Paper Critique

Shuvrajeet Das, DA24D402

Course: DA7400, Fall 2024, IITM
Paper: [OptLayer - Practical Constrained Optimization for Deep Reinforcement Learning in the Real World]
Date: [13-09-2024]

Make sure your critique Address 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 The problem the paper is trying to address

In real-world robotics, executing reinforcement learning (RL) policies without safety constraints is impractical. Unconstrained RL can lead to harmful actions, such as robot collisions or unsafe behavior, due to unpredictable trial-and-error exploration.

The goal is to find a solution where RL can be applied in real-world environments, ensuring that actions a_t generated by the policy π_θ are safe and respect predefined constraints. Develop a constrained optimization layer, OptLayer, that takes potentially unsafe actions from a neural network and projects them to the closest safe actions that respect the predefined constraints.

The solution ensures that:

1. Only safe actions are executed in the environment.
2. Unsafe actions are penalized during training, improving policy robustness.

2 Key contributions of the paper

The paper presents several key contributions to the field of deep reinforcement learning (RL) and constrained optimization for real-world robotic applications:

- **OptLayer:** A novel optimization layer that projects potentially unsafe actions generated by a neural network onto the set of feasible and safe actions. This ensures that only actions satisfying the safety constraints are executed.
- **Differentiability:** OptLayer is fully differentiable, allowing end-to-end training of RL policies under safety constraints without compromising the learning efficiency.
- **Safe Reinforcement Learning:** The approach guarantees safe exploration during the learning phase, preventing unsafe actions by design, which is crucial for real-world robot applications.
- **Reward Strategy:** A simple reward strategy is proposed that penalizes unsafe predictions during training, making it readily compatible with standard policy optimization methods.
- **Real-world Validation:** The effectiveness of OptLayer is demonstrated through experiments in both simulated environments and real-world robotic tasks, such as 3D reaching with obstacle avoidance.

3 Proposed algorithm/framework

Algorithm 1 Safe Reinforcement Learning with OptLayer

Require: Environment E, Neural Network Policy π_θ, OptLayer O, Number of Episodes N_{episodes}

Ensure: Safe Policy π_θ that satisfies safety constraints

for each episode $k = 1$ to N_{episodes} **do**
  Initialize state $s_0 \sim \rho_0$
  Initialize storage: State sequence $S \leftarrow ()$, Action sequence $A \leftarrow ()$, Reward sequence $R \leftarrow ()$
  for each step $t = 0$ to T **do**
    Predict action $a_t \sim \pi_\theta(s_t)$
    Compute corrected action $a_t^* \leftarrow O(s_t, a_t)$ ▷ Using OptLayer to project action onto safe set
    Execute corrected action a_t^* in environment E
    Observe reward r_t and new state s_{t+1}
    Store (s_t, a_t, r_t) into S, A, R
    if s_{t+1} is terminal **then**
      Break
    end if
  end for
  Compute constraint violation cost $c_t = \| \max(Ga_t - h, 0)\| + \|Aa_t - b\|$
  Compute discounted reward with penalty: $\tilde{r}_t \leftarrow r_t - c_t$
  Perform policy update using Trust Region Policy Optimization (TRPO):

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

  subject to:

$$\mathbb{E}_{s_t \sim \rho_{\theta_k}} \left[D_{\text{KL}} \left(\pi_{\theta_k} \| \pi_\theta \right) \right] \leq \delta_{\text{KL}}$$

end for

4 How the proposed algorithm addressed the problem

- **Safe Exploration:** By using OptLayer during both the training and execution phases, the robot can explore its environment safely, ensuring that even during policy learning, the actions it takes are guaranteed to be safe, preventing collisions or other unsafe behavior.
- **Constraint Violation Penalty:** The algorithm penalizes any unsafe action generated by the neural network during training, encouraging the network to learn a policy that inherently avoids unsafe predictions.
- **Policy Optimization:** The algorithm incorporates standard policy optimization techniques, such as Trust Region Policy Optimization (TRPO), to iteratively update the policy while ensuring that it satisfies the Kullback-Leibler (KL) constraint:

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

  subject to:

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