CS 188 Robotics Week 6

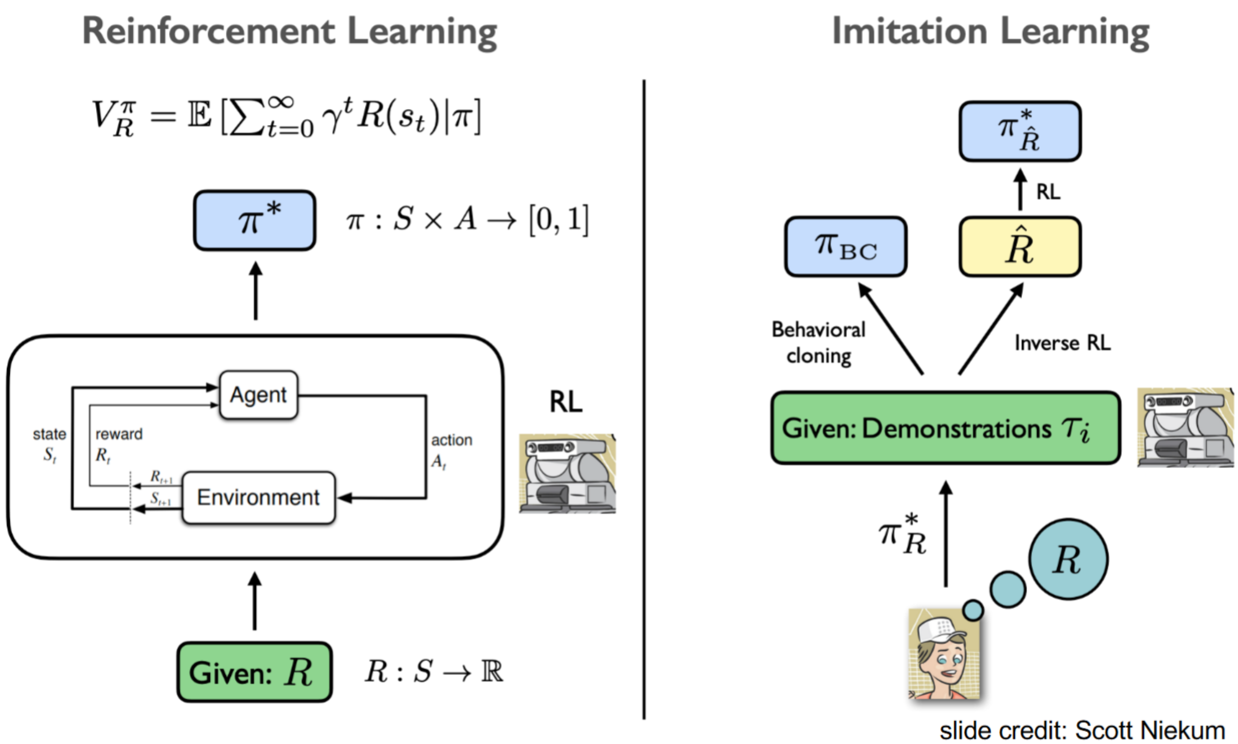
Aidan Jan

May 13, 2025

Imitation Learning

Specifying reward for RL is hard...

- **Reward hacking:** AI system learns to exploit loopholds or unintended behaviors in its reward function to achieve high rewards without actually accomplishing the intended task



Why learn from demonstrations?

- Natural and expressive
- No expert knowledge required
- Valuable human intuition
- Program new tasks as-needed

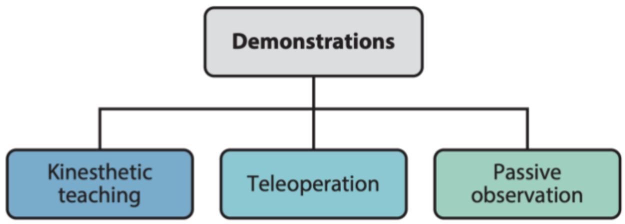
Human babies imitate adults when to learn.

How to Imitate?

Demonstrations to Autonomous Behavior

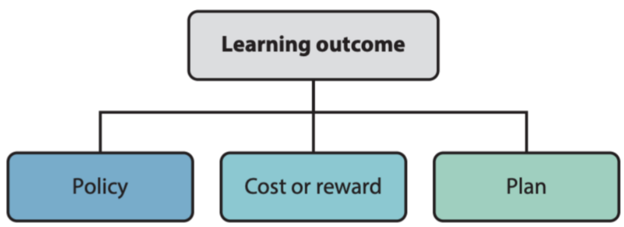
- Dynamic Movement Primitives (DMP): replay the motion
- **Behavior Cloning (BC)**: supervised learning of behavior
  - This is what everyone (as in, robotics companies) is trying to do
- Inverse Reinforcement Learning (IRL): inferring the underlying intent

