Statistical Techniques in Robotics (16-831, S22) Lecture #23 (Wednesday, April 13)

Linear Programming IRL

Lecturer: Kris Kitani *Scribes: Xiaofeng Guo*

1 Review

In last lecture, we introduced the concept of Imitation Learning. In this lecture, we will introduce one Inversve Reinforcement Learning (IRL) method: Linear Programming. In this section, we will review the concept of Imitation Learning.

1.1 Imitation Learning Overview

Different with normal Reinforcement Learning, which learns everything from sketch and takes a long time to learn, the main idea of Imitation Learning is to learn a policy from demonstrated expert behavior. Imitation Learning is found to be useful for robot learning especially when the state space is too large and would take too long to learn. However, expert demonstrations can speed up the learning process. Another common case is that the reward function is too complex to define, making the normal Reinforcement Learning method to be struggle. However, it is easier to just show examples for completing the tasks, where the Imitation Learning can work.

1.2 Problem Formulation

The Imitation Learning(IL) problem can be formed as:

$$\mathcal{A}_{IL} : \langle \mathcal{D}^*, \pi^*, \mathcal{T}, R \rangle \rightarrow \pi$$

$\mathcal{A}_{IL}$ denotes the Imitation learning algorithm. $\mathcal{D}^*$ denotes the expert demonstrations, which is a set of trajectories $\{\zeta_n\}_{n=1}^N$ sampled from the optical policy π^* interacting with the environment $\mathcal{E}$. π^* denotes the optimal policy or "oracle". However, we do not get the entire optimal policy during learning. What we can really get is the samples from the optimal policy π^*. $\mathcal{T}$ denotes the dynamics function of the environment $\mathcal{E}$. $\mathcal{R}$ denotes the reward function. π denotes the estimated policy.

1.3 Different types of Imitation Learning

Given the formulation of the Imitation Learning problem, based on different assumptions, there are 3 types of Imitation Learning (IL): Passive IL, Active IL, and Interactive IL. The difference between them are show in Table 1.

	Passive IL	Active IL	Interactive IL
Demonstrations $\mathcal{D}$	yes	yes	optional
Environment $\mathcal{E}$	no	yes	yes
Oracle π^*	no	no	yes
Dynamics $\mathcal{T}$	no	optional	optional
Reward $\mathcal{R}$	no	optional	optional

Table 1: Different Imitation Learning methods

In Passive IL, which is commonly called behavior cloning, we can only get the expert demonstrations and cannot get the access to the environment, oracle, dynamics or reward functions. It is very similar to a supervised learning problem, where we optimized the estimate policy so that it is close to the expert demonstrations. However, it is easy to have the covariate shift problem. It only models short-term (one-shot) behavior and cannot model long-term behavior well.

In Active IL, we get the expert demonstrations as well as the access to the environment $\mathcal{E}$. It is optional to get the access to the dynamic function or the reward function of the environment. It is like in addition to the passive IL, we can also do some reinforcement learning in the environment to improve the estimate policy.

In the Interactive IL, we are optional to get the expert demonstrations. Instead of that, we are given the access to the environment and the oracle, which means that we can ask for the oracle feedback when we are acting in the environment.

1.4 Inverse Reinforcement Learning

Sometimes just mimicking expert demonstrations is not helpful because we can't generalize to new environments. In addition to mimicking expert demonstrations, understanding why we should take such an action (reward function) could help us to generalize our learning better, which leads to the Inverse Reinforcement Learning. It is a type of Active IL, and its goal is to learn the reward function R from expert demonstrations. With the learned reward function, we can take Reinforcement Learning to train a policy. More importantly, the algorithm can be transferred to a new similar environment.

2 Summary

2.1 Linear Programming IRL

Here we discuss three Inverse Refinforcement Learning (IRL) scenarios:

1. IRL for finite state spaces;
2. IRL for large state spaces;
3. IRL from sampled trajectories.

The solution of each scenario involves solving a linear program (LP) to obtain the reward function.

2.1.1 IRL for Finite State Spaces

First we make some assumptions here:

1. State space is finite;
2. Transition model is known;
3. Complete policy is known.

Recall that the goal of IRL is to find a reward function such that the policy is optimal. To do that, we define the objective function to be:

$$\max_{\mathbf{R}}\{\sum_{s} Q^{\pi}(s,a^*) - \max_{a\neq a^*} Q^{\pi}(s,a)\},$$

where $Q^{\pi}(s,a^*)$ denotes the cumulative reward of taking the best action and $\max_{a\neq a^*} Q^{\pi}(s,a)$ denotes the cumulative reward of the second best action. Therefore, here we are trying to maximize the cumulative reward difference between the best action and the second best action. In order to bound the reward function, we furthermore add a regularization term to make the objective function be:

$$\max_{\mathbf{R}}\{\sum_{s} Q^{\pi}(s,a^*) - \max_{a\neq a^*} Q^{\pi}(s,a)\} - \lambda||\mathbf{R}||_1 \tag{1}$$

We will show that this is a linear equation in rewards R. Recall that we have

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s'} p(s'|s,a)V^{\pi}(s'), \quad \forall s,a$$

If we express it in matrix form, we have:

$$\mathbf{Q}^{\pi}(a) = (\mathbf{R} + \gamma \mathbf{P}_a \mathbf{V}^{\pi}), \quad \forall a$$

where all the equations for some action expressed as a matrix.

