AA203 Optimal and Learning-based Control

Imitation Learning





The basics of Imitation Learning

- Assume access to a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{u}_n)\}_{n=1}^N$ of state-control pairs collected by an expert
- At a high-level, approaches to IL belong to two main categories:
 - Behavior Cloning
 - Inverse Reinforcement Learning (a.k.a. Inverse Optimal Control)

"learn the policy used by expert" "learn the objective optimized by the expert"

```
Behavior Cloning \pi_{\theta}(\mathbf{u}|\mathbf{x})
```

Inverse Reinforcement Learning $R: \mathcal{X} imes \mathcal{U} o \mathbb{R}$

Outline

- Imitation Learning:
 - Behavior Cloning (BC)
 - Common pitfalls
 - Design strategies for effective IL
 - Other paradigms (RvS, Inverse RL)
 - Inverse Reinforcement Learning (IRL)

Recap: BC as "supervised learning of behavior"

1. SL: learn a (parametric) mapping from inputs **x** to outputs **y** by minimizing a measure of prediction error

$$\mathbf{x} = \bigcup_{\substack{\mathbf{y} \in \mathcal{X} \to \mathcal{Y} \\ \text{channel}}} \mathcal{Y} = \{0, 1, \dots, C\}$$
$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N \quad \text{e.g., mean squared error (MSE)}$$
$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \quad \mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} ||\mathbf{y} - \hat{\mathbf{y}}||_2^2, \text{ where } \hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

- 2. Skeleton of a BC algorithm:
 - Collect a dataset of "expert" demonstrations
 - Train policy $\pi_{\theta}(\mathbf{u}|\mathbf{x})$ to mimic expert:

$$\mathcal{L} = rac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{u})\in\mathcal{D}} \lVert \mathbf{u} - \hat{\mathbf{u}}
Vert_2^2, \hspace{1em} ext{where} \hspace{1em} \hat{\mathbf{u}} = \pi_ heta(\mathbf{x})$$

• Deploy

٠

 $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{u}_n)\}_{n=1}^N$

Is this it?

Today:

- What could go wrong?
- Strategies to address these challenges

Common Pitfalls: 1) Compounding Errors

In statistical learning theory, data is assumed to be independent and identically distributed (i.i.d) $p(\mathbf{x})$

When learning behavior policies however:

- Predicted controls influence future states
- Even small errors can lead to drift away from the data distribution
- Errors compound



This leads to the *state distributions* under the expert and the learner to diverge

 $p_{ ext{expert}}(\mathbf{x})
eq p_{\pi_{ heta}}(\mathbf{x})$

"covariate shift"

Probability of making mistakes grows quadratically with the length of the trajectory $\mathcal{O}(\epsilon T^2)$

[Ross & Bagnell '10]

 \mathbf{X}_3

 \mathbf{X}_2

Common Pitfalls: 2) Multimodal behavior

- In many tasks, there exist multiple (equivalent) solutions
- In IL, this becomes relevant the moment the dataset is characterized by multimodal demonstrations

e.g., fitting multimodal behavior using MSE

• This is extremely common in practice, e.g., data collected by multiple experts



A useful mental model

We can address these challenges in a few different ways:

- Targeted algorithms
- Smart data collection (and augmentation)
- Use expressive models (e.g., that are able to capture the multimodality of behavior data)



A useful mental model

We can address these challenges in a few different ways:

- Targeted algorithms
- Smart data collection (and augmentation)
- Use expressive models (e.g., that are able to capture the multimodality of behavior data)



The value of corrective data

- A lot of the methods that make naïve BC work try to leverage the fact that, during deployment, we can recover from mistakes
 - Modify the learning problem to allow BC to learn how to correct from mistakes
- Paradox:
 - Learning from a broader set of trajectories that make some mistakes (and recoveries) is likely to work better than learning from a narrow set of perfect demonstration data



Learning from interventions: DAgger

DAgger is a well-known method for addressing the covariate shift problem through data aggregation techniques

DAgger seeks to reduce the covariate shift by explicitly gathering expert demonstrations under the state distribution induced by the learner.



