

Imitation Learning

Rohan Paul

Acknowledgements

These slides contain figures from tutorial material by Chesea Finn and Peter Abeel. Material is also derived from recent papers that have been referenced. This material is intended only for tutorial purposes in the COL864 course.

Policy Gradients

Training Algorithm

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward

log-likelihood of action

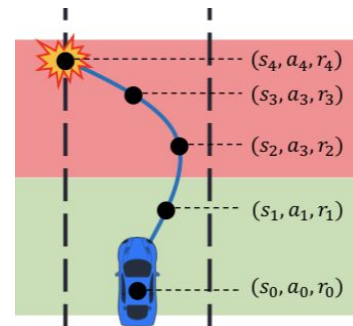
$$\text{loss} = -\log P(a_t | s_t) R_t$$

reward

Gradient descent update:

$$w' = w - \nabla \text{loss}$$
$$w' = w + \nabla \log P(a_t | s_t) R_t$$

Policy gradient!

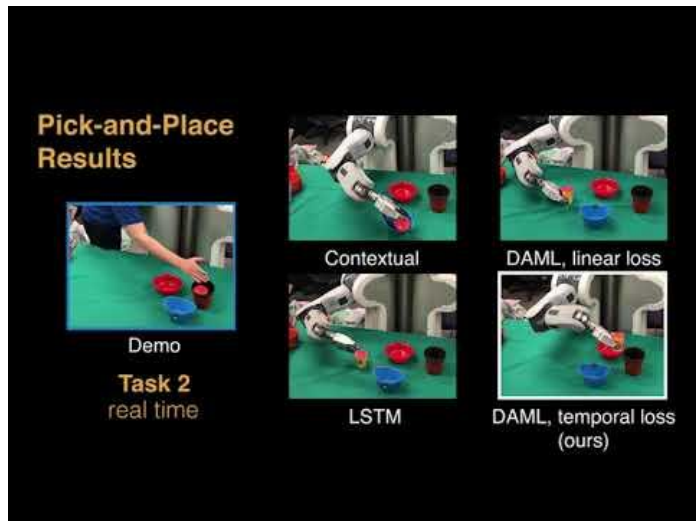


Reward functions for complex real world tasks are hard to engineer

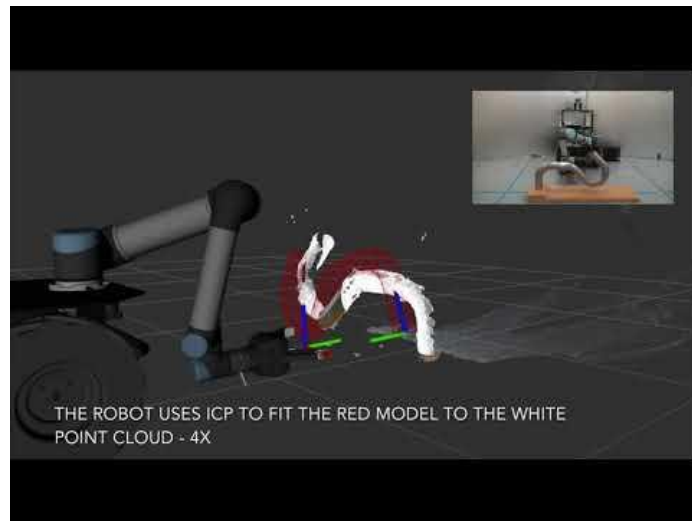


- Reward functions communicate our intent or goal to the agent.
- Simply demonstrating the task is easier than engineering a complex reward function

