Approximate Robust Linear Programming

Brendan J. Crowe

May 2021

1 Backround

In the Inverse Reinforcement Learning (IRL) setting an optimizer is trying to solve a Markov Decision Process (MDP) where there is an unknown or ambiguous reward function. An MDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P_0, T(a, s))$. In the Reinforcement Learning (RL) for a tabular (fully represented) MDP the objective is to find a policy $\pi^\star$ that maximizes the expected discounted reward $\sum_{s \in \mathcal{S}} \rho(s, \pi^\star(s))$. However in many cases there is no tabular representation for the MDP causing the need for approximations. An additional concern comes when there is ambiguity in the reward function, which is where IRL comes in. In the next sections we discuss how one can solve these problems.

2 ALP

When an MDP has no complete tabular representation, but the reward function $r(a, s) \quad r \in \mathcal{R}$ is know, one can approximate the value function $v(s)$, or q-functions $q(a, s)$ using state or state-action features respectively, that encode information about the given state action pair, as well as its relationship to other state-action pairs. We can approximate $v(\mathcal{S}) \approx \Phi w$, where Φ are the state features, and w are weights to be learned. Now the MDP can be solved with the following linear program, as proposed in Approximate Linear Programming (ALP)[**10.1287/opre.51.6.850.24925**]

$$\begin{aligned} \min_{w} \quad & c^T \Phi w \\ \text{s.t.} \quad & (\mathbb{I} - \gamma P_a)\Phi \geq r_a \quad a = 1, \ldots, A \end{aligned} \qquad [2.1]$$

Here, $c = P_0$ is the distributions over initial state-actions pairs, γ is the discount rate, P_a, r_a are the transition probabilities and reward for each action a, and $A = |\mathcal{A}|$ where $\mathcal{A}$.

3 BROIL

When the reward function is unknown, but we have a tabular MDP, we want to find a $r^\star \in \mathcal{R}$ that is feasible for our MDP, and induces a good policy $\pi^\star$ on average. This is IRL, and it too can be posed as an optimization problem. Bayesian Robust Optimization for Imitation Learning (BROIL)[**brown2020bayesian**]. As simplified version of the linear program solved in BROIL is as follows

$$\begin{aligned} \max_{u \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \min_{r \in \mathcal{R}} \quad & r^T u \\ \text{s. t.} \quad & Au = c \\ & u \geq 0 \end{aligned} \tag*{[3.1]}$$

Here

$$A = \begin{bmatrix} \mathbb{I} - \gamma P_a \\ \vdots \\ \mathbb{I} - \gamma P_A \end{bmatrix}$$

u is the state-action occupancy frequencies

4 ARLP

We want to extend BROIL to be able to deal with none tabular MDPs, and thus we want to approximate the state-actions pairs using features as in ALP. That gives us the following problem.

$$\begin{aligned} \max_{u} \min_{i=1,\ldots,n} \quad & \tilde{r}_i^T \Phi^T u \\ \text{s. t.} \quad & \Phi^T A u = \Phi^T c \\ & u \geq 0 \end{aligned} \tag*{[4.1]}$$

Where $\tilde{r}_i$ are some reward features, each corresponding to possible reward function.

The corresponding linear program would be

$$\begin{aligned} \max_{u,z} \quad & z \\ \text{s. t.} \quad & z \leq \tilde{r}_i^T \Phi^T u \quad i = 1, \ldots, n \\ & \Phi^T A u = \Phi^T c \\ & u \geq 0 \end{aligned} \tag*{[4.2]}$$

Or its dual

$$\begin{aligned} \min_{w,\xi} \quad & c^T \Phi w \\ \text{s. t.} \quad & A \Phi w \geq \hat{\Phi} \tilde{R} \xi \\ & \mathbb{1}^T \xi = 1 \\ & \xi \geq 0 \end{aligned} \tag*{[4.3]}$$

Here

$$\hat{\Phi} = \begin{bmatrix} \Phi_1 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & \\ 0 & & \ddots & & \vdots \\ \vdots & & & & \\ 0 & & \cdots & & \Phi_A \end{bmatrix}$$

and

$$\tilde{R} = \begin{bmatrix} \tilde{r_i} & \cdots & \tilde{r_A} \end{bmatrix}$$

Where all $\Phi_a = \Phi$
ξ is a weight for each reward function represented by $\hat{\Phi}\tilde{R}$

5 Result

Do to the length of the write-up, I have omitted extensive discussion of the results. I have shown in some small examples that this method is capable to solving MDPs satisfactorily. More experiments are need to verify its true efficacy.