At its core, DAgger defines an iterative procedure: Algorithm 22.3: DAGGER Algorithm Data: Initial dataset of expert demonstrations, \mathcal{D} , initial policy, π^1_{θ} , number of iterations, N Recently Twined policy, π^N

Result: Trained policy, π_{θ}^{N} for i = 1, 2, ..., N do Collect trajectories, $\tau = \{(x_t, u_t^{\text{learner}})\}$, using the policy π_{θ}^{i} . Gather dataset of states visited by the learner and actions given by the expert, $\mathcal{D}^{i} = \{(x_t, u_t^{\text{expert}})\}$. Aggregate the dataset, $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}^{i}$. Train the policy π_{θ}^{i+1} on the updated dataset, \mathcal{D} . **return** Trained policy, π_{θ}^{N} .

Learning from interventions: DAgger

Essentially, DAgger operates as a form of iterative supervised learning:

 Reduces the amount of expert data required (i.e., the alternative would be collecting an extremely broad dataset of demonstrations)



Many algorithms have been developed that follow a similar scheme:

- Confidence-based methods: "query the expert when the agent is uncertain about its decisions"
- Human-gated DAgger: "query the expert when the policy makes a mistake"



A useful mental model

We can address these challenges in a few different ways:

- Targeted algorithms
- Smart data collection (and augmentation)
- Use expressive models (e.g., that are able to capture the multimodality of behavior data)



Case Study 1

End to End Learning for Self-Driving Cars

Mariusz Bojarski NVIDIA Corporation Holmdel, NJ 07735 Davide Del Testa NVIDIA Corporation Holmdel, NJ 07735

Daniel Dworakowski NVIDIA Corporation Holmdel, NJ 07735 Bernhard Firner NVIDIA Corporation Holmdel, NJ 07735

Beat Flepp NVIDIA Corporation Holmdel, NJ 07735 Prasoon Goyal NVIDIA Corporation Holmdel, NJ 07735

Lawrence D. Jackel NVIDIA Corporation Holmdel, NJ 07735 Mathew Monfort NVIDIA Corporation Holmdel, NJ 07735

Urs Muller NVIDIA Corporation Holmdel, NJ 07735

Jiakai Zhang NVIDIA Corporation Holmdel, NJ 07735

Xin Zhang NVIDIA Corporation Holmdel, NJ 07735 Jake Zhao NVIDIA Corporation Holmdel, NJ 07735

Karol Zieba NVIDIA Corporation Holmdel, NJ 07735



Case Study 1



Bojarski et al. '16, NVIDIA

An important detail





Case Study 2

A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

Alessandro Giusti¹, Jérôme Guzzi¹, Dan C. Cireşan¹, Fang-Lin He¹, Juan P. Rodríguez¹ Flavio Fontana², Matthias Faessler², Christian Forster² Jürgen Schmidhuber¹, Gianni Di Caro¹, Davide Scaramuzza², Luca M. Gambardella¹



Case Study 2







AA 203 | Lecture 15

Key idea: Being intentional with data collection





Intentionally add **mistakes** and **corrections** to your data:

- Augment your collection pipeline with "fake" demonstrations of corrections (e.g., side cameras)
- Use algorithms that automate the collection of correction data where it matters more (e.g., DAgger)

A useful mental model

We can address these challenges in a few different ways:

- Targeted algorithms
- Smart data collection (and augmentation)
- Use expressive models (e.g., that are able to capture the multimodality of behavior data)



Improving the expert's predictions

At a high level, two main classes of methods:





E.g., mixture of Gaussians, conditional VAEs, diffusion models, etc.

Non-Markovian behavior

5/19/2025

Often, the behavior observed in demonstration data does not depend only on the current observation

$$\pi(\mathbf{u}_t|\mathbf{x}_t) \qquad \quad \pi(\mathbf{u}_t|\mathbf{x}_t,\mathbf{x}_{t-1},\ldots,\mathbf{x}_{t-H})$$

Idea: represent the policy via any sequence model (e.g., Transformers, LSTM/GRU cells, etc.)