Learn from Human Demonstration



C. Finn et al.

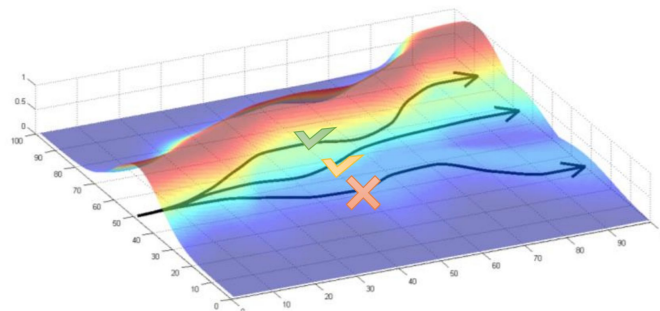
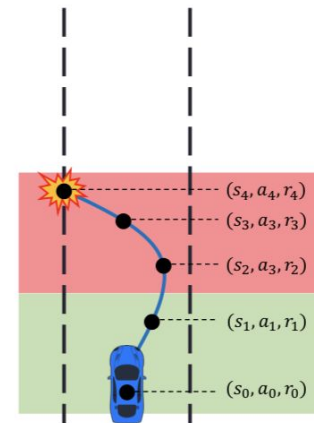


D. Park and R. Paul et al.

- Learning from demonstration improves the data efficiency of standard RL algorithms.
- Can learn from smaller amounts of data.

Sparse Rewards are Challenging in RL

- Rewards that are dense in time can closely guide the agent.
- Implicitly specify them using demonstrations.
- **Demonstrations as a form of reward shaping.**



PG works better with reward shaping

Imitation Learning Successes

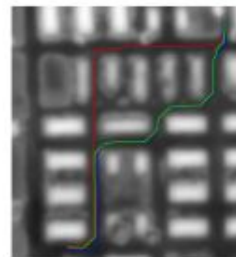
Simulated highway driving

- Abbeel and Ng, ICML 2004
- Syed and Schapire, NIPS 2007
- Majumdar et al., RSS 2017



Aerial imagery-based navigation

- Ratliff, Bagnell, and Zinkevich, ICML 2006



Parking lot navigation

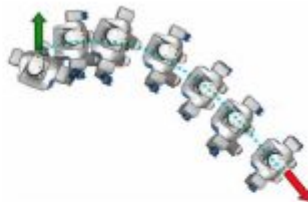
- Abbeel, Dolgov, Ng, and Thrun, IROS 2008



Imitation Learning Successes

Human path planning

- Mombaur, Truong, and Laumond, AURO 2009



Human goal inference

- Baker, Saxe, and Tenenbaum, Cognition 2009



Quadruped locomotion

- Ratliff, Bradley, Bagnell, and Chestnutt, NIPS 2007
- Kolter, Abbeel, and Ng, NIPS 2008



Learning From Demonstrations

- Expert provides a set of demonstration trajectories: sequences of states and actions
- Imitation learning is useful when is easier for the expert to demonstrate the desired behavior rather than:
 - come up with a reward that would generate such behavior,
 - coding up the desired policy directly

Imitation Learning or Learning from Demonstration

- Input:
 - State space, action space
 - Transition model $P(s' | s, a)$
 - No reward function R
 - Set of one or more teacher's demonstrations $(s_0, a_0, s_1, s_0, \dots)$
(actions drawn from teacher's policy π^*)
- Behavioral Cloning:
 - Can we directly learn the teacher's policy using supervised learning?
- Inverse RL:
 - Can we recover R ?
- Apprenticeship learning via Inverse RL:
 - Can we use R to generate a good policy?

Behaviour Cloning

- Simply mimic the teacher's actions.
- Reduction to supervised learning
 - Given the state action pairs from the demonstrator, learn to predict the same action as the expert
- Minimise the 1-step deviation

