CS 294-115 Algorithmic Human-Robot Interaction **Fall 2016**

Lecture 13: Learning from Demonstration

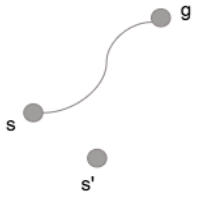
Scribes: Samee Ibraheem and Malayandi Palaniappan - Adapted from Notes by Avi Singh and Sammy Staszak

13.1 Introduction

LfD seeks to answer the following: if a human gives a robot a demonstration of how to achieve a task, how can the robot generalize it to a new setting?

For example, suppose the expert shows the robot a trajectory from a fixed start point s to a goal point g. The trajectory ξ may look as follows.

Now we give the robot a new starting state $\hat{s}$. Based on this demonstration, the robot may be tasked with determining a new trajectory $\hat{\xi}$ from the new starting position s' to the previous goal point g that still achieves the task (e.g. moves naturally, makes the right gesture, moves on the road and not on rocks, etc.)



LfD has two schools of thought: inverse optimal control (IOC) (or inverse reinforcement learning (IRL)) recovers a cost or reward function that explains the demonstration, which the robot can optimize in any new setting. On the other hand, trajectory adaptation or policy learning leads directly a policy that maps state to action.

13.2 Inverse Optimal Control

In order to understand inverse optimal control, let's first discuss the optimal control problem where, given a cost function(al), we want to find the optimal behavior:

$$\mathcal{U} \mapsto \xi^* = \min_{\xi \in \Xi_{(s-g)}} \mathcal{U}[\xi] \tag{13.1}$$

In IOC, we do the opposite: given an optimal trajectory, we want to find the cost function(al) that explains it:

$$\xi_D \longmapsto \mathcal{U} \mapsto \mathbb{R}_+ \, s.t. \mathcal{U}[\xi_D] \leq \mathcal{U}[\xi], \forall \xi \in \Xi_(s - g) \tag{13.2}$$

13.2.1 Why IOC?

There are four main reasons for using IOC.

1. In some settings, it is hard to design a cost function but easy to demonstrate the desired behavior.
   - e.g., driving
2. In other settings, the cost function is unknown entirely.
   - e.g., driving style: how can we define a cost function that incentivizes "aggresive" (or "defensive") driving styles?
   - e.g., predictable human-aware motion: how can we define a cost function that incentivizes a robot to move in a manner that is predictable to humans?
3. IRL can be used to model human behavior and allow robots to better anticipate humans' actions.
4. IRL also has scientific motivations: it can be used to better understand the behavior of humans and animals.
   - e.g., what cost function are cockroaches implicitly optimizing when they walk?

13.2.2 Maximum Margin Planning

The IOC problem involves finding a cost function $\mathcal{U}$ such that

$$\mathcal{U}[\xi_D] \leq \mathcal{U}[\xi], \, \forall \xi \tag{13.3}$$

which can be written as

$$\mathcal{U}[\xi_D] \leq \min_{\xi} \mathcal{U}[\xi] \tag{13.4}$$

The above problem has a trivial solution, $\mathcal{U}[\xi] = k, \forall \xi$. In other words, any constant cost function satisfies the constraints of the given problem. Maximum margin solves this issue by making the following modification to the IOC problem

$$\mathcal{U}[\xi_D] \leq \min_{\xi} [\mathcal{U}[\xi] - l(\xi, \xi_D)] \tag{13.5}$$

In other words, instead of finding a cost function that ensures minimum cost for the demonstrations, we find a cost function that gives minimum cost for the demonstrations by a *margin*. The margin is given by $l(\xi, \xi_D)$, and its value should be small when ξ is close to ξ_D, and large otherwise. A simple choice of l is given by:

$$l(\xi, \xi_D) = \begin{cases} 0 & \text{if } \xi = \xi_D \\ 1 & \text{otherwise} \end{cases}$$

We want that Equation 13.5 holds true with the maximum margin possible, so the problem becomes

$$\max_{\mathcal{U}} [\min_{\xi} [\mathcal{U}[\xi] - l(\xi, \xi_D) - \mathcal{U}[\xi]]] \tag{13.6}$$

