ECE 598: Interplay between Control and Machine Learning **Fall 2020**

Lecture 15

Imitation Learning

Lecturer: Bin Hu, Date:11/10/2020

In many situations, whether RL can work or not depends on the cost used in the problem formulation. However, the design of the cost function itself is not a trivial thing and typically requires domain expertise. For relatively simple control tasks such as regulations or tracking, quadratic cost can be used. For more complicated tasks in robotics and autonomy, the cost function is typically handcrafted in a case-by-case manner. Sometimes the objective functions are so complicated such that we do not know how to specify them beforehand. For example, consider the design of self-driving systems. We must carefully trade off competing objectives such as performance, safety, and comfort level in driving. It can be difficult to come up with a precise metric for "comfortable driving." Therefore, how to formulate the cost function for the control design of self-driving is unclear in the first place. This motivates the use of imitation learning.

The rationale behind imitation learning is that we know that human experts are good at driving and grasping. The idea of imitation learning is to utilize expert demonstrations for such control problems where the objective function is not obvious. In general, imitation learning can be applied to settings where demonstrations from human experts are available. Several main techniques used in imitation learning include

1. Behavior Cloning
2. DAGGER
3. Inverse RL
4. Generative Adversarial Imitation Learning (GAIL)

Behavior Cloning. The most basic algorithm in imitation learning is the behavior cloning method which takes in demonstrations, such as a human driver's steering and throttle commands, and attempts to fit a state-action control policy in a supervised learning fashion. Let's say that we are given a control design problem whose objective function is hard to specify beforehand. We assume some demonstrations from experts are available. Specifically, we assume a sequence of state/action pairs $\{x_k, u_k\}_{k=0}^N$ has been demonstrated by the expert. One way to do the control design here is to directly fit a policy on expert demonstrations. The hope is that the fitted policy can mimic the behaviors of human experts and hence perform well on the given control problem. It is natural to formulate such a fitting

problem as a supervised learning problem:

$$\underset{K}{\text{minimize}} \quad \frac{1}{N}\sum_{k=0}^{N} L(K(x_k), u_k) + R(K) \tag{15.1}$$

where L is some loss function measuring the empirical performance of the fitted policy on observed demonstrations, and R is a regularization term introduced to prevent overfiting. In the above formulation, the policy is typically parameterized as a linear function or a neural network. The resultant optimization problem is unconstrained and can be efficiently solved by applying stochastic gradient descent (SGD) or other first-order methods.

DAGGER. One issue for behavior cloning is how to sample the demonstrations. Let's say the human expert is implicitly following some policy K^* which is unknown to us. Then typically one just runs the system $x_{t+1} = f(x_t, K^*(x_t), w_t)$ to generate the trajectories. However, when implementing the fitted controller K, the state space visited can be quite different. This is called the distributional shift. DAGGER fixes this issue by relabeling the states which have not been seen before. Specifically, when K is fitted, then run the system $x_{t+1} = f(x_t, K(x_t), w_t)$ to generate some new sequence $\{x_t, u_t\}$. Throw away the control action here and ask the expert to relabel the state. This gives us a new sequence of demonstrations. Then we can augment all the demonstrations and fit K again and then iterate.

Other methods. The goal of inverse RL is to learn the cost function other than the policy from the demonstrations. GAIL uses the idea of GANs to learn a policy that mimics the experts. See references provided in the class for more explanations of these methods.