Lecture 20 - Imitation learning

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [Click]

Imitation Learning

Motivation. Learning policies from rewards is successful in situations where data is cheap and easily gathered. This approach fails, however, when data gathering is slow, failure must be avoided (e.g. autonomous vehicles), or safety is desired.

- One approach to mitigate the sparse reward problem is to manually design reward functions that are dense in time. However, this approach requires a human to hand-design a reward function with the desired behavior in mind.
- It is therefore desirable to learn by imitating agents performing the task in question.

Imitation Learning

Generally, experts provide a set of demonstration trajectories, which are sequences of states and actions. More formally, we assume that we are given

- State space, action space;
- Access to the transition oracle $\mathbb{P}(s' \mid s, a)$;
- Set of one or more teacher demonstrations $(s_0, a_0, s_1, a_1, \dots)$, where actions are drawn from the teacher's policy π^*.

However, no reward function oracle $\mathcal{R}$ and no explicit transition model $\mathbb{P}(s' \mid s, a)$ are given.

Behavioral Cloning

A natural question raised out of this context is then

Can we learn the teacher's policy using supervised learning?

In behavioral cloning, we aim simply to learn the policy via supervised learning.

- Specifically, we will fix a policy class and aim to learn a policy mapping states to actions given the data tuples $\{(s_0, a_0), (s_1, a_1), \ldots\}$.

Behavioral Cloning

One challenge to this approach is that data is not distributed i.i.d. in the state space.

In RL, errors are compounding and they accumulate over the length of the episode.

- The training data for the learned policy will be tightly clustered around expert trajectories.
- If a mistake is made that puts the agent in a part of the state space that the expert did not visit, the agent has no data to learn a policy from.
- The error scales quadratically in the episode length, as opposed to the linear scaling in standard RL.

Behavioral Cloning

DAGGER: Dataset aggregation:

- This algorithm aims to mitigate the problem of compounding errors by adding data for newly visited states.
- As opposed to assuming there is a pre-defined set of expert demonstrations, we assume that we can generate more data from an expert.
- The limitation of this, of course, is that an expert must be available to provide labels, sometimes in real-time.

Behavioral Cloning

Algorithm 1: DAGGER

Initialize $\mathcal{D} \leftarrow \emptyset$
Initialize $\hat{\pi}_1$ to any policy in Π
for $i = 1$ *to* N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$
  Sample T-step trajectories using π_i
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$
return best $\hat{\pi}_i$ on validation

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Inverse Reinforcement Learning

Motivation. Behavior cloning directly learns the policy as desired, but its practical performance can be limited. The reason is that apart from the input provided by the experts, there are not many generalizations that are provided by the algorithm. Instead, a better generalization can be obtained by learning the reward function, which is a succinct description of the task, from the expert input.

Can we recover the reward function $\mathcal{R}$ from expert input?

Inverse Reinforcement Learning

Can we recover the reward function $\mathcal{R}$ from expert input?

In inverse reinforcement learning, the goal is to learn the reward function (that has not been provided) based on the expert demonstrations.

Linear Feature Reward Inverse RL

We consider a reward which is represented as a linear combination of features

$$R(s) = w^T x(s),$$

where $R(\cdot)$ is a deterministic realization of $\mathcal{R}(\cdot)$ and $w \in \mathbb{R}^d, x : \mathcal{S} \to \mathbb{R}^d$ represent the weight and the feature. The IRL problem is to identify the weight vector w, given a set of demonstrations. The resulting value function for a policy π can be expressed as

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0 = s\right] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t w^T x(s_t) \mid s_0 = s\right] \\ &= w^T \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t x(s_t) \mid s_0 = s\right] = w^T \mu(\pi), \end{aligned}$$

where $\mu(\pi \mid s_0 = s) \in \mathbb{R}^d$ is the discounted weighted frequency of state features $x(s)$ under policy π.

Linear Feature Reward Inverse RL

$$\mathbb{E}_{\pi^*}\left[\sum_{t=0}^{\infty}\gamma^t R^*(s_t) \mid s_0 = s\right] \geq \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t R^*(s_t) \mid s_0 = s\right], \quad \forall \pi,$$

where R^* denotes an optimal reward function. Thus, if an expert's demonstrations are optimal (i.e. actions are drawn from an optimal policy), to identify w it is sufficient to find some w^* such that

$${w^*}^T \mu(\pi^* \mid s_0 = s) \geq {w^*}^T \mu(\pi \mid s_0 = s), \quad \forall \pi, \forall s,$$

where some restrictions are put on w^* to avoid trivial solutions to the linear system. As long as this constraint is linear, the problem can be solved by linear programming.

$$\max_{{w^*}^T} {w^*}^T \mu(\pi^* \mid s_0 = s) - {w^*}^T \mu(\pi \mid s_0 = s), \quad \forall \pi^* \neq \pi, \forall s,$$

$$s.t., \|{w^*}^T\| = 1$$

Apprenticeship learning

Armed with inverse reinforcement learning, the question we are asking is

Can we use the recovered reward to generate a good policy?

