Principles of Robot Autonomy II

Imitation Learning
Today’s itinerary

• Intro to Imitation Learning

• Behavioral Cloning

• Imitation Learning with Interactive Experts

• Inverse RL (MMP, Max Ent IRL)

• Learning from other sources of data (preferences, physical feedback)
Today’s itinerary

• Intro to Imitation Learning

• Behavioral Cloning

• Imitation Learning with Interactive Experts

• Inverse RL (MMP, Max Ent IRL)

• Learning from other sources of data (preferences, physical feedback)
Imitation Learning in a Nutshell

• **Given:** Demonstrations or Demonstrator
• **Goal:** Train a policy to mimic demonstrations
Ingredients of Imitation Learning

Demonstrator or Demonstrations

Environment/Simulator

Policy Class

Loss Function

Learning Algorithm
Why Imitation Learning?

For the Sake of Robot Learning:

• It is difficult to learn from sparse rewards (unless data is cheap and you don’t care about seeing lots of failures).

• Hand-designing rewards is hard.
Just design the right reward function

\[ R_H(s) \]

\[ a^*_R = \arg\max_{a_R} R_H(s) \]
Why Imitation Learning?

For the Sake of Robot Learning:

• It is difficult to learn from sparse rewards (unless data is cheap and you don’t care about seeing lots of failures).
• Hand-designing rewards is hard.

For the Sake of Learning Human Models:

• Learning human’s intents, preferences, and underlying reward functions.
Problem Setup

MDP with no reward functions:

- State space, $S$ (sometimes partially observable)
- Actions space, $A$
- An expert policy $\pi^*$ that maps states to distributions over actions: $\pi^*(s) \rightarrow P(s)$
- Transition model $P(s_{t+1}|s_t, a_t)$: simulator or environment

**Goal:** Learn an imitating policy $\pi_\theta(s)$ that imitates the expert demonstrations
Problem Setup

**Rollout:** Sequentially execute $\pi(s_0)$ on an initial state
- produce trajectory: $\tau = (s_0, a_0, s_1, a_1, ...)$.

$P(\tau|\pi)$: Distribution of trajectories induced by a policy
1. Sample $s_0$ from $P_0$ (distribution over initial states).
2. Initialize $t = 1$. Sample action $a_i$ from $\pi(s_{t-1})$.
3. Sample next state $s_t$ from applying $a_t$ to $s_{t-1}$ (requires access to environment).
4. Repeat form step 2 with $t = t + 1$.

$P(s|\pi)$: Distribution of States induced by a policy
- Let $P_t(s|\pi)$ denote distribution over $t$-th state.
- $P(s|\pi) = \frac{1}{T} \sum_t P_t(s|\pi)$
Example: Racing Game

\( s = \) game screen
\( a = \) turning angle

**Training set:** \( D = \{ \tau = \{(s_i, a_i)\} \} \) from \( \pi^* \)

**Goal:** Learn \( \pi_\theta(s) \to a \)
Today’s itinerary

• Intro to Imitation Learning

• Behavioral Cloning

• Imitation Learning with Interactive Experts

• Inverse RL (MMP, Max Ent IRL)

• Learning from other sources of data (preferences, physical feedback)
Behavioral Cloning (reduction to supervised learning)

Define $P^* = P(s|\pi^*)$ (distribution of states visited by the expert)

(Recall $P(s|\pi^*) = \frac{1}{T} \sum_t P_t(s|\pi^*)$)

(sometimes abuse notation: $P^* = P(s, a^* = \pi^*(s)|\pi^*)$)

Learning Objective:

$$\arg \min_{\theta} \mathbb{E}_{(s,a^*) \sim P^*} L(a^*, \pi_{\theta}(s))$$

Interpretations:

1. Assuming perfect imitation so far, learn to continue imitating perfectly
2. Minimize 1-step deviation error along the expert trajectories
Behavioral Cloning: ALVINN

Learning Objective:

\[
\begin{align*}
\arg\min_\theta \mathbb{E}_{(s,a^*) \sim P} & L(a^*, \pi_\theta(s)) \\
= & \arg\min_\theta \mathbb{E}_{(s,a^*) \sim P} KL(a^*, \pi_\theta(s))
\end{align*}
\]

(General) Imitation Learning vs Behavioral Cloning

• Behavioral Cloning (supervised learning):

\[
\arg \min_{\theta} \mathbb{E}_{(s,a^{*}) \sim P^*} L(a^{*}, \pi_{\theta}(s))
\]
Distribution provided exogenously

• (General) Imitation Learning:

\[
\arg \min_{\theta} \mathbb{E}_{S \sim P(S|\theta)} L(\pi^{*}(s), \pi_{\theta}(s))
\]
Distribution depends on the rollout
\[
P(S|\theta) = \text{state distribution of } \pi_{\theta}
\]
What can go wrong?

