

Lecture 13: Learning from Demonstration

Scribes: Avi Singh and Sammy Staszak

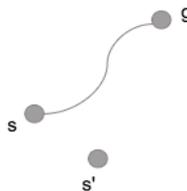
13.1 Introduction

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 ζ 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 $\hat{\zeta}$ 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 trajectory:

$$\mathcal{U} \mapsto \zeta^* = \min_{\zeta \in \Xi(s-g)} \mathcal{U}[\zeta] \quad (13.1)$$

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

$$\zeta_D \mapsto \mathcal{U} : \Xi \rightarrow \mathbb{R}_+ \text{ s.t. } \mathcal{U}[\zeta_D] \leq \mathcal{U}[\zeta], \forall \zeta \in \Xi (s - g) \quad (13.2)$$

13.2.1 Maximum Margin Planning

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

$$\mathcal{U}[\zeta_D] \leq \mathcal{U}[\zeta], \forall \zeta \quad (13.3)$$

which can be written as

$$\mathcal{U}[\zeta_D] \leq \min_{\zeta} \mathcal{U}[\zeta] \quad (13.4)$$

The above problem has a trivial solution, $\mathcal{U}[\zeta] = k, \forall \zeta$. 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

$$\mathcal{U}[\zeta_D] \leq \min_{\zeta} [\mathcal{U}[\zeta] - l(\zeta, \zeta_D)] \quad (13.5)$$

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 $l(\zeta, \zeta_D)$, and its value should be small when ζ is close to ζ_D , and large otherwise. A simple choice of l is given by:

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

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

$$\max_{\mathcal{U}} [\min_{\zeta} [\mathcal{U}[\zeta] - l(\zeta, \zeta_D)] - \mathcal{U}[\zeta_D]] \quad (13.6)$$

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

$$\min_{\mathcal{U}} [\mathcal{U}[\zeta_D] - \min_{\zeta} [\mathcal{U} - l(\zeta, \zeta_D)] + \lambda R(\mathcal{U})] \quad (13.7)$$

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 $\mathcal{U}(\zeta) = w^T f_{\zeta}$, where f_{ζ} 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:

$$C(w) = w^T f_{\zeta} - \min_{\zeta} [w^T f_{\zeta} - l(\zeta, \zeta_D)] + \frac{\lambda}{2} \|w\|^2 \quad (13.8)$$

The gradient descent step is then given by

$$w_{i+1} = w_i - \nabla_{w_i} C \quad (13.9)$$

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

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

13.2.1.1 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 $f_{\xi_D} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. 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

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

In the above update, $w(0)$, 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.

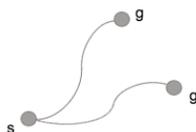
13.3 Supervised Learning

An alternative to IOC is to directly learn a policy $\pi : q \rightarrow \dot{q}$ 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, \dot{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.).



Trajectory adaptation simply calculates the desired trajectory ζ'

$$\zeta' = \min_{\zeta} \|\zeta_D - \zeta\|_A^2 \quad (13.12)$$

s.t. the new constraint $\zeta(t) = g'$.

13.5 Pros of each method

13.5.1 Pros of IOC

- Cost 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.

13.5.2 Pros of Trajectory Adaptation

- Computationally cheaper
- Do not need to know the dynamics, simply roll out the policy.
- Better generalization locally, no need to "explain" the expert demonstration via a cost function