Apprenticeship learning

For a policy π to perform as well as the expert policy π^*, it suffices that we have a policy such that its discounted cumulative feature expectations match the expert's policy . More precisely, if

$$\|\mu(\pi \mid s_0 = s) - \mu(\pi^* \mid s_0 = s)\|_1 \leq \varepsilon\,,$$

then by the Cauchy-Schwartz inequality, for all w with $\|w\|_\infty \leq 1$,

$$|w^T \mu(\pi \mid s_0 = s) - w^T \mu(\pi^* \mid s_0 = s)| \leq \varepsilon\,.$$

Apprenticeship learning

Algorithm 2: Apprenticeship learning via linear feature IRL

Initialize policy π_0

for $i = 1, 2, \dots$ **do**

Find the reward function weights w such that the teacher maximally outperforms all previous controllers via the following program

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & \underset{C}{\text{maximize}} \quad C \\ \text{subject to} \quad & w^T \mu(\pi^* \mid s_0 = s) \geq w^T \mu(\pi \mid s_0 = s) + C\,, \\ & \forall \pi \in \{\pi_0, \pi_1, \dots, \pi_{i-1}\}, \forall s\,, \\ & \|w\|_2 \leq 1 \end{aligned}$$

Find optimal policy π_i for current w

if $C \leq \epsilon/2$ **then**

return π_i

hen

Apprenticeship learning

In practice, there are challenges associated with this approach:

- If the expert policy is suboptimal, than the resulting policy is a mixture of somewhat arbitrary policies that have the expert policy in their convex hull.
- This approach relies on being able to compute an optimal policy given a reward function, which may be expensive or impossible.
- There is an infinite number of reward functions with the same optimal policy, and an infinite number of stochastic policies that can match feature counts.

Maximum entropy inverse RL

To address the problem of ambiguity, Maximum Entropy (MaxEnt) IRL considers the collection of all possible H-step trajectories in a deterministic MDP. For a linear reward model, a policy is completely specified by its distribution over trajectories.

Given this, which policy should we choose given a set of k distributions?

Maximum entropy inverse RL

Again, assume that the reward function is a linear function of the features $R(s) = w^T x(s)$. Denoting trajectory j as τ_j, we can write the feature counts for this trajectory as

$$\mu_{\tau_j} = \sum_{s_i \in \tau_j} x(s_i)\,.$$

Averaging over m trajectories, we can write the average feature counts

$$\tilde{\mu} = \frac{1}{k} \sum_{j=1}^{k} \mu_{\tau_j}\,.$$

Maximum entropy inverse RL

The principle of maximum entropy motivates choosing a distribution with no additional preferences beyond matching the feature expectations in the demonstration dataset

$$
\begin{aligned}
\underset{P}{\text{maximize}} \quad & -\sum_{\tau} P(\tau) \log P(\tau) \\
\text{subject to} \quad & \sum_{\tau} P(\tau) \mu_{\tau} = \tilde{\mu}\,, \\
& \sum_{\tau} P(\tau) = 1\,.
\end{aligned}
$$

In the case of linear rewards, this is equivalent to specifying the weights w that yield a policy with the maximum entropy, constrained to matching the feature expectations.

Maximum entropy inverse RL

Maximizing the entropy of the distribution over the paths subject to the feature constraints from observed data implies we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution

$$P(\tau_j \mid w) = \frac{1}{Z(w)} \exp\left(w^T \mu_{\tau_j}\right) = \frac{1}{Z(w)} \exp\left(\sum_{s_i \in \tau_j} w^T x(s_i)\right),$$

with

$$Z(w, s) = \sum_{\tau_s} \exp\left(w^T \mu_{\tau_s}\right).$$

Maximum entropy inverse RL

This induces a strong preference for low cost paths, and equal cost paths are equally probable. Many MDPs of interest are stochastic. In these cases, the distribution over paths depends on both the reward weights and on the dynamics

$$P(\tau_j \mid w, \mathbb{P}(s' \mid s, a)) \approx \frac{\exp\left(w^T \mu_{\tau_j}\right)}{Z(w, \mathbb{P}(s' \mid s, a))} \prod_{s_i, a_i \in \tau_j} \mathbb{P}(s_{i+1} \mid s_i, a_i).$$

The weights w are learned by maximizing the likelihood of the data

$$w^* = \arg\max_w L(w) = \arg\max_w \sum_{\text{examples}} \log P(\tau \mid w).$$

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Maximum entropy inverse RL

The gradient is the difference between expected empirical feature counts and the learner's expected feature counts, which can be expressed in terms of the expected state visitation frequencies

$$\nabla L(w) = \tilde{\mu} - \sum_{\tau} P(\tau \mid w)\mu_{\tau} = \tilde{\mu} - \sum_{s_i} D(s_i)x(s_i)\,,$$

where $D(s_i)$ denotes the state visitation frequency. This approach has been influential, as it provides a principle way to select among the many possible reward functions.

Maximum entropy inverse RL

Algorithm 3: Maximum entropy IRL

Backward pass

Set $Z_{s_i,0} = 0$

Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k \mid s_i, a_{i,j}) \exp(R(s_i \mid w)) Z_{s_k} \,.$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}} \,.$$

Local action probability computation

$$P(a_i, j \mid s_i) = \frac{Z_{a_{i,j}}}{Z_{s_i}}$$

Forward pass

Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$

Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j} \mid s_i) P(s_k \mid a_{i,j} s_i) \,.$$

Summing frequencies

$$D_{s_i} = \sum_t D_{s_i,t}$$

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Question and Answering (Q&A)