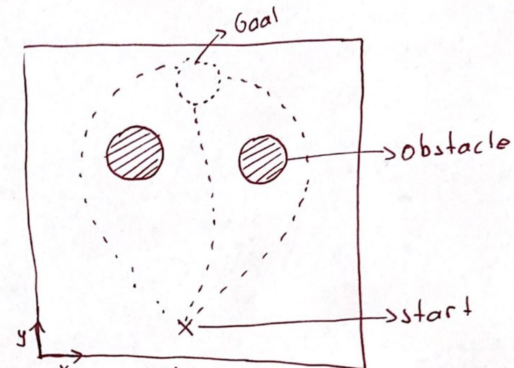
Problem Formulation

=> For a given environment with obstacles, find a safe path between a starting point and a goal position without any collisions.



Map

Robot Dynamics

$$x_{k+1} = Ax_k + Bu_k + w_k$$

$x_k \in \mathcal{X} \subset \mathbb{R}^2$: $[x, y]$ position of the robot

$u_k \in A \subset \mathbb{R}^2$: Input

$w_k \in \mathbb{R}^2$: Process disturbance with unknown distribution P_{true}

Markov Decision Problem

$\langle S, A, r, P \rangle$

$S \subset \mathbb{R}^n$: Bounded cont. state space $\Rightarrow s_k = [\underbrace{x_k}_{\text{Position}}, \underbrace{O_k^{(i)}}_{\text{Distance to obstacle centroids}}, \underbrace{g}_{\text{goal position}}]$

$A \subset \mathbb{R}^2$: Finite action space

$r : S \times A \times S \to \mathbb{R}$: Reward function

$P : S \times A \to \Delta_S$: Transition probability $P(s_{k+1} | s_k, a_k)$

$\pi : S \to \Delta_A$: Policy, $P(a_k | s_k)$

Trajectory: $\tau^\pi = (s_0, a_0, s_1, a_1, \dots, s_N)$

Discounted Returns: $R(\tau^\pi) = \sum_{i=0}^{N-1} \gamma^i r(s_i, a_i, s_{i+1})$

Value Function: $V^\pi(s) = \mathbb{E}_{\substack{a_k \sim \pi \\ s_{k+1} \sim P}} \left[R(\tau^\pi | s_0 = s) \right]$

Q-Function: $Q(s, a) = \mathbb{E}_{\substack{a_k \sim \pi \\ s_{k+1} \sim P}} \left[R(\tau^\pi | s_0 = s, a_0 = a) \right]$

The goal

$$\pi^* = \arg\max_{\pi \in \Pi} V^\pi(s), \quad \forall s \in S$$

Distributionally Robust Optimization

$h(x,z)$: objective function

$x \in \mathcal{X}$: controlled variable

$z \in Z$: stochastic variable,

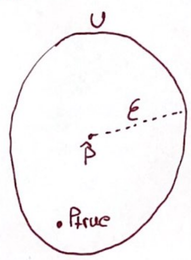
DRO: $\inf_{x\in\mathcal{X}} \underbrace{\sup_{P\in U} E_{z\sim P}[h(x,z)]}_{\text{worst-case scenario}}$

U: ambiguity set

$U := \{P \mid W_p(\hat{P}, P) \leq \varepsilon\}$

select ε such that

$P(P_{true} \in U) \geq 1-\beta$



Wasserstein Distance

= Earth movers distance

$$W_p(Q, Q') = \left(\inf_{\pi\in\Pi(Q,Q')} \int_{Z\times Z} \|z - z'\|^p \pi(dz, dz')\right)^{\frac{1}{p}}$$

=> Min energy required to convert Q into Q'

$\Pi(Q,Q')$: set of all joint prob. distributions with marginals Q, Q'

Primal: $v_P = \sup_{P\in U} \int_Z h(z)\, dP(z) \ : W_p(P, \hat{P}) \leq \varepsilon$

=> The primal problem has a strong dual

Dual: $v_D = \inf_{\lambda\geq 0} \left\{ \lambda\varepsilon^p - \int_Z \inf_{z\in Z} [\lambda \|z - z_0\|^p - h(z)]\, d\hat{P}(z_0) \right\}$

