Robots, Reinforcement Learning and Data Mining
Kenneth Bogert
Doctoral candidate

Indrajit Das
Masters student

Maulesh Trivedi
Masters student
What is inverse reinforcement learning?
< state, reward >
reward function?

< state, action >
Definition as stated in the original paper by Stuart Russell, 1998
“Given: 1) measurements of an agent’s behavior over time, in a variety of circumstances; 2) if needed, measurements of the sensory inputs to that agent; 3) if available, a model of the environment

Determine: the reward function being optimized”
Presuppositions

The reward function is a succinct, compact and transferable component of the task definition

It is easier to learn than the policy and the value function from observations

Often, it is not easily specified especially when multiple attributes are involved
Learned the reward function containing 24 features using IRL

(Abeel, Coates, Quigley & Ng 07)
How do we learn the reward function from trajectory data?
Almost all methods model the reward function as a linear combination of basis functions (feature functions)

\[ R(s,a) = \sum_k \theta_k \phi_k(s,a) \]

Problem reduces to learning the weights \( \theta \) from trajectory data
Match feature expectations due to a candidate policy $\pi$ with those from observed data

$$\sum_s \mu_\pi(s) \phi(s, \pi(s)) = \sum_{(s,a) \in traj} \phi(s, a)$$
Challenge:
Degeneracy – Several reward functions and associated policies may match observed data
We adopt the principle of maximum entropy (Jaynes 57) applied to IRL (Ziebart & Maas 08, Boulimarias et al. 12)

MaxEnt realizes the highest-entropy distribution over policies that satisfy the constraints

Avoids bias toward any particular policy
Nonlinear program solved using Lagrangian relaxation 
\( \mathcal{L}(\Pr, \eta, \theta) \) and BFGS gradient descent
Multi-robot patrolling
Multi-robot patrolling
Coordination is needed
Learner

Multimodal streaming data

\{ (s, a)^1, (s, a)^2, (s, a)^3, ..., (s, a)^n \}
Learner

Raw sensor data
Raw sensor data (11 GB)

- Each 640 x 480 frame is about 1.2 MB
- Learner observing for 5 mins at 30 frames/sec

Key events

- Extract 4 variables: e.g., blob centroid, distance to blob, position on map
- CMVision system in ROS

State-action pairs

- Classify events into state and action
- Time step of 1 sec
- 30 frames per sec result in 30 \((x,y)\) coordinates so use a voting system to resolve any conflict

\[\{(s, a)^1, (s, a)^2, (s, a)^3, \ldots, (s, a)^n\}\]
N = 2 robots

Partially occluded trajectories

Coordination between robots disturbs their policy-guided behaviors

Representative of real-world challenges
Model each robot as a MDP and inversely learn the reward functions. Use observed portions of state space only. Model the coordination as equilibrium of a coordination game. Coordination between robots disturbs their policy-guided behaviors. N = 2 robots partially occluded trajectories.
Added challenges:
Further degeneracy due to reduced data
Cannot use BFGS because gradient is undefined for unobservable states
Interactions may occur in occluded portions of state space
Coordination game has multiple equilibria
Why not model multiple robots as a joint MDP (Decentralized MDP)?

Large, does not scale to many robots

Occlusion introduces partial observability of states
We use individual MDPs and an interaction game

Better suited for sparse interactions
State-visitation frequency is now computed from simulations of each robot’s policy under consideration + equilibrium policies during interaction.
Now that the patrollers’ behaviors are acquired, when should the learner start moving?
Experiment:

mIRL*+Int  Occlusion and interaction
mIRL*  Occlusion but no interaction
Known policy  Exact preferences known
Modeling interaction is beneficial
Learning accuracy affects success rate
Success rate improves with observation time
Experiment with physical robots
<table>
<thead>
<tr>
<th>N ≥ 2 robots</th>
<th>Partially occluded trajectories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination between robots disturbs their policy-guided behaviors</td>
<td>Relax prior knowledge requirements pertaining to the (patrollers’) model</td>
</tr>
</tbody>
</table>
Focus so far has been on learning reward function in single-expert settings

Transition function is either assumed to be known or is assumed to be deterministic that can easily be determined from data

High levels of occlusion preclude using supervised learning for determining transition function
Limit our scope to a transition function that is composed of a deterministic core perturbed by transition error probabilities

$$\varphi: S \times A \rightarrow S$$
Key Observation

Let transition probabilities be a function of underlying component outcome probabilities

E.g.: probability of moving forward successfully is a function of both wheels rotating at the same speed correctly

The observed trajectory informs associated component probabilities

If some of these components are shared with transitions in occluded portions of state space, then information is transferred that facilitates obtaining occluded transition probabilities
Associate each state-action pair with its transition features, $\xi_{s,a} = \{\tau_1, \tau_2, \ldots, \tau_k\}$

$$T(s, a, \varphi(s, a)) = \Pr(\tau_1, \tau_2, \ldots, \tau_{|\xi_{s,a}|}) \approx \prod_{\tau \in \xi_{s,a}} \Pr(\tau)$$
From the observed trajectory 
\( \langle (s, a)^0, (s, a)^1, (s, a)^2, \ldots, (s, \emptyset)^T \rangle \)
we obtain probability of next state given observed state-action pair, 
\( q^{\varphi(s,a)} \)

Aggregate

\[
\prod_{\tau \in \xi^{s,a}} \Pr(\tau) = q^{\varphi(s,a)}
\]
Degenerate due to many feature distributions

Principle of maximum entropy solutions

\[
\begin{align*}
\max_{\Delta_1, \ldots, \Delta_N} & \quad \left( -\sum_{n=1}^{N} \sum_{\tau \in \mathcal{T}_n} Pr(\tau) \log Pr(\tau) + Pr(\bar{\tau}) \log Pr(\bar{\tau}) \right) \\
\text{subject to} & \quad \sum_{\tau \in \xi^s_n, \bar{\tau}} \log Pr(\tau) = \log q_n \\
& \quad Pr(\tau) + Pr(\bar{\tau}) = 1 \\
& \quad \forall \tau \in \mathcal{T}_n, n = 1 \ldots N
\end{align*}
\]

(Bard 1950)
Experiments

\[ \text{mIRL}^*_{/T} + \text{Int} \]

learn T followed by mIRL*+Int

\[ \text{mIRL}^* + \text{Int} \]

set transition probabilities

\[ \text{DBN}_{EM} \]

model T using a DBN and learn parameters using EM

\[ \text{Known R}_{/T} \]

Learn T but reward function is known
What is the benefit of learning the transition distributions of patrollers?

Can we correctly identify the patroller with the damaged wheel?

Does learning the patrollers’ behaviors help?
Experiments with physical robots
<table>
<thead>
<tr>
<th>Method</th>
<th>J’s left wheel damaged</th>
<th>No damaged wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td>mIRL*___T + Int</td>
<td>.6</td>
<td>.5</td>
</tr>
<tr>
<td>mIRL* + Int</td>
<td>.5</td>
<td>.4</td>
</tr>
<tr>
<td>Random</td>
<td>.4</td>
<td>.2</td>
</tr>
</tbody>
</table>
Future work

In trajectory data, actions may not be easily discernible
E.g., force applied toward lifting objects is hard to visually ascertain

Can a robot observing another performing a task join in to form an ad hoc team (DARPA challenge)?