Capturing the data distribution



- Minimizing average prediction error is inherently limited in presence of more complex data distributions
- Goal: represent the entire distribution (as opposed to e.g., only the mean)

Parametrizing distributions

• Use a parametric function (e.g., a neural network) to output the parameters of a distribution



+ handles multimodality- scaling to high-dim action spaces

Continuous distribution



steering angle

+ captures the continuous nature of actions- not very expressive (e.g., unimodal)

Expressive policy architectures

Gaussian Mixture Model



$$\pi(\mathbf{u}_t|\mathbf{x}_t) = \sum_{i=1}^k w_i \, \mathcal{N}(\mu_i, \Sigma_i)$$

Discretize + Autoregressive







 \mathbf{x}_t

noise

 $p_T(\mathbf{x}_T) \sim \mathcal{N}(0, l)$

Pure

noise







 \mathbf{u}_t Sample data $p(\mathbf{x}_0) \rightarrow \text{turn to noise}$ $p_0(\mathbf{x}_0)$ Clean X₀ \mathbf{X}_1 X_{T-1} XT $\mathbf{x}_t \ \mathbf{u}_{t,<i}$ sample Reverse / denoising process • Sample noise $p_T(\mathbf{x}_T) \rightarrow \text{turn into data}$ AA 203 | Lecture 15



Diffusion Policy

COLUMBIA Engineering TOYOTA RESEARCH INSTITUTE Massachusetts Institute of Technology

Diffusion Policy

Visuomotor Policy Learning via Action Diffusion







Robotics Transformer

RT-1: Robotics Transformer

Anthony Brohan Noah Brown Justice Carbajal Yevgen Chebotar Joseph Dabis Chelsea Finn Keerthana Gopalakrishnan Karol Hausman Alex Herzog Jasmine Hsu Julian Ibarz Brian Ichter Alex Irpan Tomas Jackson Sally Jesmonth Nikhil Joshi Ryan Julian Dmitry Kalashnikov Yuheng Kuang Isabel Leal Kuang-Huei Lee Utsav Malla Sergey Levine Yao Lu Deeksha Manjunath Igor Mordatch Ofir Nachum Carolina Parada Emily Perez Jodilyn Peralta Karl Pertsch Jornell Quiambao Kanishka Rao Michael Ryoo Grecia Salazar Pannag Sanketi Sumedh Sontakke Huong Tran Kevin Sayed Jaspiar Singh Austin Stone Clayton Tan Vincent Vanhoucke Quan Vuong Peng Xu Tianhe Yu Brianna Zitkovich Steve Vega Fei Xia Ted Xiao Sichun Xu

Authors listed in alphabetical order (see paper appendix for contribution statement).

Robotics at Google L Everyday Robots Google Research





5/19/2025

A useful mental model

We can address these challenges in a few different ways:

- Targeted algorithms
- Smart data collection (and augmentation)
- Use expressive models (e.g., that are able to capture the multimodality of behavior data)



Pros and Cons

Until now, we discussed IL (namely, Behavior Cloning) as the problem of learning to "mimic" the behavior of an expert from demonstrations

Pros:

- *Simplicity*: essentially, SL; easy to implement and monitor
- *Efficiency*: no need for trial-and-error learning (as in RL)
- No need for an explicit reward signal: bypasses the task of designing a reward function; only demonstrations are needed

Cons:

- Dependence on demonstration quality: poor demos lead to suboptimal policies
- Generalization and compounding errors: struggles with covariate shift
- No exploration: does not explore the space of solutions for better policies

Outline

- Imitation Learning:
 - Behavior Cloning (BC)
 - Common pitfalls
 - Design strategies for effective IL
 - Other paradigms (RvS, Inverse RL)
 - Inverse Reinforcement Learning (IRL)

Reinforcement Learning via Supervised Learning (RvS)

Recent work has explored the idea of converting the reinforcement learning problem (more in the next lecture) into a *conditional*, *filtered*, or *weighted* imitation learning problem.

Key idea: rather than relying solely on "optimal" demonstrations, leverage a broader set of demonstrations from, e.g., suboptimal policies, related tasks, etc.

Two main classes of methods:

- 1. Filtering or Weighting Demonstrations
- 2. Goal or Reward Conditioning

Filtering or Weighting Demonstrations

One common approach to RvS is to filter or weight the expert demonstrations based on their quality

Filtering: A simple approach could entail ranking the expert demonstrations based on their return: T_{-1}

$$r(au) = \sum_{t=0}^{t-1} \gamma^t R(\mathbf{x}_t, \mathbf{u}_t) \, ,$$

Then, we filter the dataset to include only the top-k% of trajectories:

$$\mathcal{D} = \{ au \in \mathcal{D} \mid r(au) \geq ar{r}\}.$$

Weighting: A different approach might involve weighting each individual transition of a trajectory based on its reward, rather than filtering entire trajectories.