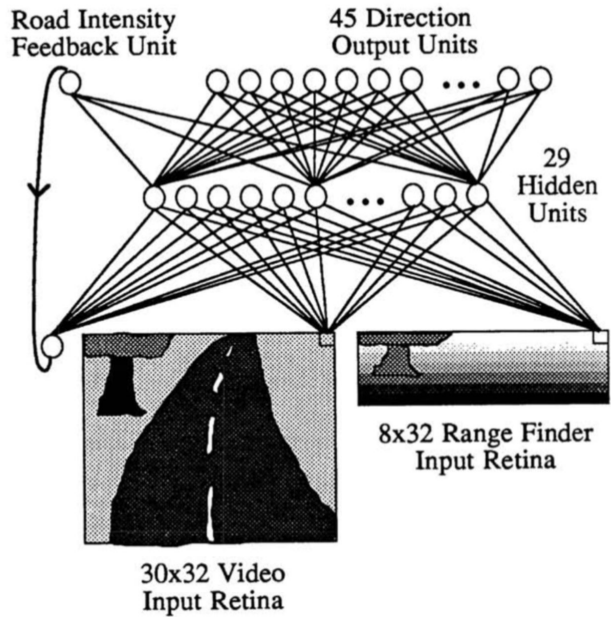
Learning objective:

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

Behaviour Cloning

- Formulate problem as a standard machine learning problem:
 - Fix a policy class (e.g. neural network, decision tree, etc.)
 - Estimate a policy from training examples $(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots$
- Two notable success stories:
 - Pomerleau, NIPS 1989: ALVINN
 - Summut et al., ICML 1992: Learning to fly in flight simulator

Behaviour Cloning: ALVINN

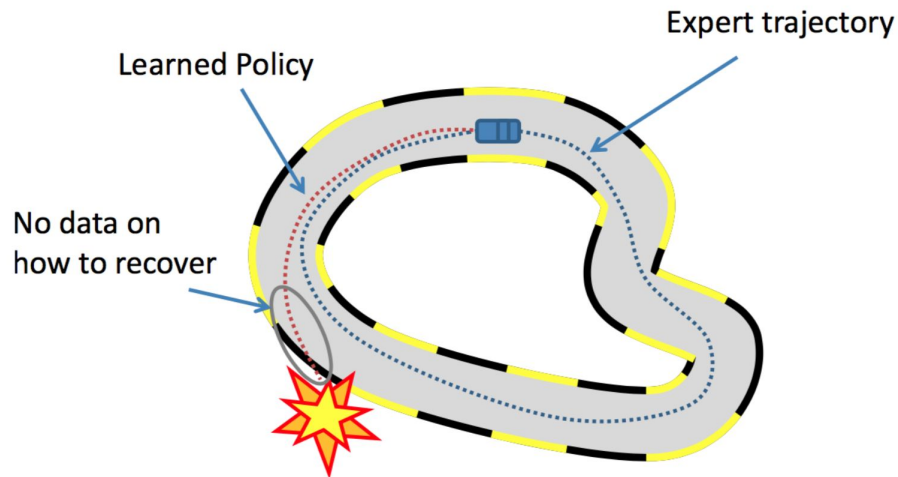


Behaviour Cloning: Problems



- Samples are from the teacher's distribution.
- What the agent experiences are states from rollouts of its own policy.
- Catastrophic failures

Behaviour Cloning: Why catastrophic failures?



Data distribution mismatch!

In supervised learning, $(x, y) \sim D$ during train **and** test. In MDPs:

- Train: $s_t \sim D_{\pi^*}$
- Test: $s_t \sim D_{\pi_\theta}$

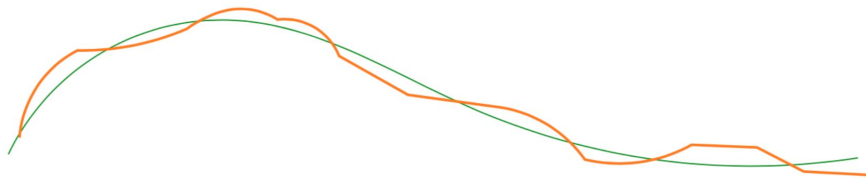
Behaviour Cloning: Why catastrophic failures?

Supervised learning approach assumes that the training and the test distributions are the same.

Expected number of total errors grows as T

Now apply this supervised learning setup to the MDP or RL context.

Supervised learning assumes iid. (s, a) pairs and ignores temporal structure
Independent in time errors:



Error at time t with probability ϵ

$$\mathbb{E}[\text{Total errors}] \leq \epsilon T$$

Behaviour Cloning: Why catastrophic failures?

