Formulate as a Markov Decision Process (MDP)

\[
\langle S, A, R, T \rangle
\]
\[ S: \]

\[ A: \]

\[ R: \]

\[ T(s' | s, a). \]
How do we estimate state from observation?

RGBD Image (Observation $Z_b$)

We don't see state, we see observations.

State
- Pose of objects
- Configuration of robot

Observation
- RGBD Image
- Encoders on joints
- Force-torque sensor
- IMU
How do we estimate 3D pose of objects from RGBD image?

**Step 1**

Train an instance segmentator (Mask-RCNN)

To propose:

- **Bounding Box**
- **Mask**
- **Object**
- **Class**

---

How?

- **CNN**
- **Region Proposal Network**
- **Mask**
**Step 2**

Extract 3D Point Cloud from 2D Image

**Step 3**

Train Point Net to Predict Accurate 6DOF Pose from Point Cloud

3D Point Cloud → Point Net → $X, Y, Z, \phi, \theta, \psi$
Let's take stock: We know state, we know actions.

\( \langle S, A, R, T \rangle \)

Goal: Learn a policy \( \pi \) that maps state to actions.

What is the simplest approach?
Behavior Cloning

Step 1: Human provides demonstrations

1.  
2.  
3.  
4.  

\[ D = \left\{ \begin{array}{c} (s_1^*, a_1^*, s_2^*, a_2^*, \ldots, s_T^*, a_T^*) \\ (s_1^*, a_1^*, s_2^*, a_2^*, \ldots, s_T^*, a_T^*) \\ (s_1^*, a_1^*, s_2^*, a_2^*, \ldots, s_T^*, a_T^*) \end{array} \right\} \]

Step 2: Train a policy to map states to actions

\[ \arg\min_{\pi} \mathbb{E} \ell(a^*, \pi(s^*)) \]

How do we implement this in practice?

What is the loss for discrete actions?

Continuous actions?
Step 3: Check Validation Loss

Train Loss  

Val Loss

How can BC fail?

Overfits!
DAGGER

Insight: Ask the human expert for "corrections" on states the robot visits.

\[ \arg\min_{\pi} \mathbb{E} \left[ l(\pi^*(s), \pi(s)) \right] \]

DAGGER Algorithm

\[ \pi_0 \leftarrow \text{initialize policy with behavior cloning.} \]

D \leftarrow \{ \}

\text{Initialize empty data buffer}

\text{For } i = 1 \ldots N

\text{Rollout } \pi_i

\left( s_1, a_1, s_2, a_2, \ldots \right)
Query human $x^*$ for correct actions

$(s_1, x^*(s_1), s_2, x^*(s_2), \ldots)$

$D \leftarrow D \cup \{(s_1, x^*(s_1), s_2, x^*(s_2), \ldots)\}$

$x_i \leftarrow \text{Train}(D)$

Practical issues with DAGGER?
Ask human to design reward function.

$\langle S, A, R, T \rangle$

How can we train a policy $\pi$ by interacting with world $T(s'|s,a)$?

**Policy Gradient**

Idea: Rollout policies, estimate value, update.

$$\arg\max_{\pi} \mathbb{E}_{s \sim \pi, t \sim \tau_{\theta}} \left[ \sum_{t=1}^{T} \tau(s_t, a_t) \right]$$
Policy Gradient Theorem

$$\dot{\theta} = \theta + \eta \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \hat{Q}(s_t, a_t) \right]$$
**REINFORCE**

\[ \tau_0 \leftarrow \text{INITIALIZE POLICY} \]

For \( i = 1 \ldots N \)

**Rollout** \( \tau_i \):

\[
(s_1, a_1, r_1, s_2, a_2, r_2, \ldots)
\]

**Compute** \( \hat{Q} : (\text{REWARD-TO-GO}) \)

\[
(s_1, a_1, \hat{Q}_1, s_2, a_2, \hat{Q}_2, \ldots)
\]

**Update policy** \( \pi_\theta \):

\[
\theta \leftarrow \theta + \eta \left[ \sum_{t=1}^{T} \nabla_\theta \log \tau_\theta(a_t | s_t) \hat{Q}(s_t, a_t) \right]
\]

Problems with REINFORCE?
Why is MODEL-FREE RL CHALLENGING for ROBOTICS?
What if we learn $T(s' | s, a)$?

$\langle S, A, R, T \rangle$

What happens if we plan with a model learned only from human demonstration?
Performance Difference

\[ J(\frac{a}{x}) - J(\frac{ax}{M^*}) \]

Policy That Planner Computes

Optimal Policy

Real World

Model Loss on States Learner Visits

Model Loss on States Expert Visits

Suboptimality of Your Planner

against sys identifying
DAGGER for Model Based RL

Given: Data from an expert policy \( \pi^* \)
\[ D_{exp} = \{(s_i, a_i, s'_i)\}_{i=1}^N \]

Init: Initialize a model \( \hat{H}(s, a) \rightarrow s' \)

\[ L(H) = \frac{1}{100} \sum_{s \in D_{exp}} \| \hat{H}(s, a(s)) - s' \|^2 \]
Optimize loss

\[ D_{learner} = \{s_i\} \# Initialize learner data
\]
\[ \text{for } i = 1 \ldots N \] (\# N is rounds of data collect)

\# Call a planner on my model \( \pi_i \)
\[ \hat{\pi}_i = \arg\min_{\pi \in \Pi} J(\pi) \]

\# Collect data by running policy \( \hat{\pi}_i \) in real world.
\[ D_{learner}^i = \{(s_i, a_i, s'_i)\}_{i=1}^N \]

\# Aggregate learner data.
\[\mathcal{D}_{\text{learn}} = \mathcal{D}_{\text{learn}} \cup \mathcal{D}^{i}_{\text{learn}}\]

\# Train Model.

\[L_1(\mathcal{H}) = \frac{1}{|\mathcal{D}_{\text{learn}}|} \sum_{(s,a,s') \in \mathcal{D}_{\text{learn}}} ||\hat{M}(s,a) - s'||^2\]

\[L_2(\mathcal{H}) = \frac{1}{|\mathcal{D}_{\text{exp}}|} \sum_{(s,a,s') \in \mathcal{D}_{\text{exp}}} ||\hat{M}(s,a) - s'||^2\]

\# Cost New Model

\[\hat{M}_{\text{th}} = \text{Optimize} \left( L_1(\mathcal{H}) + L_2(\mathcal{H}) \right)\]

\[\text{Policy}\]

\[s \rightarrow 0 \rightarrow a\]

\[\text{Model}\]

\[s \rightarrow 0 \rightarrow s'\]

\[a \rightarrow 0 \rightarrow 0\]