Paper Critique

Anirud N, CE21B014

Course: DA7400, Fall 2024, IITM
Paper: OptLayer - Practical Constrained Optimization for Deep Reinforcement Learning in the Real World
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

Typical robotics RL is usually limited to simulated worlds and restricted environments. Using unconstrained exploration/interaction in the real world may not always be feasible. Due to model uncertainties, transferring control policies from simulation to reality is a challenge. The difference between the simulated and the observed environments is always an issue.

2 Key Contributions

This paper proposes a method to constrain the neural network predictions to lie within the safety constraints.

- Neural network architecture with a constrained optimization layer, OptLayer. This layer gives arbitrary constraints on the actions and is fully differentiable for smooth learning under safety constraints.
- An easy-to-implement reward strategy that is compatible with existing policy optimization techniques.
- More efficient exploration and sample efficiency, demonstrated using collision avoidance on an industrial manipulator.

3 Proposed Algorithm/Framework

The formulation of TRPO can be expressed as:

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s\sim\rho_{\theta_k}, a\sim\pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s,a) \right] \tag{1}$$

$$\text{such that } \mathbb{E}_{s\sim\rho_{\theta_k}} \left[D_{KL} \left(\pi_{\theta_k}(\cdot|s) \parallel \pi_\theta(\cdot|s) \right) \right] \leq \delta_{KL} \tag{2}$$

They employ a neural network, N, an MLP that takes in state s_i and outputs action-value pairs (a_i, v_i). The constrained optimization is formulated as a quadratic program, ensuring that the predicted action a_t by the neural network satisfies the quadratic constraints. This is achieved using the constrained optimization layer, OptLayer, which computes the action a_i^* closest to the predicted action, satisfying the constraints dependent on s_i.

3.1 Quadratic Program

- **Objective matrices**: Minimize the L2 distance between the optimization variable x and the predicted action from the neural network.
- **Constant constraint matrices**: Static constraints.
- **Constraint matrices affine to the state vector**: State-dependent constraints.
- **Auxiliary state parameters**: Constraints that are not a simple function of the state.

3.2 Overall Algorithm

procedure BUILDTRAJ($\mathcal{E}, \mathcal{N}, \mathcal{O}, constrained$)
  $S, V, R \leftarrow (), ()$ ▷ States, values, rewards
  $\widetilde{A}, A^* \leftarrow (), ()$ ▷ Predicted, optimal actions
  $C \leftarrow ()$ ▷ Constraint violation costs
  $\mathbf{s} = \mathcal{E}$.reset(); $end =$ false ▷ Get initial state
  while not end **do**
    S.add($\mathbf{s}$) ▷ Store current state
    $\widetilde{\mathbf{a}}, v \leftarrow \mathcal{N}(\mathbf{s})$ ▷ Predict action, value
    $\mathbf{a}^*, c \leftarrow \mathcal{O}(\mathbf{s}, \widetilde{\mathbf{a}})$ ▷ Constrain prediction
    $\widetilde{A}$.add($\widetilde{\mathbf{a}}$); V.add(v); A^*.add($\mathbf{a}^*$); C.add(c)
    if $constrained$ **then**
      $\mathbf{s}, r, end = \mathcal{E}$.do($\mathbf{a}^*$) ▷ Respects constraints
    else
      $\mathbf{s}, r, end = \mathcal{E}$.do($\widetilde{\mathbf{a}}$) ▷ Can violate constraints
    end if
    R.add(r) ▷ Store reward
  end while
  return $S, \widetilde{A}, R, V, A^*, C$
end procedure

Figure 1: Caption

An interior point method to solve the Quadratic Program, to incorporate training in mini-batches for the neural networks.

4 Conclusions and Results

The paper shows that the stochastic control policies can be efficiently combined with constrained optimization for better, safer learning. The methods are simpler to implement. The paper also shows the effectiveness of this algorithm using a collision avoidance 3D task environment. Real-world experiments showed that the algorithm can better accommodate slight variances in the environment.
The approach shows improvement in better exploiting unsafe predictions, but it also suffers from some drawbacks. Learning policies from scratch is time-consuming, even for simpler tasks. The approach also shows that formalizing the safety constraints may be difficult due to dynamic model uncertainties.