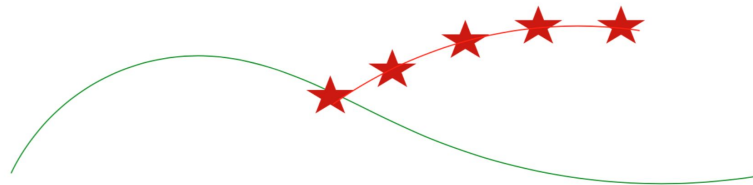
Now, consider the error in the RL context, where the actions determine the distribution of states the agent will be encountering.

The errors compound (navigation example).

An error can lead the agent to parts of the state space where it hasn't been trained on. Hence, it will make more errors.

At any time step will make T more errors.

Hence, the error overall grows as T^2 .



Error at time t with probability ϵ

$$\mathbb{E}[\text{Total errors}] \leq \epsilon(T + (T - 1) + (T - 2) \dots + 1) \propto \epsilon T^2$$

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011



Sequence of states experienced in an MDP depends on the actions taken while executing a policy.

Problem

- Actually, we need data in rollouts experienced by the agent's policy.
- Otherwise can't recover from errors made the by agent's policy.
- *What if the agent could query for data?*

DAGGER (Data set Aggregation)

- Interactive expert: that can give labels for any state the agent experiences.
- Idea:
 - Get more labels of the expert action in the new states along the policy computed by BC.
- Just keep adding the data.
 - Essentially train from all past mistakes.
 - Perform policy blending.
- Obtains a stationary deterministic policy with good performance under its induced state distribution.

Initialize $\mathcal{D} \leftarrow \emptyset$.

Initialize $\hat{\pi}_1$ to any policy in Π .

for $i = 1$ **to** N **do**

 Let $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$.

 Sample T -step trajectories using π_i .

 Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.

 Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$.

 Train classifier $\hat{\pi}_{i+1}$ on \mathcal{D} .

end for

Return best $\hat{\pi}_i$ on validation.

Essentially, a reduction to sequential learning problems.

Autonomous Flight using DAGGER



Behaviour Cloning: Limitations

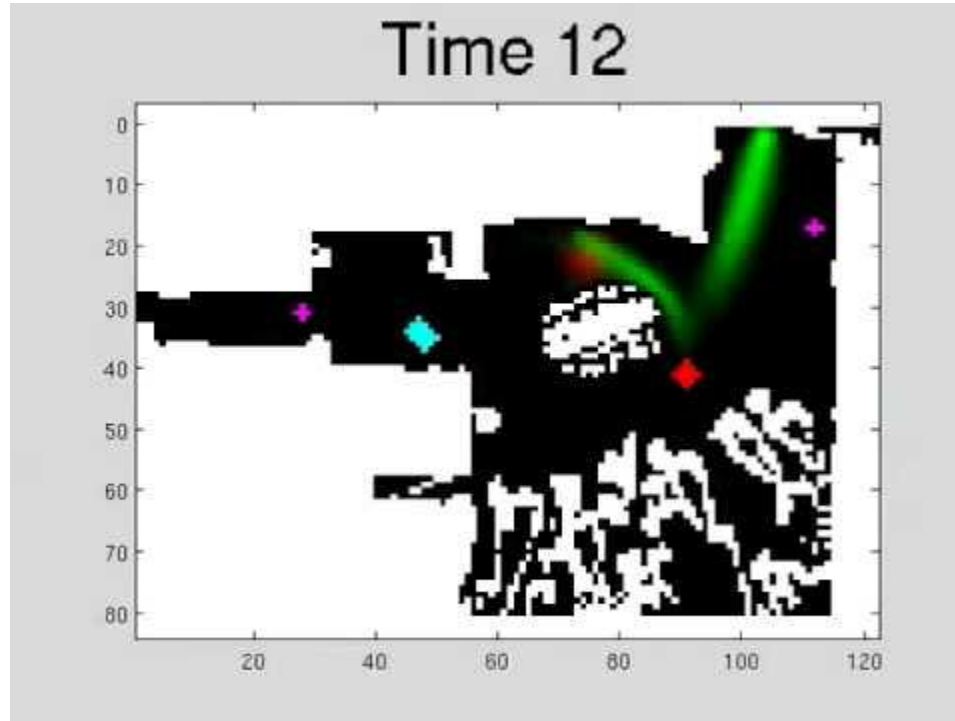
- Essentially trying to “mimic” the expert/teacher.
- **No notion of the expert’s goal or intention.**
 - Agent does not know what the teacher is trying to teach and hence cannot generalize.
 - If the agent can infer what the teacher is doing, it can potentially do better than the demonstrator.
- Need for “optimal” demonstrations from the expert