$v_P = v_D$

Discrete Empirical $\hat{P}_N$

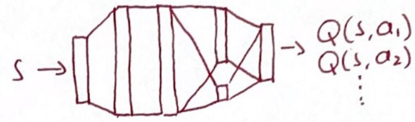
=> N samples of z

$$\hat{P}_N \triangleq \frac{1}{N}\sum_{i=1}^{N} \delta_{z_i}(z)$$

$$v_P = v_D = \min_{\lambda\geq 0} \left\{ \lambda\varepsilon^p + \frac{1}{N}\sum_{i=1}^{N} \sup_{z\in Z} [h(z) - \lambda\|z - z_i\|^p] \right\}$$

Distributionally Robust Q-Learning

Q-function: $Q(s,a) = E\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k, s_{k+1}) \mid s_0 = s, a_0 = a\right]$



Bellman Operator $\mathcal{T}$

$$\mathcal{T} Q(s,a) = E_{s' \sim p}\left[r(s,a,s') + \gamma \underbrace{E_{a' \sim \pi}\left[Q(s',a')\right]}_{\sum_{a' \in A} \pi(a'|s') Q(s',a')}\right]$$

=> Contraction

$$\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{a' \in A} e^{Q(s,a')}}$$

Distributionally Robust Bellman Operator $\hat{\mathcal{T}}$

$$\hat{\mathcal{T}} Q(s,a) = \inf_{p \in U} E_{s' \sim p}\Big[\underbrace{r(s,a,s') + \gamma E_{a' \sim \pi}\left[Q(s',a')\right]}_{h(s,a,s') \Rightarrow h(s')}\Big]$$

=> worst case expected discounted returns

$h(s') \Rightarrow$ non-linear, non-convex

=> $Q(s,a) \rightarrow$ approximated by a neural network

TD-Learning

=> store transition (s_k, a_k, s_{k+1}, r_k)

td-error => $\delta = \underbrace{r_k + \gamma \max_a Q(s_{k+1}, a)}_{\text{targets}} - Q(s_k, a_k)$

$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \delta$

Distributionally Robust target $= \inf_{p \in U} E_{s' \sim p}\left[h(s_k, a_k, s')\right]$

=> Double DQN

=> Duelling layer

=> Prioritized experience replay

Solving DRO

Lipschitz Approximation

- $$\sup_{P \in U} E_{s' \sim P}[-h(s')]$$

$$Lip(\phi) = \sup_{\xi \neq \xi'} \frac{|\phi(\xi) - \phi(\xi')|}{\|\xi - \xi'\|}$$

$$\Rightarrow \sup_{P \in U} E_{s' \sim P}[-h(s')] \leq E_{s' \sim \hat{P}_N}[-h(s)] + \epsilon \, Lip(-h)$$

$\Rightarrow$ $Lip(h)$ can be estimated by using LipSDP in python

↳ neural network lipschitz const. estimation

$\Rightarrow$ $Lip(h)$ can be estimated when the target network is aligned.

Robust Program Approximation

$$V_P = V_D = \min_{\lambda \geq 0} \left\{ \lambda \epsilon^p + \frac{1}{N} \sum_{i=1}^{N} \sup_{z \in Z} \left[-h(z) - \lambda \|z - z_i\|^p \right] \right\}$$

for $K > 0$

$$V_K := \sup_{(z^{ik})_{i,k} \in M_K} \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} -h(z^{ik})$$

$$M_K := \left\{ (z^{ik})_{i,k} : \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \|z^{ik} - z^i\|^p \leq \epsilon^p, \; z^{ik} \in Z, \; \forall i,k \right\}$$

$$V_K \uparrow \sup_{P \in U} E_{s' \sim P}[-h(s')] \text{ as } K \to \infty$$

$\Rightarrow$ Local optimums can be found

$\Rightarrow$ Must solve for each transition in memory

$\Rightarrow$ Will be slow