CMPUT 655 Assignment 8

Seth Akins

October 2024

1 Policy Gradient Theory Questions

1.1 Differences Between Policy Gradient and Value-Based Methods

One of the main differences between value-based and policy gradient methods is that value-based methods learn the values of states or state-action pairs, while policy gradient methods learn a parameterized policy that selects actions by maximizing a performance estimate. Value methods minimize the mean squared error, whereas policy gradient methods maximize the performance of the policy by optimizing the reward function $J(\theta)$. This function computes the reward for following the parameterized policy and is typically the value of the start state following the parameterized policy.

1.2 Importance of the Policy Gradient Theorem

The policy gradient theorem is so important because the derivative of the policy performance contains the stationary distribution of states for the current policy. This distribution changes indirectly based on the interaction of the current policy with the environment, which is difficult to measure. Instead, the policy gradient theorem tells us that there is a derivative proportional to the original which only involves taking the derivative of the policy parameters, not the stationary distribution.

1.3 The REINFORCE Method

The REINFORCE method is a Monte Carlo method, meaning it has low bias, but high variance. We use a baseline to help alleviate this bias.

1.4 Actor-Critic Methods

Actor-critic methods learn both a critic, a value function, and an actor, a parameterized policy. The critic estimates the value of states and this is used to update the actor, which picks our actions. The actor is updated based on how good the critic thinks the state is, so the critic evaluates the choice of actions of the current policy.

In the use of a critic, we have a baseline which helps to mitigate the high variance issues in REINFORCE; however, to train the critic, we use stochastic gradient descent with bootstrapping, and this introduces bias into the algorithm which is not present when we use Monte Carlo methods.

2 Min-Variance Constant Baseline

For the REINFORCE algorithm, the gradient estimate is as follows:

$$\nabla J(\theta) \propto \mathbb{E}[G_t \nabla \ln \pi(A_t|S_t, \theta)] \tag{1}$$

With the inclusion of a baseline, denoted as B, this becomes the following:

$$\nabla J(\theta) \propto \mathbb{E}[(G_t - B)\nabla \ln \pi(A_t|S_t, \theta)] \tag{2}$$

The main point of the baseline is to help mitigate the large variance of each update due to the REINFORCE algorithm being a Monte Carlo method. As such, we take the variance of our update, letting $D = \nabla \ln \pi(A_t|S_t, \theta)$ for simplicity:

$$\begin{aligned}
\mathrm{Var}[\nabla J(\theta)] &= \mathrm{Var}[(G_t - B)D] \\
&= \mathbb{E}[((G_t - B)D)^2] - (\mathbb{E}[(G_t - B)D])^2 \\
&= \mathbb{E}[((G_t - B)D)^2] - (\mathbb{E}[G_t D - BD])^2 \\
&= \mathbb{E}[((G_t - B)D)^2] - (\mathbb{E}[G_t D] - \mathbb{E}[BD])^2 \\
&= \mathbb{E}[((G_t - B)D)^2] - (\mathbb{E}[G_t D])^2
\end{aligned} \tag{3}$$

The last two steps are possible due to the linearity of expectations, and because B is a constant and $\mathbb{E}[D] = 0$.

Now, to minimize the variance, we differentiate the final equation in (3), denoted by V, with respect to the baseline:

$$\begin{aligned}
\frac{dV}{dB} &= \mathbb{E}[((G_t - B)D)^2] - (\mathbb{E}[G_tD])^2 \\
&= \frac{d}{dB}(\mathbb{E}[((G_t - B)D)^2]) - \frac{d}{dB}((\mathbb{E}[G_tD])^2) \\
&= \frac{d}{dB}\mathbb{E}[D^2G_t^2 + B^2D^2 - 2BD^2] \\
&= \frac{d}{dB}(\mathbb{E}[D^2G_t^2] + \mathbb{E}[B^2D^2] - \mathbb{E}[2BD^2G_t]) \\
&= \frac{d}{dB}(\mathbb{E}[B^2D^2] - \mathbb{E}[2BD^2G_t]) \\
&= \frac{d}{dB}(B^2\mathbb{E}[D^2] - 2B\mathbb{E}[D^2G_t]) \\
&= 2B\mathbb{E}[D^2] - 2\mathbb{E}[D^2G_t]
\end{aligned} \tag{4}$$

Setting the derivative to be zero, we get the following:

$$\begin{aligned}
0 &= 2B\mathbb{E}[D^2] - 2\mathbb{E}[D^2G_t] \\
\mathbb{E}[D^2G_t] &= B\mathbb{E}[D^2] \\
B &= \frac{\mathbb{E}[D^2G_t]}{\mathbb{E}[D^2]}
\end{aligned} \tag{5}$$

