Useful Robots

Robots should be *general purpose machines*.

That means:

- Programmable
- End-user programmable

How do we do this?
Learning from Demonstration

Procedural knowledge is hard for robots but easy for humans.

Rather than program the robot using code, how about we program it by showing it what to do?

This is known as learning from demonstration (LfD).
Demonstration and Imitation

Humans (and some primates) are good at this.
Imitation and Culture

Many motor control behaviors are extremely hard to learn from scratch, even if you have the reward function. They are easy to copy though!

If you have a large population of people, then they can be considered to be performing a massive, parallel search for new interesting behaviors.

Then we have mechanisms for copying, retaining, and transmitting the new knowledge.

(the Fosbury flop, circa 1965)
Fosbury Flop
More Formally

Quite simple setting (we will elaborate later).

Let’s say we have an MDP:  \((S, A, R, T, \gamma)\)

We would like to solve it, and are given demonstrations. Each demonstration is a trajectory:

\[ \tau = ((s_1, a_1), \ldots, (s_n, a_n)) \]

We’d still like to produce a policy \( \pi \).
Learning from Demonstration

Families of approaches:
- Learn policy directly
- Learn model \((R \text{ and } T)\)
- Use samples to learn \(V\)
- Learn reward function itself
  - Inverse reinforcement learning - later
- Learn options from demonstration
LfD: Learning a Policy Directly

Most immediate and direct approach:

- Learn a policy: classifier
- Each state $s_i$ is input
- Each action $a_i$ is label
- Training data: observed $(s_i, a_i)$ pairs.
- Learn to predict new “label” (action)
- Hope that this generalizes nicely.

Generally, this is very fragile!

(Demiris and Hayes, 1994)
Generalizing

We can generalize this somewhat by being flexible about the actions predicted from the states, to depend on the goal.

(Pastor et al., ICRA 2009)
Learning Model/V

Very simple:

- Use trajectories as samples for $V$ or $R+T$.
- Do off-policy RL or learn a model and plan.

- This is similar to what we’ve already covered.
Unstructured Demonstrations

More interesting:

• Break demonstration into multiple subtasks
• Allows the user to give unstructured demonstration
• Reuse subtasks, not whole demonstration
• Perhaps resequence subtasks later.
Changepoint Detection

\[ g(j - i - 1)p(q_i) \]

unobserved abstractions

\[ s_{i+2}, r_{i+2} \]

observed transitions

\[ P(i, j, q_i)(1 - G(j - i - 1)) \]

\[ s_{j-1}, r_{j-1} \]

\[ s_j, r_j \]

\[ P(j, t, q) = \frac{\pi^{-\frac{n}{2}}}{\delta m} |(A_q + D)^{-1}|^{\frac{1}{2}} \frac{u^{\frac{n}{2}}}{(y_q + u)^{\frac{n+v}{2}}} \frac{\Gamma\left(\frac{n+v}{2}\right)}{\Gamma\left(\frac{v}{2}\right)} \]

(Fearnhead and Liu 2007)
LfD with Skills

Combine skill chaining with skill-specific abstractions.

(Konidaris, Kuindersma, Grupen and Barto, NIPS 2010)
LfD with Skills

Trajectory segmented into appropriate skills + abstractions.
<table>
<thead>
<tr>
<th>#</th>
<th>Abstraction</th>
<th>Description</th>
<th>Trajectories Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>torso-purple</td>
<td>Drive to door.</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>endpoint-purple</td>
<td>Push the door open.</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>torso-orange</td>
<td>Drive toward wall.</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>torso-yellow</td>
<td>Turn toward the end wall.</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>torso-purple</td>
<td>Drive to the panel.</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>endpoint-purple</td>
<td>Press the panel.</td>
<td>3</td>
</tr>
</tbody>
</table>
Follow-on Work

(Niekum et al., IROS 2012)
Building Higher-level Controllers

(Niekum et al., RSS 2013)