We can reverse the sign to go from max to min, and add regularizer R on the cost function to obtain the following optimization problem

$$\min_{\mathcal{U}}[U[\xi_D] - \min_{\xi}[\mathcal{U} - l(\xi, \xi_D)] + \lambda R(\mathcal{U})] \tag{13.7}$$

In order to solve the above optimization problem, we have to parameterize the function $\mathcal{U}$. For simplicity, let's assume that $\mathcal{U}$ is linear in the feature space. This give us $\mathcal{U}(\xi) = w^T\phi(\xi)$, where $\phi(\xi)$ denotes the feature vector. We can solve the above problem using gradient descent if we can compute gradients of the objective with respect to w. Let us write down the objective $C(w)$ as follows:

$$C(w) = w^T\phi(\xi) - \min_{\xi}[w^T\phi(\xi) - l(\xi, \xi_D)] + \frac{\lambda}{2}\|w\|^2 \tag{13.8}$$

The gradient descent step is then given by

$$w_{i+1} = w_i - \nabla_{w_i} C \tag{13.9}$$

Calculating the gradient $\nabla_w C$ is trivial except for the $-\min_{\xi}[w^T\phi(\xi) - l(\xi, \xi_D)]$ term. Let us define $\xi^*_w = \arg\min_{\xi} w^T\phi(\xi) - l(\xi, \xi_D)$. We observe that $-\min_{\xi}[w^T\phi(\xi) - l(\xi, \xi_D)]$ is piecewise linear, so we get its gradient as $\nabla_{w_i} = -\phi(\xi^*_{w_i})$. Inserting this back into the gradient descent update, we get

$$\begin{aligned} w_{i+1} &= w_i - \alpha[\phi(\xi_D) - \phi(\xi_{w_i}) + \lambda w] \\ w_{i+1} &= w_i(1 - \alpha\lambda) - \alpha(\phi(\xi_D) - \phi(\xi^*_{w_i})) \end{aligned} \tag{13.10}$$

13.2.2.1 A simple example

To get some intuition about the above update equation, let us consider an example. Let us consider an expert demonstration ξ_D, in which a robot goes over the grass (and not over rock). In the feature space f, we can denote this as $\phi\xi_D = \begin{pmatrix}1\\0\end{pmatrix}$. In this representation, the top element in the vector is 1 when the robot goes over the grass, and the bottom element is 1 when the robot goes over a rock. Let us say that the current best trajectory $\xi^*_{w_i}$ goes over the rock, so $f^*_{w_i} = \begin{pmatrix}0\\1\end{pmatrix}$. Applying the gradient descent update step in Equation 13.10 gives us the following

$$\begin{aligned} w_{i+1} &= (1 - \alpha\lambda)w_i - \alpha[\begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0\\1\end{pmatrix}] \\ w_{i+1} &= (1 - \alpha\lambda)w_i - \alpha\begin{pmatrix}1\\-1\end{pmatrix} \end{aligned} \tag{13.11}$$

In the above update, $w(O)$, the grass weight, decreases, and $w(1)$, the rock weight, increases. This makes going over grass less expensive, and going over rock more expensive. This makes sense since the expert trajectory goes over the grass.

13.2.3 Maximum Entropy IRL

Maximum Margin Planning assumes that our demonstration (or demonstrations) is perfectly optimal. What if our demonstrator performs suboptimally? Maximum Entropy IRL attempts to capture this noisiness by defining a probability function over trajectories from which we would like to sample:

$$p(\xi|w) \propto e^{-\beta w^T \phi(\xi)},$$

where β determines the noisiness of our distribution. $\beta = 0$ implies a completely irrational demonstrator, whereas $\beta = \infty$ implies that our demonstrator is completely rational.

