CS 294-115: Algorithmic Human-Robot Interaction Fall 2015

Lecture 7 — Oct. 1, 2015

Prof. Anca Dragan *Scribe: Rohan, Evan*

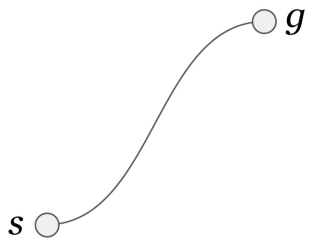
1 Learning from Demonstration (LfD) Overview

So far, we talked about how to optimize. Now we address *what* to optimize, starting with Learning from Demonstration (LfD).

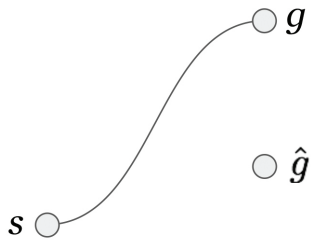
LfD seeks to answer the following: if a human gives the robot a demonstration of how to achieve a task, how can the robot make use of this to execute the appropriate policy for the task in a new setting?

LfD has two schools of thought: **model-based** and **model-free**. The model-based methods seek to recover a cost function for which demonstrations are optimal, while the model-free methods determine direct mappings of demonstrations into new settings.

LfD is crucial because the environment and properties of tasks that robots must execute can be constantly changing. The robot must learn to adapt the demonstrations it receives to different settings, layouts, and locations. For example, suppose the expert shows the robot a trajectory from a fixed start point s to a goal point g. The trajectory ξ_D may look as follows.



Based on this demonstration, the robot may be tasked with determining a trajectory $\hat{\xi}$ from s to a new goal point $\hat{g}$.



The key question here that LfD seeks to answer is how to find a good $\hat{\xi}$ given the new $\hat{g}$.

2 Model-Based LfD

Model-based LfD attempts to recover a cost function from the demonstrations. In other words, we seek an objective/cost $\mathcal{U}$ such that $\mathcal{U}(\xi_D) \leq \mathcal{U}(\xi)\ \forall \xi \in \Xi_{s \to g}$. In this sense, we treat the demonstrations we receive as optimal, or at least close to optimal. This method of recovering a cost function from demonstrations is very linked with the fields of inverse optimal control and inverse reinforcement learning.

Once we've found a suitable $\mathcal{U}$ based on demonstrations, we can compute a trajectory for a new setting with goal $\hat{g}$ as $\hat{\xi} = \arg\min_{\xi \in \Xi_{s \to \hat{g}}} \mathcal{U}(\xi)$

The bottleneck here is not only computation, but also learning and representation. Optimization is imperfect, but it is feasible for these techniques, although further work is needed and underway for applying this to higher dimensions. If the cost function $\mathcal{U}$ happens to be convex, then there's no trouble since gradient descent will come to the rescue.

2.1 How to Learn a Cost Function: Maximum-Margin Planning

We write the inequality describing $\mathcal{U}$ with a minimum, since it will be optimal with respect to all trajectories by transitivity:

$\mathcal{U}(\xi_D) \leq \min_{\xi \in \Xi_{s \to g} \setminus \{\xi_D\}} \mathcal{U}(\xi)$.

For robustness, max-margin planning requires that $\mathcal{U}(\xi_D)$ is better than $\mathcal{U}$ for any other trajectory ξ by a margin of l:

$\mathcal{U}(\xi_D) \leq \min_{\xi} \mathcal{U}(\xi) - l$.

We then pick $l = l(\xi, \xi_D)$, a loss measure depending on the distance between two trajectories, for a smooth, adaptive margin:

$\mathcal{U}(\xi_D) \leq \min_{\xi} \mathcal{U}(\xi) - l(\xi, \xi_D)$.

We transform this into an optimization problem and add regularization R:

$\min_{\mathcal{U}} \mathcal{U}(\xi_d) - \min_{\xi} [\mathcal{U}(\xi) - l(\xi, \xi_D)] + \frac{\lambda}{2} R(\mathcal{U})$.

We can have multiple demonstrations for multiple situations (in this example, different goals), and sum this cost over all of them. What if we had multiple, different demonstrations for the same g? This method would assume that all demonstrations are optimal.

