Robot Path Planning via Distributionally Robust Deep Reinforcement Learning

Cem Alpturk[1], Venkatraman Renganathan[2], and Anders Rantzer[3]

[1]Student: ce5368al-s@student.lu.se
[2]Supervisor: venkat@control.lth.se
[2]Examiner: anders.rantzer@control.lth.se

January 2022 - June 2022

1 Introduction

2 Problem Definition

A robot that has discrete LTI dynamics can move on a 2D environment. The goal is to find a path from a starting position $p_0 \in \mathbb{R}^2$ to a goal position $p_g \in \mathbb{R}^2$ while avoiding obstacles that are located in the environment by using a set of inputs $u \in \mathcal{U} \subseteq \mathbb{R}^2$. The robot dynamics are modeled by setting $x = \{p, \dot{p}\} \in \mathbb{R}^4$.

$$\begin{aligned} &x_{t+1} = Ax_t + Bu_t + w_t \\ &x_0 \sim \mathcal{N}(\bar{x_0}, \Sigma_{x_0}) \\ &w_t \sim \mathbb{P}_w \in \mathcal{P}^w := \{\mathbb{P}_w | W_p(\hat{\mathbb{P}}_w, \mathbb{P}_w) < \epsilon\} \end{aligned} \tag{1}$$

The distribution of w_t comes from an ambiguity set $\mathcal{P}^w$ which is the set of all distributions that have a Wasserstein distance to a nominal distribution $\hat{\mathbb{P}}_w$, less than ϵ. This addition is the main idea of distributionally robust optimization. The nominal distribution $\mathbb{P}_w$ can be an initial guess or an empirical distribution that is obtained by collecting data. In summary, the distribution of w is not known.

The total configuration space of the robot is denoted as $\mathcal{Q}$. The obstacles in the environment are static, bounded convex polygons. The set of obstacles are defined as,

$$\mathcal{Q}_{obs} = \bigcup_{i=1}^{M} \mathcal{Q}_{obs}^{(i)} \tag{2}$$

The goal of this problem is to reach an area $\mathcal{Q}_{goal}$, that surrounds the goal position p_g without colliding with the obstacles,

$$\begin{aligned} P(q_N \notin \mathcal{Q}_{goal}) < \delta_1 \\ P(q_t \in \mathcal{Q}_{obs}) < \delta_2 \end{aligned} \tag{3}$$

Aside from the disturbance w, the robot cannot observe its states directly. The observations are done by sensors that are placed on the robot, which are modeled like lasers that calculate the distance to the nearest obstacle in a certain direction. This causes the observation to be a nonlinear function of the state x_t, the position of the obstacles $\mathcal{Q}_{obs}$.

3 Markov Decision Process

A Markov Decision Process is characterized by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$. $\mathcal{S} \subset \mathbb{R}^n$ is the state space, $\mathcal{A} \subset \mathbb{R}^m$ is the finite set called the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition probability which defines the probability distribution over the next states, given the current state and action.

At a given time t, the agent's state $s_t \in \mathcal{S}$ contains the measurements from the distance sensors $d_t^{(i)}$, the position of the obstacles $q_{obs}^{(j)}$ and the position of the goal p_g.

$$s_t = \{d_t^{(i)}, q_{obs}^{(j)}, p_g\}, i = 1, ..., k_1, j = 1, ..., k_2 \tag{4}$$

An action $a_t \in \mathcal{A}$ performed by the agent at state $s_t \in \mathcal{S}$, will transition the agent in to a new state $s_{t+1} \in \mathcal{S}$ with the probability $\mathcal{P}(s_{t+1}|s_t, a_t)$. After the transition, the agent will receive a reward $\mathcal{R}(s_t, a_t)$ from the environment. The actions are chosen based on a policy $\pi : \mathcal{S} \to \mathcal{A}$, $\pi(s_t) = a_t$. The collection of all possible policies are denoted by Π. A sequence

$$\tau_t^{(\pi)} = (s_t, \pi(s_t), s_{t+1}, \pi(s_{t+1}), ..., s_{N-1}, \pi(s_{N_1}), s_N) \tag{5}$$

with a terminal state s_N is called a π sample path. The discounted cumulative reward for this sample path is given by

$$R^{\pi}(\tau_t) = \sum_{n=t}^{N-1} \gamma^{n-t} \mathcal{R}(s_n, \pi(s_n)) \tag{6}$$

where $\gamma \in [0, 1]$ is the discount factor. Due to the randomness in the state transitions, an expected value can be calculated for the returns starting from state s,

$$V^{\pi}(s) = \mathbb{E}[R^{\pi}(\tau_t|s_t = s)] \tag{7}$$

The goal is to find a policy π^* such that,

$$\pi^* = arg \max_{\pi} V^{\pi}(s_t) \tag{8}$$

4 Q Learning

For a policy π, the Q-value for a state action pair is defined as

$$Q^{\pi}(s_t, a_t) = \int_{\mathcal{S}} \left(\mathcal{R}(s_t, a_t) + \gamma V^{\pi}(s_{t+1})\right) \mathcal{P}^{\pi}(s_{t+1}|s_t, a_t) ds_{t+1} \tag{9}$$

The Q-value is the expected return obtained by applying action a at state s and following the policy π. The objective of the Q-learning algorithm is to find the Q-values for the optimal policy π^*.

$$Q^{\pi^*}(s_t, a_t) = \int_{\mathcal{S}} \left(\mathcal{R}(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^{\pi^*}(s_{t+1}, a_{t+1})\right) \mathcal{P}^{\pi}(s_{t+1}|s_t, a_t) ds_{t+1} \tag{10}$$

The relationship between state-action pairs and the Q-values are learned via a non-linear function $f : \mathbb{R}^n \to \mathbb{R}^m$, where n and m are the dimensions of the state and action space. The optimal Q-function satisfies,

$$Q^*(s, a) = \mathbb{E}_{s'} \left[\mathcal{R}(s, a) + \gamma \max_{a'} Q^*(s', a')|s, a\right] \tag{11}$$