Likewise, we could express the V function:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} p(s'|s,a=\pi(s))V^{\pi}(s'), \quad \forall s$$

in matrix form:

$$\begin{aligned} \mathbf{V}_{\pi} &= \mathbf{R} + \gamma \mathbf{P}_{a^*}\mathbf{V}_{\pi} \\ \rightarrow \mathbf{V}_{\pi} &= (\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R} \end{aligned} \tag{2}$$

Therefore, we could write the difference in Q-values in Equation 1 in matrix form.

$$\begin{aligned}\mathbf{Q}^{\pi}(a^*) - \mathbf{Q}^{\pi}(a) &= (\mathbf{R} + \gamma \mathbf{P}_{a^*})\mathbf{V}^{\pi}) - (\mathbf{R} + \gamma \mathbf{P}_a)\mathbf{V}^{\pi}) \\ &= \gamma(\mathbf{P}_{a^*} - \mathbf{P}_a)\mathbf{V}^{\pi} \\ &= \gamma(\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R}\end{aligned} \tag{3}$$

Now we can express the objective function in matrix form:

$$\arg\max_{\mathbf{R}}(\sum_{s} \min_{a}\{(\mathbf{P}_{a^*}(s) - \mathbf{P}_a(s))(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R}\}) - \gamma||\mathbf{R}||_1$$

According to [1], we add one more constraint to the problem. Based on the theory that the future payoff of the best action is greater than that of another action, we have:

$$\sum_{s'} P(s'|s, a^*)V^{\pi}(s') \geq \sum_{s'} P(s'|s, a)V^{\pi}(s'), \quad \forall s, a$$

By writing it in matrix form, we have:

$$\mathbf{P}_{a*}\mathbf{V}_{\pi} \succeq \mathbf{P}_a\mathbf{V}_{\pi}, \quad \forall a$$

where "$\succeq$" denote "element-wise inequality". Bring Equation 2 into it, we get:

$$\begin{aligned}&\mathbf{P}_{a^*}(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R} \succeq \mathbf{P}_a(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R} \\ \rightarrow &(\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R} \succeq \mathbf{0}, \quad \forall a \neq a^*\end{aligned}$$

Therefore, for the finite state spaces scenario, we get the Linear Program Formulation:

$$\begin{aligned}\hat{\mathbf{R}} = \quad & \arg\max_{\mathbf{R}}(\sum_{s} \min_{a}\{(\mathbf{P}_{a^*}(s) - \mathbf{P}_a(s))(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R}\}) - \gamma||\mathbf{R}||_1 \\ \text{s.t.} \quad & (\mathbf{P}_{a^*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a^*})^{-1}\mathbf{R} \succeq \mathbf{0}, \quad \forall a \neq a^* \\ & |R(s)| \leq R_{max}, \quad \forall s \in S\end{aligned}$$

We can solve it with linear program routine and we get the reward function R. However, when the state space is large, the linear programming problem is too big to solve. In that case, we need to change our method.

2.1.2 IRL for Large State Spaces

In this scenario, we get rid of the assumption that the state space is finite. It can be large or even infinite. We change the assumptions to be:

1. State space is large or infinite;

2. Transition model is known;

3. Complete policy is known.

In this case, the reward function will be too big to store as a table and we need to use an approximation function to estimate it. Here we use a linear function of some features of the state to approximate the reward function.

$$R_\theta(s) = \theta_1\phi_1(s) + ... + \theta_d\phi_d(s) = \boldsymbol{\theta}^T\boldsymbol{\phi}(s)$$

where $\boldsymbol{\theta}^T$ is the weight vector and $\boldsymbol{\phi}(s)$ are the features of a state. In this case, the value function is a linear function of expected state features $\boldsymbol{\mu}^\pi(s)$.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_p[\sum_{t=0}^{\infty}\gamma^t r(s^{(t)})|s^{(0)} = s] \\ &= \mathbb{E}_p[\sum_{t=0}^{\infty}\gamma^t \boldsymbol{\theta}\cdot\boldsymbol{\phi}(s^{(t)})|s^{(0)} = s] \\ &= \boldsymbol{\theta}\cdot\mathbb{E}_p[\sum_{t=0}^{\infty}\gamma^t \boldsymbol{\phi}(s^{(t)})|s^{(0)} = s] \\ &= \boldsymbol{\theta}\cdot\boldsymbol{\mu}^\pi(s) \\ \rightarrow V^\pi(s) &= \boldsymbol{\theta}^T\boldsymbol{\mu}^\pi(s) \end{aligned} \tag{4}$$