**Errors in supervised learning:**
- Assume *independent and identically distributed* (IID) state, action pairs, then if we have error at time $t$ with probability $\epsilon$, then over a time period the error would be bounded by $\epsilon T$ in expectation.

In imitation learning, the state distribution of our data depends on the choice of actions.

End up in states that you have not seen before...

... compounding errors

During training:
\[
s \sim P^*
\]

In test time:
\[
s \sim P(s|\pi_\theta)
\]
Limitations of Behavioral Cloning: Compounding Errors

\( \pi_\theta \) makes a mistake
New state sampled not from \( P^* \)!
Worst case is catastrophic!
Cannot recover from new states
When to Use Behavioral Cloning?

**Advantages:**
- Simple
- Efficient

**Disadvantages:**
- Distribution mismatch between training and testing
- No long-term planning

**Use When:**
- 1-step deviations not too bad!
- Learning reactive behaviors
- Expert trajectories “cover” state space

**Don’t Use When:**
- 1-step deviations can lead to catastrophic error
- Optimizing long-term objective (at least not without a stronger model)
Types of Imitation Learning

**Behavioral Cloning**

\[
\arg \min_\theta \mathbb{E}_{(s,a^*) \sim P^*} L(a^*, \pi_\theta(s))
\]

Works well when \( P^* \) is close to \( P_\theta \)

**Direct Policy Learning (via Interactive Demonstrator)**

Requires Interactive Demonstrator (BC is a 1-step special case)

**Inverse RL**

Learn \( r \) such that:

\[
\pi^* = \arg \max_\theta \mathbb{E}_{s \sim P(S|\theta)} r(s, \pi_\theta(s))
\]

Assume learning \( r \) is statistically easier than directly learning \( \pi^* \)
# Types of Imitation Learning

<table>
<thead>
<tr>
<th></th>
<th>Direct Policy Learning</th>
<th>Reward Learning</th>
<th>Access to Environment</th>
<th>Interactive Demonstrator</th>
<th>Pre-collected Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioral Cloning</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Direct Policy Learning (interactive IL)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Optional</td>
</tr>
<tr>
<td>Inverse Reinforcement Learning</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Today’s itinerary

- Intro to Imitation Learning
- Behavioral Cloning
- Imitation Learning with Interactive Experts
- Inverse RL (MMP, Max Ent IRL)
- Learning from other sources of data (preferences, physical feedback)
Interactive Direct Policy Learning

Behavioral Cloning is simplest example

Beyond BC: using interactive demonstrator

Often analyzed via learning reductions
  • Reduced “harder” learning problem to “easier” one
  • Imitation Learning $\rightarrow$ Supervised Learning
Learning Reductions

Behavioral Cloning:

\[
\mathbb{E}_{s \sim p(s|\theta)} L(a^*(s), \pi_\theta(s)) \rightarrow \mathbb{E}_{(s,a^*) \sim P^*} L(a^*, \pi_\theta(s))
\]

A: General Imitation Learning  
B: Behavioral Cloning

What does learning well on B imply about A?  
- e.g., can one lift PAC learning results from B to A?
Direct Policy Learning via Interactive Expert (sequential learning reductions)

Sequence of distributions (sequence of supervised learning problems)

- $\mathbb{E}_{S \sim P(m)} L(\pi^*(S), \pi_\theta(S))$
- Ideally converges to $\pi_{OPT}$ (best in policy class)

Usually start from:

- $\mathbb{E}_{S \sim P} L(\pi^*(S), \pi_\theta(S))$

Requires Interactive Demonstrator (BC is a 1-step special case)
Interactive Expert

Can query expert at any state
Construct loss function: $L(\pi^*(s), \pi(s))$

• Typically applied to rollout trajectories of policies we are training: $s \sim P(s|\pi)$

