PPO Crash Course

Fundamental Concepts & Implementation Notes

A Quick Introduction to PPO

How do we keep performance from falling off a cliff?

A Quick Introduction to PPO

- Actor critic methods are sensitive to perturbations
- Limits update to policy network
- Base the update on the ratio of new policy to old
- Have to account for goodness of state (advantage)
- Clip loss function and take lower bound with min

A Quick Introduction to PPO

- Keeps track of a fixed length trajectory of memories

- Uses multiple network updates per data sample

    - Minibatch stochastic gradient ascent

- Can also use multiple parallel actors (CPU)

Implementation Notes

- Memory indices = [0, 1, 2, …, 19]

- Batches start at multiples of batch_size [0, 5, 10, 15]

- Shuffle memories then take batch size chunks

Implementation Notes

- Two distinct networks instead of shared inputs
- Critic evaluates states (not s,a pairs)
- Actor decides what to do based on current state
  - Network outputs probs (softmax) for a distribution
  - Exploration due to nature of distribution

Implementation Notes

- Memory is fixed to length T (say, 20) steps
- Track states, actions, rewards, dones, values, log probs
- Shuffle memories and sample batches (5)
- Perform 4 epochs of updates on each batch

Implementation Notes

- Update rule for Actor is complex

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right].$$

- Based on ratio of new policy to old (can use logs)
- Also takes into account the advantage (A_t)
- Can still be large!

Implementation Notes

- Define epsilon (~0.2) for clip/min operations

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big]$$

- This serves as a pessimistic lower bound to the loss

- Smaller loss, smaller gradient, smaller update

Implementation Notes

- Which leads us to the advantage at each time step ...

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$
$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- Tells us the benefit of the new state over the old
- Lambda is a smoothing parameter (~0.95)
- Nested for loops

Implementation Notes

- Critic loss is more straight forward ...
- Return = advantage + critic value (from mem)
- L_{critic} = MSE(return – critic value (from network))

Implementation Notes

- Total loss is sum of clipped actor and critic

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

- Note that we are doing gradient ascent!

- $C_1 = 0.5$

Stuff We Won't Implement

- Coupled actor critic requires entropy term (S)
- Can also be used for continuous actions
- Won't do the multi core cpu implementation → GPU

Data Structures We Will Need

- Class for replay buffer → lists
- Class for actor network, class for critic network
- Class for agent (ties everything together)
- Main loop to train and evaluate