtg}}^2] = \frac{1}{4\theta^2} \mathbb{E}[k^4 + 2k^3 + k^2]$$

For a geometric distribution with parameter $p = 1 - \theta$ (probability of stopping):

$\mathbb{E}[k] = \frac{\theta}{1-\theta}$ $\mathbb{E}[k^2] = \frac{\theta(1+\theta)}{(1-\theta)^2}$ $\mathbb{E}[k^3] = \frac{\theta(1+4\theta+\theta^2)}{(1-\theta)^3}$ $\mathbb{E}[k^4] = \frac{\theta(1+11\theta+11\theta^2+\theta^3)}{(1-\theta)^4}$

$$\mathbb{E}[\hat{g}_{\text{rtg}}^2] = \frac{1}{4\theta^2} \mathbb{E}\left[K^4 + 2K^3 + K^2\right] = \frac{1}{4\theta^2} \left[\frac{\theta(1 + 11\theta + 11\theta^2 + \theta^3)}{(1-\theta)^4} + 2\frac{\theta(1 + 4\theta + \theta^2)}{(1-\theta)^3} + \frac{\theta(1+\theta)}{(1-\theta)^2} \right] = \frac{1 + 4\theta + \theta^2}{\theta(1-\theta)^4}.$$

$$\text{Var}[\hat{g}_{\text{rtg}}] = \mathbb{E}[\hat{g}_{\text{rtg}}^2] - \left(\frac{1}{(1-\theta)^2} \right)^2 = \frac{1 + 4\theta + \theta^2}{\theta(1-\theta)^4} - \frac{1}{(1-\theta)^4} = \frac{1 + 4\theta + \theta^2 - \theta}{\theta(1-\theta)^4} = \frac{1 + 3\theta + \theta^2}{\theta(1-\theta)^4}$$

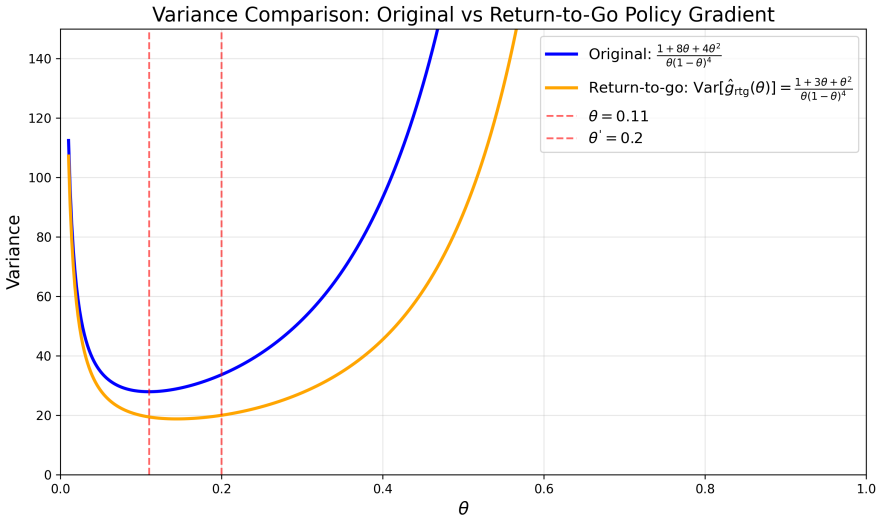
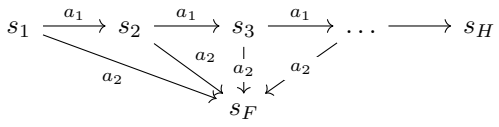


Figure 13: Variance comparison between original and return-to-go policy gradient estimators over $\theta \in [0, 1]$. The return-to-go estimator exhibits lower variance.

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.