• Driving example: $L(\pi^*(s), \pi(s)) = (\pi^*(s) - \pi(s))^2$

Expert provides feedback on state visited by policy
Alternating Optimization (Naïve Attempt)

1. Fix $P$, estimate $\pi$
   • Solve $\arg\min_\theta \mathbb{E}_{s \sim p} L(\pi(s), \pi_\theta(s))$

2. Fix $\pi$, estimate $P$
   • Empirically estimate via rolling out $\pi$

3. Repeat

**Not guaranteed to converge!**
Sequential Learning Reductions

• Initial predictor: $\pi_0$ (initial predictor: initial expert demonstrations)

• For $m$ sequence of predictors (initialize $m=1$)
  - Collect trajectories $\tau$ via rolling out $\pi_{m-1}$ (typically rollout multiple times)
  - Estimate state distribution $P_m$ using $s \in \tau$
  - Collect interactive feedback $\{\pi^*(s) | s \in \tau\}$ (requires interactive expert)

- **Data Aggregation** (e.g., DAgger)
  - Train $\pi_m$ on $P_1 \cup \cdots \cup P_m$

- **Policy Aggregation** (e.g., SEARN & SMILe)
  - Train intermediate policy $\pi'_m$ on only $P_m$
  - $\pi_m = \beta \pi'_m + (1 - \beta)\pi_{m-1}$ (geometric blending of policies)
DAgger in Practice
Direct Policy Learning via Interactive Expert

Reduction to sequence of supervised learning problems
  • Constructed from rollouts from previous policies
  • Requires interactive expert feedback

Two approaches: Data Aggregation & Policy Aggregation
  • Ensure convergence
  • Motivated by different theory

Not covered:
  • What is expert feedback and loss function? (depends on application)
Application of DAgger: Negotiation Domain

Shared Items $i$

1  2  2
Application of DAgger: Negotiation Domain
Application of DAgger: Negotiation Domain

Bob's Utility $u_B$
- 0 books, 1
- 4 hats, 4
- 1 balls

Alice's Utility $u_A$
- 2 books, 2
- 3 hats, 3
- 1 balls

Shared Items $i$
1 book, 2 hats, 2 balls

Bob proposes (0 books, 2 hats, 2 balls)
Application of DAgger: Negotiation Domain

Bob’s Utility $u_B$
- Book: 0
- Hat: 4
- Ball: 1

Alice’s Utility $u_A$
- Book: 2
- Hat: 3
- Ball: 1

Shared Items $i$
- Book: 1
- Hat: 2
- Ball: 2

Alice
- Agreed

Bob
- Proposed: 0 books, 2 hats, 2 balls

Supervised Learning
Application of DAgger: Negotiation Domain

Bob's Utility $u_B$

- Book: 0
- Hat: 4
- Ball: 1

Alice's Utility $u_A$

- Book: 2
- Hat: 3
- Ball: 1

Bob proposes: (0 books, 2 hats, 2 balls)

Alice agrees:

Bob insists: (1 book, 2 hats, 2 balls)
Application of DAgger: Negotiation Domain

Shared Items $i$

- 1 book
- 2 hats
- 2 balls

Bob

propose(0 books, 2 hats, 2 balls)

Alice

Agree

Supervised Learning

Targeted Acquisition

Reinforcement Learning

Bob’s Utility $u_B$

- 0 book
- 4 hats
- 1 ball

Alice’s Utility $u_A$

- 1 book
- 3 hats
- 1 ball
Optimal Dialogue Acts

Targeted Acquisition

\( \phi_1 \)

\( \phi_2 \)

SL

RL

RL+SL
Targeted Data Acquisition Framework

**Alice RL Training**

- `propose(0 books, 2 hats, 2 balls)`

**Alice**

- `insist(1 book, 2 hats, 2 balls)!`

**Negotiation n**

**Bob**

- *This looks novel!*
Targeted Data Acquisition Framework

\[
\sum_{x_t \in X^A} p_{\theta}(x_t | x_{0:t-1}, c^A)
\]

**Alice RL Training**
- propose(0 books, 2 hats, 2 balls)

**Alice**
- insist(1 book, 2 hats, 2 balls)!

**Bob**
- This looks novel!
Targeted Data Acquisition Framework

### Flag negotiation $n$

**Alice RL Training**

**Bob**

- propose(0 books, 2 hats, 2 balls)

**Alice**

- insist(1 book, 2 hats, 2 balls)!

