Algorithm

For my master thesis, I am trying to implement a side effect minimization in Proximal Policy Optimization (PPO), as described in Schulman et al. (2017) by including an impact measurement. However, I am implementing a simplified version; see Algorithm 1.

Algorithm 1 PPO-Clip

1: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots, k_{\max}$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$
4: Compute rewards-to-go $\hat{R}_t$
5: Compute advantage estimates, $\hat{A}_t$ based on the current value function V_{ϕ_k}
6: Update the policy by maximizing the PPO-CLIP objective:

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

where $\delta_{\pi_\theta} = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$, and $\pi_\theta(a_t|s_t)$ is the probability for action a_t given state s_t with the policy generated by the actor π_θ

7: Fit value function by regression by minimizing the mean-squared error:

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

8: **end for**

In step 4 the rewards-to-go is computed the following way:

$$\hat{R}_t(s_k) = \sum_{i=k}^{T} \gamma^{i-k} R(s_i),$$

where $R(s_i)$ is the reward yielded at state s_i and γ is a discount factor defined in the range $\gamma \in (0, 1]$.

In step 5, we compute the advantage the following way:

$$\hat{A}_t(s_i) = \hat{R}_t(s_i) - V_{\phi_k}(s_i),$$

where $V_{\phi_k}(s_i)$ is the critics estimate of $\hat{R}_t$ based on the previous runs. The idea behind using the estimated advantage is to see how the rewards of the agent's current run compare to what the critic has seen and learned. Thus if the agent receives a higher reward than the critic's estimate, we know that the agent has found a new, better solution, and we will reinforce that behavior by making the actions made more probable in the future. Contrary, if the agent receives a worse reward, we will make those actions less likely.

Then in step 6 of the algorithm, we will update the weights according to how likely the action is for the new policy π_θ compared to the old policy π_θ times the advantage. However, a too large update can render the policy useless. Thus we use the minimum of the first term and the clipped version to reduce the amount we update the parameters in the actor's network. We define the clip function as the following:

$$\text{clip}(\delta, 1-\epsilon, 1+\epsilon) = \min(\max(\delta_{\pi_\theta}, 1-\epsilon), 1+\epsilon).$$

Using the clip function ensures that we do not make an update that changes the policy too much and would render it useless.

Impact measurement
To add an impact measurement, I use an approach similar to the one used in Krakovna et al. (2020), where the authors propose including an auxiliary reward whose purpose is to represent the agent's ability to complete future tasks. We add this auxiliary reward to the normal reward generated in each state $r + \beta r_{aux}$, where β is a scaling parameter signifying how much the agent should value future tasks. The idea is that keeping future tasks possible will limit side effects that would prevent the agent from completing them.

The auxiliary reward is in the paper defined as the discounted reward yielded when simulating the agent performing an auxiliary task when starting at the current position. This auxiliary task is drawn randomly from all possible future tasks. However, to avoid simulating this at every time step, my idea is to estimate this discounted reward for auxiliary tasks instead.

My approach is first to run the algorithm and save the critic from that run. Then in a second run, use the first pre-trained critic to estimate the future reward for the auxiliary tasks for each state. Then add the estimates to the rewards-to-go $\hat{R}_t$. This approach would allow for a quicker computation since we are estimating instead of simulating the discounted reward. Furthermore, due to the speed up, we are not required to use only one auxiliary task and can instead use all.

Issue
Initially, I had an issue getting the first critic to be general enough to generate accurate estimates of the value function in states that the critic has not encountered during the training. To solve this, I trained it on a larger distribution of initial states and later even added some augmentation to the states to get more variety. As a result, I can now train the first critic to estimate the auxiliary rewards well. However, when I apply this in the algorithm, I see no significant change in the frequency of side effects the agent makes.

So my question is: Is this a reasonable approach for implementing the future task approach in this version of the PPO algorithm? And if not, do you have any idea of how I can implement this in a better way?

Bibliography

Krakovna, V., Orseau, L., Ngo, R., Martic, M., and Legg, S. (2020). Avoiding side effects by considering future tasks. *CoRR*, abs/2010.07877.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.