For tractability, we parameterize $\mathcal{U}$ according to some featurization of the state. For simplicity, let's pick $\mathcal{U}$ to be linear in the feature space. Then, $\mathcal{U}$ is parameterized as $\mathcal{U}(\xi) = w^\top f_\xi$ for features f_ξ. Examples of these features could include trajectory length, distances to obstacles, and so on. This formulation admits a cost function on w:

$C(w) = w^\top f_{\xi_D} - \min_{\xi} \left[w^T f_\xi - l(\xi, \xi_D) \right] + \frac{\lambda}{2} \|w\|^2$.

We define $\xi_w^* = \arg\min_{\xi} w^T f_\xi - l(\xi, \xi_D)$ and rewrite $C(w)$:

$C(w) = w^\top f_{\xi_D} - (w^T f_{\xi_w^*} - l(\xi_w^*, \xi_D)) + \frac{\lambda}{2} \|w\|^2$.

Now $C(w)$ is convex in w, so we can optimize it by gradient descent.

$\nabla_w C(w) = f_{\xi_D} - f_{\xi_w^*} + \lambda w$

$w_{i+1} = w_i - \frac{1}{\alpha}\nabla_w C$

$w_{i+1} = w_i - \frac{1}{\alpha}(f_{\xi_D} - f_{\xi_w^*}) - \frac{1}{\alpha}\lambda w$

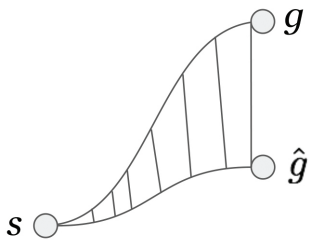
Intuitively, we are raising the cost of the current best ξ and lowering the cost of the demonstrated ξ_D. For example, suppose the demonstrated trajectory ξ_D prefers to go on grass: $f_{\xi_D} = \begin{bmatrix}1\\0\end{bmatrix}$ where the upper entry represents going on grass and the lower entry represents going on rock. Suppose also that the current best ξ_w^* prefers to go on rock: $f_{\xi_w^*} = \begin{bmatrix}0\\1\end{bmatrix}$. Then,

$w_{i+1} = w_i - \frac{1}{\alpha}\begin{bmatrix}+1\\-1\end{bmatrix} - \frac{1}{\alpha}\lambda w.$

So, we decrease the weight of grass (thus decreasing the associated cost) and increase the weight of rock, as we would intuitively expect.

3 Model-Free LfD

In model-free LfD, we determine a direct mapping of demonstrations to new settings. The following figure shows a potential mapping between ξ_D and $\hat{\xi}$ for generalizing from reaching g to reaching $\hat{g}$.



We are now solving the following optimization problem:

$\hat{\xi} = \arg\min_{\xi \in \Xi_s} \frac{1}{2}\|\xi_D - \xi\|_A^2 \text{ s.t. } \xi(T) = \hat{g}$

where Ξ_s is the space of all trajectories starting from s, the norm is represented by multiplication with the matrix A, and T is the last timestep in the trajectory.

To solve this, we form the Lagrangian: an augmented objective with Lagrange multipliers in place of constraints:

$\mathcal{L}(\xi, \lambda) = \frac{1}{2}\|\xi_D - \xi\|_A^2 + \lambda(\xi(T) - \hat{g}).$

We now enforce the first-order KKT conditions for optimality: $\nabla_\xi \mathcal{L} = 0$ and $\nabla_\lambda \mathcal{L} = 0$.

$$\nabla_\xi \mathcal{L} = A(\xi - \xi_D) + \begin{bmatrix}0\\0\\\vdots\\0\\\lambda\end{bmatrix} = 0$$

$$\xi = \xi_D - A^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \lambda \end{bmatrix}$$

$$\nabla_\lambda \mathcal{L} = \xi(T) - \hat{g} = g - A_{NN}^{-1}\lambda - \hat{g} = 0$$

$$\lambda = \frac{g - \hat{g}}{A_{NN}^{-1}}$$

If A is the identity transformation, ξ will simply follow ξ_D and then jump to $\hat{g}$ at time T (because in this case, $\lambda = g - \hat{g}$). However, we can choose A to encourage smoothness or other properties. A determines how the constraint for the last timestep (or any intermediate waypoint) will be propagated throughout ξ.

Dynamic movement primitives are a popular model-free LfD method: cast the demonstration as a spring damper system tracking a moving target, then change the start and goal parameters of that moving target. They instantiate this formalism for a particular choice of A. See optional reading for the proof.

4 Discussion

Model-based techniques can generalize further, while model free techniques are more expressive.