Linear Continuous Inverse Reinforcement Learning

Edited by Tianyang

Problem Definition

For any given Policy, the Probability Distribution of all the Trajectories induced by this Policy is determined. The Density Function of the Distribution is defined based on Max Entropy Assumption. Note that Reward R is determined for a given Policy.

$$\mathbb{P}(\tau) := ce^{\mathbb{R}(\tau)}$$

Decomposable Trajectory Assumption and Linear Features Assumption for each States in the Trajectory. Note that Theta is determined for a given Policy, and Feature Function Phi is universal for any Policies.

$$\mathbb{R}(\tau) := \sum_{s \in \tau} \theta^{\mathbb{T}} \phi(s)$$

$$\Rightarrow \mathbb{P}(\tau|\theta) = ce^{\theta^{\mathbb{T}} \sum \phi(s)} := z(\theta)^{-1} e^{\theta^{\mathbb{T}} \phi(\tau)}$$

$$\mathbf{s.t.} \int_{\tau \sim \pi} \mathbb{P}(\tau) \mathrm{d}\tau = 1$$

$$\Rightarrow z(\theta) = \int_{\tau \sim \pi} e^{\theta^{\mathbb{T}} \phi(\tau)} \mathrm{d}\tau$$

Observation of Expert Trajectories:

$$\mathcal{D} := \{\tau_1, .., \tau_n\}$$

note that this is different from

$$\{\tau | \tau \sim \pi\}$$

which depends on Theta.

Likelihood:

$$\mathbb{L}(\theta) = \Pi_{\tau_i \in \mathcal{D}} \mathbb{P}(\tau_i|\theta)$$

Objective: maximize this likelihood by choosing a proper Theta, which corresponds to / induces a Policy Pi.

Maximize Log Likelihood by Gradient Ascent

Log Likelihood:

$$\Rightarrow \log\mathbb{L}(\theta) = \sum_{\tau_i \in \mathcal{D}} \{\theta^{\mathbb{T}} \phi(\tau_i) - \log z(\theta)\} \quad \uparrow$$

whose physical meaning: choose Theta to maximize negative energy for trajectories in expert dataset, but minimize a regularize z.

Gradient Ascent:

$$\Rightarrow \frac{\partial \log\mathbb{L}(\theta)}{\partial \theta} = \sum_{\tau_i \in \mathcal{D}} \{\phi(\tau_i) - \frac{1}{z(\theta)} \frac{\partial z(\theta)}{\partial \theta}\}$$

$$z(\theta) = \int_{\tau \sim \pi} e^{\theta^{\mathbb{T}} \phi(\tau)} \mathrm{d}\tau \Rightarrow \frac{\partial z(\theta)}{\partial \theta} = \int_{\tau \sim \pi} \phi(\tau) e^{\theta^{\mathbb{T}} \phi(\tau)} \mathrm{d}\tau$$

$$\Rightarrow \frac{\frac{\partial z(\theta)}{\partial \theta}}{z(\theta)} = \int_{\tau \sim \pi} \phi(\tau) \frac{e^{\theta^{\mathbb{T}} \phi(\tau)}}{z(\theta)} \mathrm{d}\tau = \int_{\tau \sim \pi} \phi(\tau) \mathbb{P}(\tau|\theta) \mathrm{d}\tau = \mathbb{E}_{\tau \sim \pi} \phi(\tau)$$

This gives us, which gets rid of z:

$$\Rightarrow \frac{\partial \log\mathbb{L}(\theta)}{\partial \theta} = \sum_{\tau_i \in \mathcal{D}} \{\phi(\tau_i) - \mathbb{E}_{\tau \sim \pi} \phi(\tau)\}$$

note that the Expectation term is induced by Policy Pi, which is relevant to and determined by the current Theta.

Intuitively, this means that we are updating θ, so that

$$\theta^{\mathbb{T}} \sum_{\tau_i \in \mathcal{D}} \{\phi(\tau_i) - \mathbb{E}_{\tau \sim \pi} \phi(\tau)\} = \sum_{\tau_i \in \mathcal{D}} \{\mathbb{R}(\tau_i) - \mathbb{E}_{\tau \sim \pi} \mathbb{R}(\tau)\} \quad \uparrow$$

gets larger; which means to change the shape of the reward function by tuning θ, so that w.r.t. this reward function, expert trajectories performs much better than the trajectories sampled by the current policy.

Solve Expectation by Dynamic Programming if we know Pi

$$\mathbb{E}_{\tau \sim \pi} \phi(\tau) = \int_{\tau \sim \pi} \phi(\tau) \mathbb{P}(\tau|\theta) \mathrm{d}\tau = \int_{\tau \sim \pi} (\sum_{s \in \tau} \phi(s)) \mathbb{P}(\tau|\theta) \mathrm{d}\tau = (finite) \sum_{\tau \sim \pi} ((\sum_{s \in \tau} \phi(s)) \mathbb{P}(\tau|\theta))$$

For Finite Markov Decision Process, and when Transition Model T is known, when we only do not know about R, we can use Dynamic Programming over T (t = 1, t = 2, ..) to solve this. <q

$$\Rightarrow \mathbb{E}_{\tau \sim \pi} \phi(\tau) = \sum_t \sum_s \mathbb{P}(s_t = s|\pi) \phi(s)$$

Solve Pi from Theta and Corresponding R under Max Entropy Assumption with RL / Sampling methods

Not clear why this works, for example, why the policy learned by Q-Learning corresponds to the policy induced by Max Entropy Assumption. <q

Algorithm

```
initialize Theta;
repeat:
    solve Pi w.r.t. Reward, Policy given by current Theta by RL algorithms;
    compute gradient to log likelihood by DP;
    Theta <- Theta + Alpha * Gradient, gradient ascent;
```

Deep Variant for the algorithm

Make Reward function non-linear:

$$\mathbb{R}(s) := \mathcal{NN}(s)$$

$$\Rightarrow \mathbb{R}(\tau) = \sum_{s \in \tau} \mathcal{NN}(s)$$

$$\mathbb{P}(\tau|\theta) = z(\theta)^{-1} e^{\sum_{s \in \tau} \mathcal{NN}(s)}$$

note that because of the non-linearity of neural nets, Sum NN(s) does not equal to NN(Sum s), as we did before for linear features.

$$\frac{\partial \log \mathbb{L}(\theta)}{\partial \theta} = \sum_{\tau_i \in \mathcal{D}} (\sum_{s_i \in \tau_i} (\frac{\partial}{\partial \theta} \mathcal{NN}_\theta(s_i)) - \frac{1}{z(\theta)} \frac{\partial z(\theta)}{\partial \theta})$$

note that the first term

$$\sum_{s_i \in \tau_i} (\frac{\partial}{\partial \theta} \mathcal{NN}_\theta(s_i))$$

can be calculated by conventional Back-Propagation algorithms in deep nets; So now we focus on how to calculate the second term.

$$\frac{1}{z(\theta)} \frac{\partial z(\theta)}{\partial \theta} = \sum_{\tau' \sim \pi} (\frac{e^{\sum_{s' \in \tau'} \mathcal{NN}_\theta(s')}}{z(\theta)} \sum_{s' \in \tau'} \frac{\partial}{\partial \theta} \mathcal{NN}_\theta(s)) = \sum_{\tau' \sim \pi} (\mathbb{P}(\tau'|\theta) \sum_{s' \in \tau'} \frac{\partial}{\partial \theta} \mathcal{NN}_\theta(s))$$

note that unlike the linear case, this cannot use Dynamic Programming. Use sampling algorithms to solve this.