**Bob**

- This looks novel!

**Negotiation $n$**

$$\sum_{x_t \in X_A} p_\theta(x_t | x_{0:t-1}, c^A)$$

**Expert Annotation**

**Bob**

- propose(0 books, 2 hats, 2 balls)

**Alice**

- insist(1 book, 2 hats, 2 balls)!

**Expert**

- end
Targeted Data Acquisition Framework

Alice RL Training

propose(0 books, 2 hats, 2 balls)

insist(1 book, 2 hats, 2 balls)!

This looks novel!

Expert Annotation

propose(0 books, 2 hats, 2 balls)

insist(1 book, 2 hats, 2 balls)!

end

Update Dataset
$\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'$

Bob SL Training

Flag negotiation $n$

$\sum_{x_t \in X^A} p_\theta(x_t|x_{0:t-1}, c^A)$
**Targeted Data Acquisition Framework**

### Alice RL Training
- **propose**: (0 books, 2 hats, 2 balls)
- **insist**: (1 book, 2 hats, 2 balls)!

### Expert Annotation
- **propose**: (0 books, 2 hats, 2 balls)
- **insist**: (1 book, 2 hats, 2 balls)!

### Update Dataset
\[ \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}' \]

### Bob SL Training

\[
\sum_{x_t \in X^A} p_{\theta}(x_t | x_{0:t-1}, c^A)
\]

### Continue Training Alice

Flag negotiation \( n \)
Example Negotiation with a Human

<table>
<thead>
<tr>
<th>Ours</th>
<th>RL</th>
<th>RL+SL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Human</strong>: propose: item0=1 item1=2 item2=1</td>
<td><strong>Human</strong>: propose: item0=2 item1=2 item2=1</td>
<td><strong>Alice</strong>: propose: item0=2 item1=0 item2=1</td>
</tr>
<tr>
<td><strong>Alice</strong>: propose: item0=1 item1=1 item2=1</td>
<td><strong>Alice</strong>: propose: item0=2 item1=1 item2=1</td>
<td><strong>Human</strong>: propose: item0=2 item1=3 item2=1</td>
</tr>
<tr>
<td><strong>Human</strong>: agree</td>
<td><strong>Human</strong>: agree</td>
<td><strong>Alice</strong>: agree</td>
</tr>
<tr>
<td><strong>Alice</strong>: agree</td>
<td><strong>Alice</strong>: &lt;selection&gt;</td>
<td><strong>Human</strong>: agree</td>
</tr>
<tr>
<td><strong>Human</strong>: book=1 hat=2 ball=0</td>
<td><strong>Human</strong>: book=0 hat=2 ball=0</td>
<td><strong>Alice</strong>: item0=0 item1=0 item2=0</td>
</tr>
<tr>
<td><strong>Alice</strong>: item0=1 item1=1 item2=1</td>
<td><strong>Alice</strong>: item0=2 item1=1 item2=1</td>
<td><strong>Human</strong>: book=2 hat=3 ball=1</td>
</tr>
<tr>
<td>Agreement!</td>
<td>Agreement!</td>
<td>Agreement!</td>
</tr>
<tr>
<td>Alice : 8 points</td>
<td>Alice : 10 points</td>
<td>Alice : 0 points</td>
</tr>
<tr>
<td>Human : 6 points</td>
<td>Human : 4 points</td>
<td>Human : 10 points</td>
</tr>
</tbody>
</table>

Alice : book=(count:2 value:2) hat=(count:3 value:0) ball=(count:1 value:6)
Human : book=(count:2 value:2) hat=(count:3 value:2) ball=(count:1 value:0)
Today’s itinerary

• Intro to Imitation Learning
• Behavioral Cloning
• Imitation Learning with Interactive Experts
• Inverse RL (MMP, Max Ent IRL)
• Learning from other sources of data (preferences, physical feedback)
Types of Imitation Learning

Behavioral Cloning

\[
\arg \min_{\theta} \mathbb{E}_{(s,a^*) \sim p^*} L(a^*, \pi_\theta(s))
\]

Works well when \( P^* \) is close to \( P_\theta \)

Direct Policy Learning (via Interactive Demonstrator)

Requires Interactive Demonstrator (BC is a 1-step special case)

Inverse RL

Learn \( r \) such that:

\[
\pi^* = \arg \max_{\theta} \mathbb{E}_{s \sim P(s \mid \theta)} r(s, \pi_\theta(s))
\]

Assume learning \( r \) is statistically easier than directly learning \( \pi^* \)
What can go wrong with policy learning?

