ECE433/COS435 Introduction to RL
Assignment 5: Proximal Policy Optimization (PPO)
Spring 2025

Fill me in

Your name here.

Due March 28, 2025

Collaborators

Fill me in

Please fill in the names and NetIDs of your collaborators in this section.

Instructions

Writeups should be typesetted in Latex and submitted as PDFs. You can work with whatever tool you like for the code, but **please submit the asked-for snippet and answer in the solutions box as part of your writeup. We will only be grading your write-up.**

Question 1.

In this assignment, we will implement Proximal Policy Optimization (PPO) for the CartPole-v1 environment from OpenAI Gym. Along the way, we will incorporate several important policy gradient concepts:

- REINFORCE (Monte Carlo Policy Gradient)
- Importance Weighting (IW) (For Off-Policy Learning)
- Generalized Advantage Estimation (GAE) (To Balance Bias and Variance)

Question 1.a

First, you need to implement the policy network, which decides the actions to take, and the value network, which estimates the returns. Paste your class below:

Solution

```
class PolicyNetwork(nn.Module):
  def __init__(self, state_dim, action_dim):
      super(PolicyNetwork, self).__init__()
      self.fc1 = nn.Linear(state_dim, 128)
      self.fc2 = nn.Linear(128, 128)
      self.fc3 = nn.Linear(128, action_dim)

  def forward(self, x):
      x = torch.relu(self.fc1(x))
      x = torch.relu(self.fc2(x))
      x = self.fc3(x)
      return Categorical(logits=x)
```

```
class ValueNetwork(nn.Module):
    def __init__(self, state_dim):
      super(ValueNetwork, self).__init__()
      self.fc1 = nn.Linear(state_dim, 128)
      self.fc2 = nn.Linear(128, 128)
      self.fc3 = nn.Linear(128, 1)

  def forward(self, x):
      x = torch.relu(self.fc1(x))
      x = torch.relu(self.fc2(x))
      x = self.fc3(x)
      return x
```

Question 1.b

Below you need to implement GAE. The function should return the advantage estimates for each state-action pair. Paste below:

Solution

```
def compute_advantages(next_value, rewards, masks, values, gamma
    =0.99, lambda_gae=0.95):
    values = values + [next_value]  # Append bootstrap value for last
    state
    advantages = 0
    returns = []  # Store computed returns

    for step in reversed(range(len(rewards))):  # Iterate in reverse
    (backward pass)
        delta = rewards[step] + gamma * values[step + 1] * masks[step
    ] - values[step]
        advantages = delta + gamma * lambda_gae * masks[step] *
    advantages
        returns.insert(0, advantages + values[step])

```

```
    return returns
```

Question 1.c

Now let;s move to the training parts. You need to implement the function below which applies the core PPO algorithm, uses the experiences collected from the environment to perform multiple epochs of updates on the policy and value networks. Paste below:

Solution

```
def ppo_update(policy_net, value_net, optimizer, ppo_epochs,
    mini_batch_size, states, actions, log_probs, returns, advantages,
    clip_param=0.2):
     for _ in range(ppo_epochs):
         for state, action, old_log_probs, return_, advantage in
    ppo_iter(mini_batch_size, states, actions, log_probs, returns,
    advantages):
             dist = policy_net(state)
             new_log_probs = dist.log_prob(action)

             ##### Code implementation here #####
             ratio = (new_log_probs-old_log_probs).exp()
             clipped_ratio = torch.clamp(ratio, 1.0-clip_param, 1.0+
    clip_param)
             actor_loss = -torch.min(ratio*advantage, clipped_ratio*
    advantage).mean()
             value = value_net(state).squeeze(1)
             critic_loss = (return_-value).pow(2).mean()
             ##### Code implementation End #####

             loss = 0.5 * critic_loss + actor_loss

             optimizer.zero_grad()
             loss.backward()
             optimizer.step()
```

Question 1.d

Finally, you should be able to run the main PPO training. You can freely adjust all hyperparameters for better performance. The provided hyperparameters, if implemented correctly, should be able to make rewards close to/higher than 400.

Note: Please try several times if you think your code is correct, the learning curves can have some variances over different runs. Present the best run you can get.

Solution


Performance Curves
PPO training: reach reward 400.0 with 46 steps
Reward
400
350
300
250
200
150
100
50
0
10
20
30
40
Step