Using samples from our environment, we can estimate the optimal baseline in (5). This can be done as follows:

$$B = \frac{\frac{1}{t}D^2G_t}{\frac{1}{t}D^2} \tag{6}$$

3 Exercise 13.2

To prove the policy gradient theorem with discounting, we start with the derivative of the state-value function with respect to θ:

$$\begin{aligned}
\nabla_\theta v_\pi(s) &= \nabla_\theta \left[\sum_a \pi(a|s,\theta) q_\pi(s,a)\right] \\
&= \sum_a \left[\nabla_\theta \pi(a|s,\theta) q_\pi(s,a) + \pi(a|s,\theta)\nabla_\theta q_\pi(s,a)\right] \quad \text{(Product Rule)} \\
&= \sum_a \left[\nabla_\theta \pi(a|s,\theta) q_\pi(s,a) + \pi(a|s,\theta)\nabla_\theta \sum_{s',r} p(s',r|s,a)\left(r + \gamma v_\pi(s')\right)\right] \\
&= \sum_a \left[\nabla_\theta \pi(a|s,\theta) q_\pi(s,a) + \pi(a|s,\theta)\nabla_\theta \sum_{s',r} p(s',r|s,a)\gamma v_\pi(s')\right] \\
&\quad \text{(as neither } P \text{ or } r \text{ is a function of } \theta) \\
&= \sum_a \left[\nabla_\theta \pi(a|s,\theta) q_\pi(s,a) + \pi(a|s,\theta)\nabla_\theta \sum_{s'} p(s'|s,a)\gamma v_\pi(s')\right] \\
&= \sum_a \nabla_\theta \pi(a|s,\theta) q_\pi(s,a) + \sum_a \pi(a|s,\theta)\nabla_\theta \sum_{s'} p(s'|s,a)\gamma v_\pi(s')
\end{aligned}$$

Now, we define the probability of transitioning from state s to x in k steps under policy π as $\rho_\pi(s \to x, k)$. We will also denote the first time in the equation as $\phi(s) = \sum_a \nabla_\theta \pi(a|s,\theta) q_\pi(s,a)$. Using this, we can continue to expand the derivative:

$$\begin{aligned}
\nabla_\theta v_\pi(s) &= \phi(s) + \sum_a \pi(a|s,\theta) \sum_{s'} p(s'|s,a)\gamma \nabla_\theta v_\pi(s') \\
&= \phi(s) + \sum_{s'}\sum_a \pi(a|s,\theta) p(s'|s,a)\gamma \nabla_\theta v_\pi(s') \\
&= \phi(s) + \sum_{s'} \rho_\pi(s \to s', 1)\gamma \nabla_\theta v_\pi(s') \\
&= \phi(s) + \sum_{s'} \rho_\pi(s \to s', 1)\gamma \left[\phi(s') + \sum_{s''} \rho_\pi(s' \to s'', 2)\gamma^2 \nabla_\theta v_\pi(s'')\right] \\
&= \phi(s) + \sum_{s'} \rho_\pi(s \to s', 1)\gamma \phi(s') + \sum_{s''} \rho_\pi(s \to s'', 2)\gamma^2 \nabla_\theta v_\pi(s'') \\
&= \phi(s) + \sum_{s'} \rho_\pi(s \to s', 1)\gamma \phi(s') + \sum_{s''} \rho_\pi(s \to s'', 2)\gamma^2 \nabla_\theta v_\pi(s'') + \sum_{s'''} \rho_\pi(s \to s''', 3)\gamma^3 \nabla_\theta v_\pi(s''') \\
&= \sum_{x \in S} \sum_{k=0}^{\infty} \rho_\pi(s \to x, k)\gamma^k \phi(x)
\end{aligned}$$

We also need to define the on-policy distribution for episodic tasks with discounting. Let $h(s)$ be the probability of starting an episode in state s, and $\overline{s}$ are preceding states of s. Then, the on-policy distribution is as follows:

$$\eta_\gamma(s) = h(s) + \sum_{\overline{s}} \eta_\gamma(\overline{s}) \sum_a \gamma \pi(a|\overline{s}, \theta) p(s|\overline{s}, a), \quad \forall s \in S \tag{7}$$

By definition, this is equivalent to the $\sum_{k=0}^{\infty} \rho_\pi(s_0 \to, s, k)\gamma^k$ term in the equation above. With this and our derivative above, we can use it with the evaluation function $J(\theta)$ as follows:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta v_\pi(s_0) \\
&= \sum_s \sum_{k=0}^{\infty} \rho_\pi(s_0 \to, s, k)\gamma^k \phi(s) \\
&= \sum_s \eta_\gamma(s)\phi(s) \\
&= \left(\sum_s \eta_\gamma(s)\right) \sum_s \frac{\eta_\gamma(s)}{\sum_s \eta_\gamma(s)} \phi(s) \\
&\quad \text{(We normalize } \eta_\gamma(s) \text{ so it becomes a probability distribution)} \\
&\propto \sum_s \frac{\eta_\gamma(s)}{\sum_s \eta_\gamma(s)} \phi(s) \quad \text{(As } \sum_s \eta_\gamma(s) \text{ is a constant.)} \\
&= \sum_s \mu_\gamma(s)\phi(s) \\
&\quad \text{(Where } \mu_\gamma(s) \text{ is the stationary distribution depending on the discount factor)}
\end{aligned}$$