Behavioral cloning: mimics the expert directly
- No reasoning about outcomes or dynamics
- No notion of intentions
- Expert can be suboptimal
- Expert might have different embodiments
- Safety and Robustness
History of Inverse Reinforcement Learning

- 1964: Kalman posed the inverse optimal control problem and solved it in 1D
- 1994: Boyd et al. A linear matrix inequality (LMI) characterization for the linear quadratic setting
- 2000: Ng, Russell. Proposed the first MDP formulation and issues around reward function ambiguity
- 2004: Abbeel, Ng. Inverse RL with feature matching for apprenticeship learning
- 2006: Ratliff et al. Max Margin Planning (MMP) Formulation
- 2008: Zeibart et al. Max Entropy Formulation

- Since then... Active Inverse RL, Integration with other types of data, Iterative approaches to update Reward and Policy (GAIL, etc.), images as inputs, etc.
Apprenticeship Learning

[Abbeel, Ng, 2004]
Problem Setup: Behavioral Cloning

MDP with no reward functions:
- State space, $S$ (sometimes partially observable)
- Actions space, $A$
- An expert policy $\pi^*$ that maps states to distributions over actions: $\pi^*(s) \rightarrow P(s)$
- Transition model $P(s_{t+1}|s_t, a_t)$: simulator or environment

**Goal:** Learn an imitating policy $\pi_\theta(s)$ that imitates the expert demonstrations
Problem Setup: Inverse RL

MDP with no reward functions:
- State space, $S$ (sometimes partially observable)
- Actions space, $A$
- An expert policy $\pi^*$ that maps states to distributions over actions: $\pi^*(s) \rightarrow P(s)$
- Transition model $P(s_{t+1}|s_t, a_t)$: simulator or environment

**Goal:** Learn an imitating policy $\pi_{\theta}(s)$ that imitates the expert demonstrations

**Goal:** Learn a reward function assuming the experts are optimal
Inverse Reinforcement Learning

Assume the reward function is a linear combination of features:

$$R(s) = w^T \varphi(s)$$

$$w \in \mathbb{R}^n$$

$$\varphi: S \rightarrow \mathbb{R}^n$$

(a) Features for the boundaries of the road
(b) Feature for staying inside the lanes.
(c) Features for avoiding other vehicles.
Inverse Reinforcement Learning

Assume the reward function is a linear combination of features:

\[ R(s) = w^T \varphi(s) \quad w \in \mathbb{R}^n \quad \varphi: S \to \mathbb{R}^n \]

The goal is to recover the weights: \( w \)

\[
\begin{align*}
V^\pi(s) &= \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \right] \\
&= \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t w^T \varphi(s_t) \right] = w^T \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \varphi(s_t) \right] = w^T \mu(\pi)
\end{align*}
\]
How to deal with reward ambiguity?

**Reward ambiguity:** There are many reward functions under which the expert demonstrations are optimal!!
How to deal with reward ambiguity?

**Reward ambiguity:** There are many reward functions under which the expert demonstrations are optimal!!

Which reward function should we pick?

- **Maximum Margin Planning:** Looks for the one that separates the optimal policy best.
Aside: Maximum Margin Classifiers

Given a training dataset of \((x_1, y_1), \ldots, (x_n, y_n)\), where \(y_i\) is either 1 or -1 identifying the class \(x_i\) is in. We want to find the maximum margin hyperplane that divides the points so the distance between the hyperplane and the nearest point from each class is maximized.

"Minimize \(\|\vec{w}\|\) subject to \(y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1\), for \(i = 1, \ldots, n\)"
How to deal with reward ambiguity?

Reward ambiguity: There are many reward functions under which the expert demonstrations are optimal!!

Which reward function should we pick?

- **Maximum Margin Planning**: Looks for the one that separates the optimal policy best.

- **Maximum Entropy IRL**: Looks for the one where expert demonstrations are drawn from a high entropy distribution.