$$\mathcal{D} = \{ (\mathbf{x}_n, \mathbf{u}_n) \}_{n=1}^N \quad \text{NLL}(\theta) = -\frac{1}{N} \sum_{n=1}^N \log \pi_\theta(\mathbf{u} | \mathbf{x}) R(\mathbf{x}, \mathbf{u}) \quad \hat{\theta}_{\text{mle}} = \underset{\theta}{\operatorname{argmin}} \text{NLL}(\theta)$$

5/19/2025

Goal or Reward Conditioning

Conditioning on *outcomes* is particularly useful in settings where the expert demonstrations are suboptimal or collected from a different task.

Each trajectory might be described using different outcomes, such as the final state of the trajectory, the total reward obtained, or a specific state that was visited during the trajectory.

$$ext{NLL}(heta) = -rac{1}{N}\sum_{n=1}^N \log \pi_ heta(\mathbf{u}|\mathbf{x},oldsymbol{\omega})$$

Pros:

- Easily derivable from demonstration data
- Allows to extract useful behavior from suboptimal data
- Improved generalization; decouples the process of achieving desired outcomes from the reward structure

Goal-conditioning: $\omega = \mathbf{x} \in \mathcal{X}$ Reward-conditioning: $\omega = \sum_{t=0}^{T-1} R(\mathbf{x}_t, \mathbf{u}_t)$:a \mathbf{x}_2 \mathbf{u}_2 \mathbf{u}_3 \mathbf{u}_4 \mathbf{x}_4 \mathbf{x}_1 $\pi(\mathbf{u}|\mathbf{x}, \omega = \mathbf{x}_4)$ rom

Outline

- Imitation Learning:
 - Behavior Cloning (BC)
 - Common pitfalls
 - Design strategies for effective IL
 - Other paradigms (RvS, Inverse RL)
 - Inverse Reinforcement Learning (IRL)

Inverse Reinforcement Learning (IRL)

IRL takes an orthogonal approach to IL, by attempting to recover a reward function from a policy, or from demonstrations of a policy

Example: (BC vs IRL)

A warehouse robot must navigate from a starting point to a goal while avoiding obstacles.

Behavior Cloning:

- Collect demonstration data, train a policy that imitates the demonstrations
- Effective in familiar scenarios, however, it may struggle if the layout changes substantially

Inverse RL:

- Collect the same demonstration data, infer the reward function (e.g., positive for moving closer to the goal, negative for moving close to obstacles)
- The reward function encapsulates the underlying principles of the task

Inverse Reinforcement Learning (IRL)

- Formally, the goal of IRL is to recover a reward function $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ from a dataset of demonstrations $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{u}_n)\}_{n=1}^N$
- In practice, the reward is parametrized by some parameters **w**. Therefore, the goal is to find the value of **w** that better explains the expert demonstrations
- Most IRL algorithms follow an iterative learning process:

Algorithm 22.4: High-level IRL Algorithm
Data: Expert demonstrations, D, initialized reward function parameters, w, initialized policy parameters, θ
Result: Learned reward function parameters, w, learned policy parameters, θ
while not converged do
Update the reward function parameters, w.
Update the policy parameters, θ, to maximize the current estimate of the reward function.
return Optimized reward and policy parameters: w, θ.

Key challenge: *reward ambiguity* (i.e., many reward functions explain the same data)

Popular algorithms:

- Apprenticeship learning
- Maximum Margin Planning
- Maximum entropy IRL

Recap

Skeleton of a BC algorithm:

1) Collect a dataset of "expert" demonstrations

2) Train policy $\pi_{\theta}(\mathbf{u}|\mathbf{x})$ to mimic expert:

Main Challenges:

- Compounding errors
- Multimodal behavior



$$egin{aligned} \mathcal{D} &= \{(\mathbf{x}_n, \mathbf{u}_n)\}_{n=1}^N \end{aligned}$$

Potential Solutions:

- Smart data collection
- Targeted algorithms
- Use expressive models

RvS:

Leverage broader set of demonstrations under the same "BC" paradigm

IRL: Learn the reward function from demonstrations

5/19/2025

Next class

• Reinforcement Learning

Additional References:

- Stanford CS 224R
- Berkeley CS285