Using this equation, we can write $\phi(s)$ as it was originally defined above to get the following:

$$\begin{aligned}
\nabla_\theta J(\theta) &\propto \sum_s \mu_\gamma(s) \sum_a \nabla_\theta \pi(a|s, \theta) q_\pi(s, a) \\
&= \sum_s \mu_\gamma(s) \sum_a q_\pi(s, a) \pi(a|s, \theta) \frac{\nabla_\theta \pi(a|s, \theta)}{\pi(a|s, \theta)} \\
&= \mathbb{E}_\pi[\gamma^t q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta)]
\end{aligned}$$

On the last line, we convert the sums to an expectation based on our policy π_θ which we then sample with samples s_t and a_t. Sampling following our policy ensures we will encounter the states according to the distribution μ_γ, but it depends on the discount factor for the sample at that time step.

From this equation, we can finish the derivation of the update for the REINFORCE algorithm, using the fact that we use samples of the expected return, G_t, instead of our action-value function:

$$\begin{aligned}\nabla_\theta J(\theta) &\propto \mathbb{E}_\pi[\gamma^t q_\pi(s_t, a_t)\nabla_\theta \ln \pi(a_t|s_t, \theta)] \\ &= \mathbb{E}_\pi[\gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta)]\end{aligned}$$

Now, we can use these samples to update our θ using SDG as follows:

$$\theta_{t+1} = \theta_t + \alpha\gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta) \tag{8}$$

This concludes the proof.

4 Coding Implementation

My implementation for the code of this assignment and the graphs can be found here.

4.1 Graphs

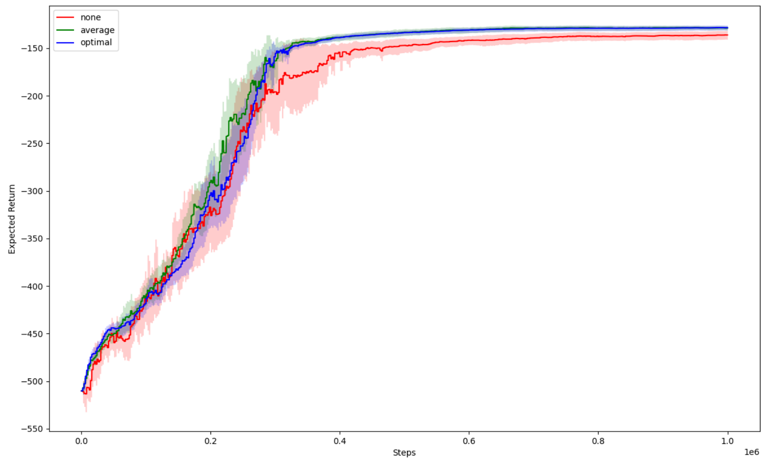

Figure 1: Comparison of Expected Returns in the Pendulum Environment with Different Baselines

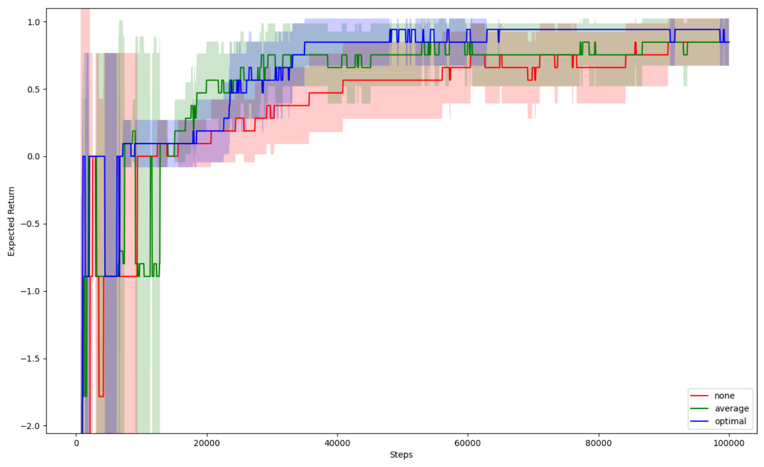

Figure 2: Comparison of Expected Returns in the Gridworld Environment with Different Baselines

5 Coding Questions

5.1 Controlling Exploration

Our algorithm does not learn epsilon in the softmax policy and does not learn sigma in the Gaussian policy.

5.2 Softmax Explortation

Letting $h(s, b; w)$ be the trained function, the formula for the softmax equation is as follows:

$$\pi(a|s, w) = \frac{e^{h(s,a;w)}}{\sum_b e^{h(s,b;w)}} \tag{9}$$

As we learn h, the preference for higher-valued state-action pairs grows. This increases the likelihood of choosing those actions, as their probability becomes greater compared to other actions, making the policy more greedy with respect to those actions.

5.3 Algorithm Improvement

To further exploration for our algorithm, we could learn the values for both epsilon and sigma using SGD. This could increase the algorithm's performance in more complicated environments requiring more exploration.

Also, instead of using Monte Carlo to estimate the gradient, we could use an algorithm such as Q-Learning.