Paper Critique

Anirud N, CE21B014

Course: DA7400, Fall 2024, IITM
Paper: Reward Constrained Policy Optimization
Date: 13 September 2024

Make sure your critique addresses the following points:

1. The problem the paper is trying to address
2. Key contributions of the paper
3. Proposed algorithm/framework
4. How the proposed algorithm addressed the described problem

Note: Be concise with your explanations. Unnecessary verbosity will be penalized. Please don't exceed 2 pages.

---

1 Problem Statement

The currently constrained policy optimization methods of safe RL suffer from many issues. They involve time-consuming and computationally expensive methods for choosing the hyper-parameters. The penalty coefficients are chosen by hyperparameter tuning. These coefficients are not shared across domains, i.e., the agent learned in one domain may be terrible in another domain. We need a way to not manually select penalty coefficients. The constraints require parameterisation of the policy and the propagation of the constraint violation signal over the entire trajectory, which might give issues in Q-Learning algorithms, DQN, etc, as they do not learn the parameterisation of the policy.

2 Key Contributions

This approach is a multi-objective problem. It incorporates the constraint into the reward signal. This algorithm converges almost surely, under mild assumptions. The paper also shows the implementation and results on a toy domain and then 6 robotics domains, showing faster convergence and better constraint satisfaction.

3 Proposed Algorithm/Framework

3.1 Constrained Policy Optimization and Gradients

$$\min_{\lambda \geq 0} \max_{\theta} L(\lambda, \theta) = \min_{\lambda \geq 0} \max_{\theta} \left[J_R^{\pi_\theta} - \lambda \cdot (J_C^{\pi_\theta} - \alpha) \right] \ , \tag{1}$$

$$\lambda_{k+1} = \Gamma_\lambda [\lambda_k - \eta_1(k) \nabla_\lambda L(\lambda_k, \theta_k)] \ , \tag{2}$$

$$\theta_{k+1} = \Gamma_\theta [\theta_k + \eta_2(k) \nabla_\theta L(\lambda_k, \theta_k)] \ , \tag{3}$$

Equation 1 is the objective function while 2 and 3 are the update equations.

$$\nabla_\theta L(\lambda, \theta) = \nabla_\theta \mathbb{E}^{\pi_\theta}_{s \sim \mu} \left[\log \pi(s, a; \theta) \left[R(s) - \lambda \cdot C(s) \right] \right] \ ,$$
$$\nabla_\lambda L(\lambda, \theta) = -(\mathbb{E}^{\pi_\theta}_{s \sim \mu}[C(s)] - \alpha) \ ,$$

3.2 Reward Constrained Policy Optimization and Gradients

The penalized reward functions are defined as:

$$\hat{r}(\lambda, s, a) \triangleq r(s, a) - \lambda c(s, a) \ ,$$
$$\hat{V}^{\pi}(\lambda, s) \triangleq \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \hat{r}(\lambda, s_t, a_t) | s_0 = s\right]$$
$$= \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) - \lambda c(s_t, a_t)\right) | s_0 = s\right] = V_R^{\pi}(s) - \lambda V_{C_\gamma}^{\pi}(s) \ .$$

Here, λ is computed using Monte Carlo sampling on the original constraint. So, we use a discounted penalty framework to train the actor and critic.

3.3 Overall Algorithm

RCPO employs a multi-timescale approach, where different components are updated at varying rates. On the fastest timescale, a TD-critic estimates an alternative discounted objective. On the intermediate timescale, the policy is updated using policy gradient methods. Finally, on the slowest timescale, the penalty coefficient λ is adjusted by ascending on the original constraint.

Algorithm 1 Template for an RCPO Implementation

Input: penalty $c(\cdot)$, constraint $C(\cdot)$, threshold α, learning rates $\eta_1(k) < \eta_2(k) < \eta_3(k)$
Initialize actor parameters $\theta = \theta_0$, critic parameters $v = v_0$, Lagrange multipliers $\lambda = 0$
for $k = 0, 1, \ldots$ **do**
  Initialize state $s_0 \sim \mu$
  for $t = 0, 1, \ldots, T-1$ **do**
    Sample action $a_t \sim \pi$, observe next state s_{t+1}, reward r_t and penalties c_t
    $\hat{R}_t = r_t - \lambda_k c_t + \gamma \hat{V}(\lambda, s_t; v_k)$
    Critic update: $v_{k+1} \leftarrow v_k - \eta_3(k) \left[\partial (\hat{R}_t - \hat{V}(\lambda, s_t; v_k))^2 / \partial v_k\right]$
    Actor update: $\theta_{k+1} \leftarrow \Gamma_\theta \left[\theta_k + \eta_2(k) \nabla_\theta \hat{V}(\lambda, s)\right]$
  end for
  Lagrange multiplier update: $\lambda_{k+1} \leftarrow \Gamma_\lambda \left[\lambda_k + \eta_1(k) \left(J_C^{\pi_\theta} - \alpha\right)\right]$
end for
return policy parameters θ

4 Conclusions and Results

The algorithm was implemented on a mars rover setup. For both the scenarios $\alpha = 0.01$ and $\alpha = 0.5$, RCPO showed faster convergence and lower variance. RCPO is better in terms of sample efficiency and has a stable learning curve. It was also implemented and compared in the Robotics environment settings. It was found that in these environment, except the Walker2d envi, RCPO gives a superior performance. Selecting a constant value for λ is not trivial- it gives different results for different tasks. This algorithm also faces a few drawbacks. In a few domains, the penalty coefficient is required to be larger to satisfy the constraints. But this can also lead to sub-optimal solutions in a few domains. It is also possible that the training of the agent , at the start would focus only on the constraint satisfaction and it might ignore the reward signal.