How we do this? We want the w that gives us the probability distribution of maximum entropy, but with minimal cost. More specifically,

$$\begin{aligned} \max_w P(\xi_D|w) \implies & \max_w \log P(\xi_D|w) \\ & = \max_w \Big[-\beta w^T \phi(\xi_D) - \log \sum_{\xi} e^{-\beta w^T \phi(\xi)} \Big]. \end{aligned} \tag{13.12}$$

Taking the gradient, we get

$$\begin{aligned} \nabla &= -\beta \phi(\xi_D) - \frac{1}{\sum_{\xi} e^{-\beta w^T \phi(\xi)}} \sum_{\xi} e^{-\beta w^T \phi(\xi)} \Big(-\beta \phi(\xi) \Big) \\ &= -\beta \phi(\xi_D) + \beta \sum_{\xi} \frac{e^{-\beta w^T \phi(\xi)}}{\sum_{\bar{\xi}} e^{-\beta w^T \phi(\bar{\xi})}} \phi(\xi) \\ &= -\beta \Big[\phi(\xi_D) - \sum_{\xi} P(\xi|w) \phi(\xi) \Big]. \end{aligned} \tag{13.13}$$

Note that we can extend this to multiple demonstrations by multiplying the probabilities for each demonstration together.

Now compare the above to what we derived for Maximum Margin Planning:

$$\nabla_{MMP} = \phi(\xi_D) - \phi(\xi_w^*).$$

The difference is whether we are using optimal features, as for Maximum Margin Planning, or expected features, as we do for Maximum Entropy IRL.

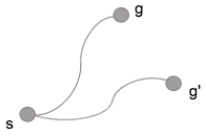
13.3 Supervised Learning

An alternative to IOC is to directly learn a policy : q q that maps state to action.

A simple way to obtain such a policy is to use supervised learning. The demonstrated trajectory contains training examples in the form (q, q') that can be used for training. Any regressor can be used to learn π.

13.4 Trajectory Adaptation

A particular policy learning method is trajectory adaptation. Let's suppose an expert shows the robot a trajectory from a fixed start point s to a goal point g. However, we are interested in finding the trajectory to a new goal point g' (or from a new start, or that avoids a new obstacle, etc.).



From this, we may calculate the desired trajectory ξ'

$$\xi' = \min_{\xi} ||\xi_D - \xi||_A^2 \tag{13.14}$$

s.t. we have a new constraint such as $\xi(t) = g'$. We call this model-free learning from demonstration.

13.5 Comparing IOC and Trajectory Adaptation

13.5.0.1 Pros of IOC

- Cost functions functions are more generalizable. The start can change, but so can the goal, and so can the environment. Policies typically only handle start changes, might be able to handle goal changes, but it is difficult to change the MDP entirely.
- Takes advantage of the dynamics model, i.e. the robot knows the effects of its actions.

13.5.0.2 Pros of Trajectory Adaptation

- Computationally cheaper
- Do not need to know the dynamics of the environment
- Better generalization locally, no need to "explain" the expert demonstration

13.6 Appendix 1: Reinforcement Learning (RL) vs. Optimal Control (OC)

Consider an agent acting in a Markov Decision Process, $MDP = (S, A, T, R)$. S denotes the set of states; A denotes the set of actions the agent can take; $T = P(s'|s, a)$ denotes the transition distribution that describes how the agent moves between states; and $R(s, a, s')$ denotes the reward function that dictates the reward the agent receives for transitioning between states. The agent's goal is to learn a policy $\pi : S \to A$.

Why learn a policy as opposed to a trajectory? Well, imagine having a plan for what to do after winning a lottery that you end up losing... Since the transition function is stochastic, the agent needs to account for its uncertainty about what state it will end up in.

In OC (or planning), the agent is given the full MDP, and is tasked to come up with π^*, the optimal policy for acting in this MDP.

In RL, we assume that we are given just (S, A, R), from which we still wish to get π^*. There are two types of RL: in Model-Based RL, we first learn a model of the transition distribution, $\tilde{T}$, and compute the

optimal policy using $\tilde{T}$ in place of the true transition distribution. In contrast, Model-Free RL aims to learn a policy directly using, for example, Policy Gradient methods or Q-learning.