Based on the theory that the future payoff of the best action is greater than that of another action, we have

$$\sum_{s'} P(s'|s,a^*)V^\pi(s') \geq \sum_{s'} P(s'|s,a)V^\pi(s') \quad \forall s,a$$

We would like to make this difference as big as possible by changing the function.

$$\max_\theta(E_{s'\sim p(s,a^*)}[V^\pi(s')] - E_{s'\sim p(s,a)}[V^\pi(s')]) \quad \forall s,a \tag{5}$$

Recall that there may be infinite number of states in this scenario. We tackle it by only optimizing a finite subset of states.

$$s \in S_0 \in \mathbf{S}$$

We also don't need to optimize for all actions. Actually, comparing to the second best action is good enough. Then we change Equation 5 to:

$$\max_\theta \min_a(\mathbb{E}_{p(s,a^*)}[V^\pi(s)] - \mathbb{E}_{p(s,a)}[V^\pi(s)]) \quad \forall s \in S_0$$

Based on Equation 4, we know it is a linear function of reward parameters.

We add one constraint to constraint the reward values: $|\theta_d| \leq 1, \forall d$. Then we get the final IRL-LP optimization objective for large state spaces scenario.

$$\begin{aligned} &\max_{\theta} \sum_{s \in S_0} \min_{a} (E_{s' \sim p(s,a^*)}[V^{\pi}(s')] - E_{s' \sim p(s,a)}[V^{\pi}(s')]) \\ &\text{s.t.} \quad |\theta_d| \leq 1 \quad \forall d \end{aligned}$$

We can solve it with linear program routine and we get the parametric reward function.

In [1], they made a small modification and put extra penalty on negative values.

$$\begin{aligned} &\max_{\theta} \sum_{s \in S_0} \min_{a} g(E_{s' \sim p(s,a^*)}[V^{\pi}(s')] - E_{s' \sim p(s,a)}[V^{\pi}(s')]) \\ &\text{s.t.} \quad |\theta_d| \leq 1 \quad \forall d \end{aligned}$$

where

$$g(x) = \begin{cases} x & x \geq 0 \\ 2x & \text{otherwise} \end{cases}$$

2.1.3 IRL from sampled trajectories

In the above two scenarios, we make the assumptions that the transition model and the complete policy are both known, which is usually not true. Here we get rid of this assumptions and suppose we only have access to expert trajectories

$$\mathcal{D} = \{\zeta_m\}_{m=1}^{M}$$

where

$$\zeta = \{s_0, a_0, s_1, a_1, ..., s_T, a_T\}$$

In this case, we no longer has the access to V^{π}. Instead, we need to get an estimate of V from a trajectory so that

$$\hat{V}^{\pi^*} \approx V^{\pi^*}$$

Here we use Monte-Carlo estimation. Based on Equation 4, we can estimate $V^{\pi}(s)$ by estimating $\boldsymbol{\mu^{\pi}(s)}$. By Treating M expert trajectories as random (Monte Carlo) samples We have

$$\begin{aligned} \boldsymbol{\mu^{\pi}(s)} &= \mathbb{E}_p[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(s^{(t)})] \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{T_m} \gamma^t \boldsymbol{\phi}(s_m^{(t)}) \end{aligned}$$

Then we get the estimate of value function using trajectories.

The objective function we want to optimize is defined as:

$$\max_{\theta}\{\hat{V}^{\pi^*}(s) - \hat{V}^{\pi}(s)\}$$

To bound the reward parameter, we add a constraint $|\theta_d| \leq 1, \forall d$. To get the other policies, we can use Reinforcement Learning. Then we get the Linear Program formulation for IRL from sampled trajectories scenario.

$$\begin{aligned}\hat{\theta} = \arg\max_{\theta}\{&\sum_{n} \hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_n}(s_0)\} \\ \text{s.t.} \quad &|\theta_d| \leq 1 \quad \forall d\end{aligned}$$

The whole algorithm is shown as Algorithm 1.

Algorithm 1 LP-IRL($\pi^{(0)}$)

1: **for** $t = 1, ..., T$ **do**
2: $\quad \hat{V}^{\pi^*} = \text{MC-Prediction}(\pi^*, \theta^{(t)})$
3: $\quad \hat{V}^{\pi^t}, \pi^{(t)} = \text{MC-Control}(\pi^{(t-1)}, \theta^{(t)})$
4: $\quad \theta^{(t+1)} = \text{LinearProg}(\hat{V}^{\pi^*}, \{\hat{V}^{\pi^{(n)}}\}_{n=1}^{t})$
5: **end for**
6: **return** θ

To conclude, here we have discussed how Inverse Reinforcement Learning can be solved by Linear Programming in different scenarios.

References

[1] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.