Inverse RL aims to recover the reward structure that the teacher is using.
Inherently, tries to recover the goal that the teacher is using.

Understanding intent is an innate ability



Intent Inference can help the agent behave naturally



Inverse Optimal Control / Inverse Reinforcement Learning:

infer reward function from demonstrations

(IOC/IRL)

(Kalman '64, Ng & Russell '00)

given:

- state & action space
- Roll-outs from π^*
- dynamics model (sometimes)

goal:

- recover reward function
- then use reward to get policy

IRL is an under-defined problem

- Many reward functions can explain a demonstration.

Challenges

underdefined problem

difficult to evaluate a learned reward

demonstrations may not be precisely optimal



Maximum Entropy Inverse RL

(Ziebart et al. '08)

handle ambiguity using probabilistic model of behavior

Notation:

$\tau = \{s_1, a_1, \dots, s_t, a_t, \dots, s_T\}$
trajectory

$R_\psi(\tau) = \sum_t r_\psi(s_t, a_t)$
learned reward

$\mathcal{D} : \{\tau_i\} \sim \pi^*$
expert demonstrations

MaxEnt formulation:

Probability of a state-action trajectory given the reward function.

$$p(\tau) = \frac{1}{Z} \exp(R_\psi(\tau))$$

Do maximum likelihood estimate for the reward parameters. Maximize the likelihood of the expert trajectories given the reward model parameterized by psi.

$$\max_{\psi} \sum_{\tau \in \mathcal{D}} \log p_{r_\psi}(\tau)$$

$$Z = \int \exp(R_\psi(\tau)) d\tau$$

(energy-based model for behavior)

Maximum Entropy IRL Optimization

$$\max_{\psi} \mathcal{L}(\psi) = \sum_{\tau \in \mathcal{D}} \log p_{r_{\psi}}(\tau)$$

How do you optimize for the parameters?

Perform gradient descent

$$= \sum_{\tau \in \mathcal{D}} \log \frac{1}{Z} \exp(R_{\psi}(\tau))$$

$$= \sum_{\tau \in \mathcal{D}} R_{\psi}(\tau) - M \log Z$$

$$= \sum_{\tau \in \mathcal{D}} R_{\psi}(\tau) - M \log \sum_{\tau} \exp(R_{\psi}(\tau))$$

$$\nabla_{\psi} \mathcal{L}(\psi) = \sum_{\tau \in \mathcal{D}} \frac{dR_{\psi}(\tau)}{d\psi} - M \frac{1}{\sum_{\tau} \exp(R_{\psi}(\tau))} \sum_{\tau} \exp(R_{\psi}(\tau)) \frac{dR_{\psi}(\tau)}{d\psi}$$

Maximum Entropy IRL Optimization

$$\nabla_{\psi} \mathcal{L}(\psi) = \sum_{\tau \in \mathcal{D}} \frac{dR_{\psi}(\tau)}{d\psi} - M \underbrace{\frac{1}{\sum_{\tau} \exp(R_{\psi}(\tau))} \sum_{\tau} \exp(R_{\psi}(\tau))}_{\sum_{\tau} p(\tau | \psi)} \frac{dR_{\psi}(\tau)}{d\psi}$$

The gradient can be reformulated in terms of the state visitation probabilities.

