CS 593 Robotics Assignment 4 : Policy Gradients

March 22, 2022

In this assignment, you'll use policy gradient methods to solve gym environments with discrete and continuous action spaces. In the first part of this assignment you will solve the CartPole-v1 environment and for the second part you solve a custom reach task for a 2-link arm.

For each of the following questions, you should implement your agent's policy using a fully connected neural network in PyTorch.

Question 1 - CartPole-v1

The goal of the CartPole environment is to keep a pole balanced on a cart. The observation space consists of the cart's position and velocity as well as the pole's angle and angular velocity. The action space is $\{0, 1\}$; 0 means move left, and 1 means move right. For more details see https://gym.openai.com/envs/CartPole-v1/.
Since the action space of the environment is discrete, the policy should return a categorical probability distribution over the set of actions. (You may find the Softmax layer useful for converting the neural network's outputs into a probability distribution).
For parts 1, 2 and 3, train the policy using 200 iterations of policy updates, with 500 episodes per iteration. For training set $\gamma = 0.99$ and for plotting the average rewards, use the data collected for training.

1. Implement a vanilla reinforce algorithm given by the following gradient update for your policy.

$$\nabla J(\theta) \approx \frac{1}{n} \sum_{i=0}^{n} G(\tau)(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)) \tag{1}$$

   Where $G(\tau) = \sum_{t=0}^{T} \gamma^t r(t)$ is the Monte Carlo estimate of the discounted reward, $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t given s_t and n is the number of episodes.

   Plot the average reward at each iteration. Note that the total reward of an episode τ is $R(\tau) = \sum_{t=0}^{T} r(t)$, which is not the same as the discounted reward $G(\tau)$.

   Hint: Try to use this equation to construct a loss function for your network.

2. As discussed in class, the gradient at step t is only dependent on the future rewards, hence we can modify the policy gradient as follows:

$$\nabla J(\theta) \approx \frac{1}{n} \sum_{i=0}^{n} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} \gamma^{t'-t} r(t') \tag{2}$$

Implement the policy gradient algorithm using the above update rule. Plot the average reward at each iteration. Compare your results with question 1.1.

3. To reduce the variance of the estimated rewards, subtract the rewards using a constant b such that the mean of the modified rewards is 0. It is also observed empirically that dividing the modified rewards with the standard deviation σ of the estimated rewards improves training. Implement the policy gradient with these modifications.

$$\nabla J(\theta) \approx \frac{1}{n} \sum_{i=0}^{n} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \frac{\sum_{t'=t}^{T} (\gamma^{t'-t} r(t')) - b}{\sigma} \tag{3}$$

   Plot the average rewards at each iteration. Compare your results with question 1.1 and 1.2.

4. Train the policy with the update in question 1.3 using 100, 300, and 1000 episodes per iteration. (Again, use 200 iterations). For each episode count, plot the average rewards in each iteration. Does increasing the episode count improve training?

Question 2 - 2 Link Arm

NOTE: Before starting this assignment, download the folder modified-gym-env, and install the package using `pip install -e modified-gym-env/`. This will give access to the modified reach environment using

```
env = gym.make("modified_gym_env:ReacherPyBulletEnv-v1")
```

In this environment, the robot is a 2D manipulator similar to assignment 2. The goal is to move the arm from the initial position to the target position as quickly as possible, using the policy gradient method.
Since the action space is continuous, implement a policy that samples from a multivariate normal distribution. ie $a \sim N(f(s), \Sigma)$, where $f(s)$ is a neural network and Σ a diagonal covariance matrix. Train your policy with $\gamma = 0.9$, and try to find the smallest possible number of episodes that succeeds. Plot the average reward for each iteration for your final model. Also upload a gif of your final policy completing the task for a single episode on canvas(you can use screen capture software to record the videos).
Expected result: The final policy should be able to reach the target location for the randomly initialized reach task.

Notes

Here are a few suggestions that you could use to complete the assignment:

1. Stochastic policies - To implement the stochastic policies, you can make use of the torch.distributions library. The library has inbuilt functions to sample from the distribution and return corresponding log probabilities of sampled points.

2. Implementing stochastic policies for continuous action space - To implement the Gaussian policy you can use the Multivariate Normal function class from the torch.distributions library. The mean of the Gaussian should be a function of your state and the covariance matrix can be a diagonal matrix initialized with constants of your choosing. Also make sure that the eigenvalues of your covariance matrix are positive, and at least 10^{-3}.

3. Understanding how PyTorch computes gradients is essential for this assignment. If you've never used PyTorch before, visit this tutorial to learn about the autograd module.

4. For question 2, you can fix the robot arm initial position by passing the argument rand_init=False in the gym.make command. This is a simpler environment to train. **You do not need to plot the performance of this environment in your report**. This is provided for you to check your implementation.

5. Seed values - For question 2, train with different pytorch seed values to make sure your code is working.

6. Visuals - For question 2, be sure to `import pybullet`, or the rendering won't work.

Submission Instructions

Your submission should consist of code as well as a PDF report that presents and analyzes your results.

It should also include instructions for running your code; these may appear either in the report itself, or in a separate README file. Be specific enough that we can reproduce your results, if needed.

Bundle all files into a zip or tarfile, and submit it via Brightspace. Name your submission "<your_username>.zip" or "<your_username>.tar.gz". For example, I would name my submission "thonegge.zip".