P2 Report

Learning Algorithm:

- The learning algorithm is PPO with loss function = clipped surrogate + MSE + entropy. The algorithm includes three neural networks which are actor network, sigma network and critic network. The action network predicts the action, the sigma network predicts the standard deviation of the action and the critic network predicts state value.

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

- The advantage function can be switched between GAE, TD or Monte Carlo method. In the experiment, I use GAE
- Hyperparameter settings
    - Expected reward discount rate = 0.99
    - GAE discount rate = 0.8
    - Epsilon for ratio clip = 0.2 with decay rate = 0.998
    - Entropy loss weight (beta) = 0.01 with decay rate = 0.99
    - MSE loss weight (mse_w) = 0.5
    - Buffer tmax = 1000
    - Steps per epoch = 20. For each step, there are 50 mini steps with batch size = 20

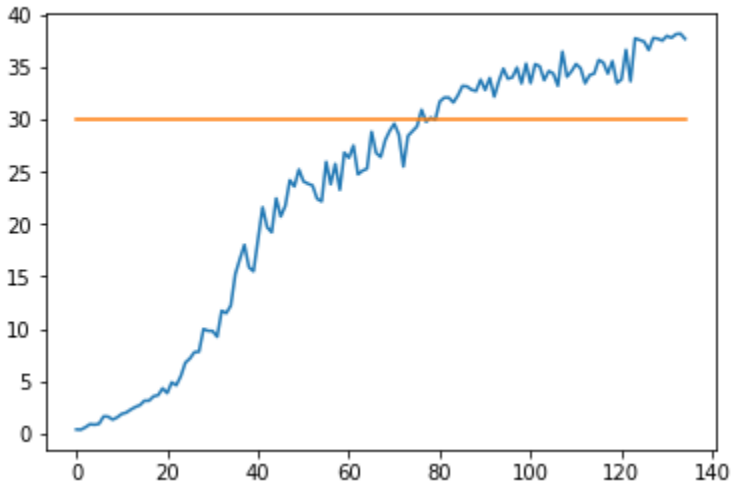
- Neural Network Model Architecture
    - Actor network (predict action)
        - Three dense layers. The first two dense layer are followed by relu activation function and the last dense layer is followed by tanh activation function.
        - RELU (Linear (33, 64))
        - RELU (Linear (64, 64))
        - Tanh (Linear (64, 4))
    - Sigma network (predict std of action)
        - Three dense layers. The first two dense layer are followed by relu activation function and the last dense layer is followed by sigmoid activation function.
        - RELU (Linear (33, 64))
        - RELU (Linear (64, 64))
        - Sigmoid (Linear (64, 4))

- Critic network (predict state value)
  - Three dense layers. The first two dense layer are followed by relu activation function and there is no activation function for the last dense layer.
  - RELU (Linear (33, 64))
  - RELU (Linear (64, 64))
  - Linear (64, 1)

Plot of Rewards:

```
training loop: 100% |##########################################| Time: 0:21:36
Environment solved in 34 episodes!      Average Score: 30.14
```

```
[<matplotlib.lines.Line2D at 0x7f5675ec0c50>]
```



Ideas of Future Works:

- Can try different algorithm such as DDPG, A3C, or D4PG.
- Can further optimize hyper parameters.
- Can further compare performance among different advantage estimation, GAE, TD and Monte Carlo.
- Try different neural network architecture to see if it speed up training