$P(\tau | \psi)$ is the probability of visiting a state under a reward function parameterized by ψ

There is a dynamic programming algorithm to obtain the state visitation.

$$\sum_{\tau} p(\tau | \psi) \frac{dR_{\psi}(\tau)}{d\psi}$$

$$\sum_{\mathbf{s}} p(\mathbf{s} | \psi) \frac{dr_{\psi}(\mathbf{s})}{d\psi}$$


Intuition:
IRL is trying to match the features of the demonstration. That is match the state visitation frequencies.

The agent should be visiting states in the same frequency as what the expert is doing.

Maximum Entropy Inverse RL

(Ziebart et al. '08)

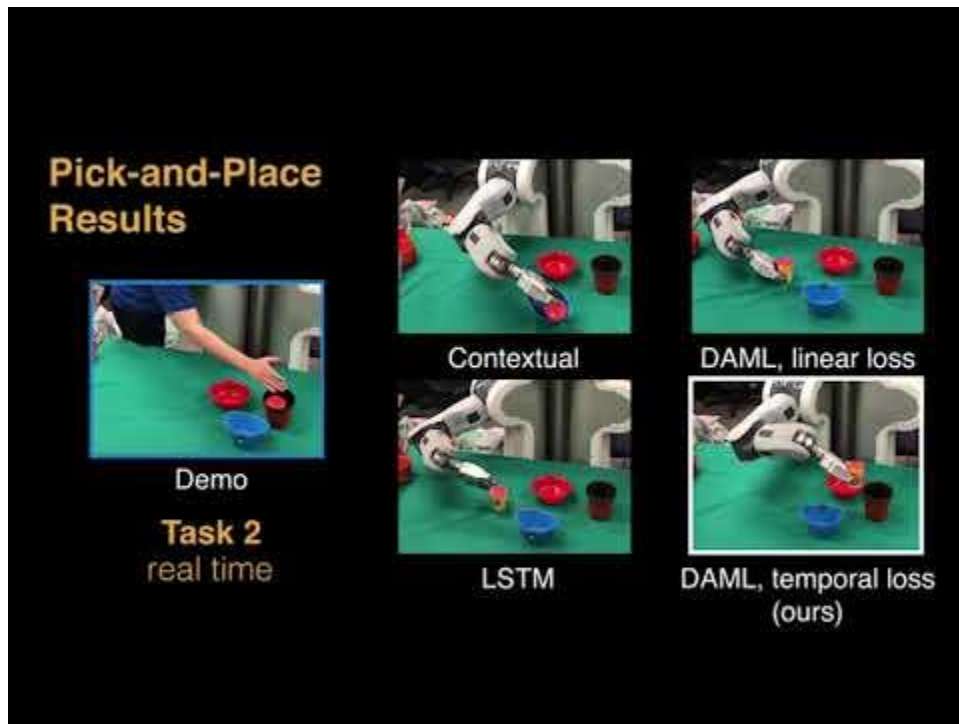
handle ambiguity using probabilistic model of behavior

0. Initialize ψ , gather demonstrations \mathcal{D}
 1. Solve for optimal policy $\pi(\mathbf{a}|\mathbf{s})$ w.r.t. reward r_ψ
 2. Solve for state visitation frequencies $p(\mathbf{s}|\psi)$
 3. Compute gradient $\nabla_\psi \mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{\tau_d \in \mathcal{D}} \frac{dr_\psi}{d\psi}(\tau_d) - \sum_s p(s|\psi) \frac{dr_\psi}{d\psi}(s)$
 4. Update ψ with one gradient step using $\nabla_\psi \mathcal{L}$
- 

MaxEnt - IRL

- Model the distribution over trajectories for a parameterized reward function.
- Find the most likely estimate of the reward function that explains the demonstration.
- **Key Idea is Feature matching.** Find the reward function under which a policy rollout will lead to state visitations that will be similar to the expert's feature distribution.

Skill learning from demonstration



Finn et al. ICML '16. *Guided Cost Learning*. Sampling based method for MaxEnt IRL that handles unknown dynamics and deep reward functions