Types of Demonstrations



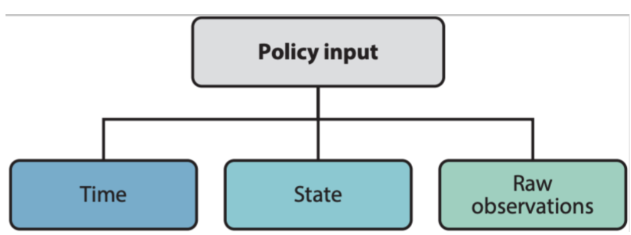
Demonstration	Ease of demonstration	High DOFs	Ease of mapping
Kinesthetic teaching	✓		✓
Teleoperation		✓	✓
Passive observation	✓	✓	

Learning Outcomes



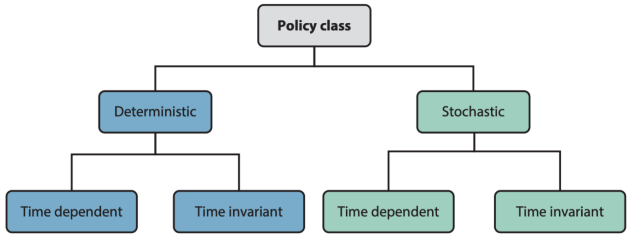
Learning outcome	Low-level control	Action space continuity	Compact representation	Long-horizon planning	Multistep tasks
Policy	✓	✓	✓		
Cost or reward	✓	✓		✓	
Plan			✓	✓	✓

Policy Parameterization



Policy input	Ease of design	Performance guarantees	Robustness to perturbations	Task Variety	Algorithmic efficiency
Time	✓	✓			✓
State		✓	✓		✓
Raw observations	✓		✓	✓	

Policy Class

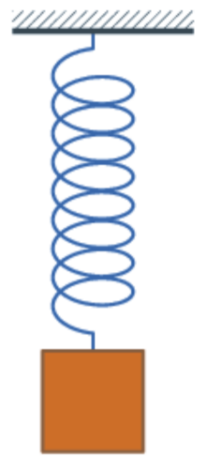


Policy class	Temporal context	Robustness to temporal perturbations	Repeatability	Multimodal behavior
Deterministic and time dependent	✓		✓	
Deterministic and time invariant		✓	✓	
Stochastic and time dependent	✓			✓
Stochastic and time invariant		✓		✓

Dynamic Movement Primitives (DMP)

$$\tau\dot{v} = K(\overset{\text{goal}}{\underline{g}} - x) - Dv + (\overset{\text{goal}}{\underline{g}} - \overset{\text{initial state}}{\underline{x_0}})\, f$$

$$\tau\dot{x} = v$$



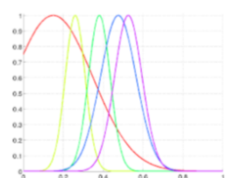
K: spring constant

D: damping term

Non-linear force function:

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)} \qquad \psi_i(s) = \exp(-h_i(s-c_i)^2)$$

Gaussian basis functions



canonical system: $\tau\dot{s} = -\alpha\, s$

s: phase variable

Learning:

$$f_{\text{target}}(s) = \frac{-K(g-x) + Dv + \tau\dot{v}}{g - x_0} \qquad f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)}$$

$$J = \sum_s \left(f_{\text{target}}(s) - f(s)\right)^2$$

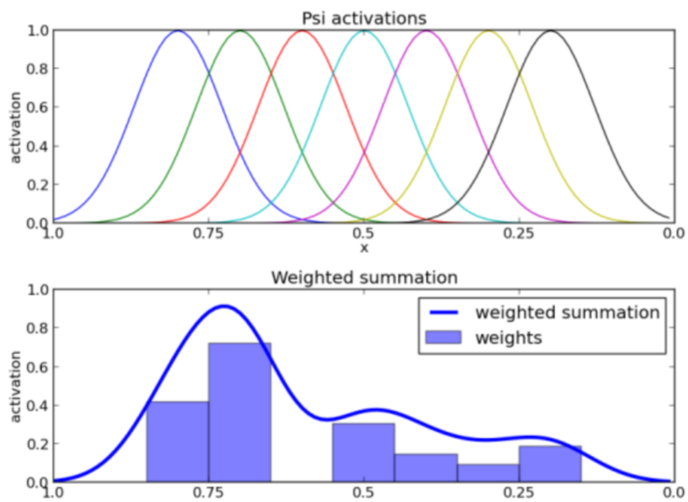
Linear regression

Pastor, Peter, et al. "Learning and generalization of motor skills by learning from demonstration." *2009 IEEE international conference on robotics and automation*. IEEE, 2009.

Characteristics of DMPs

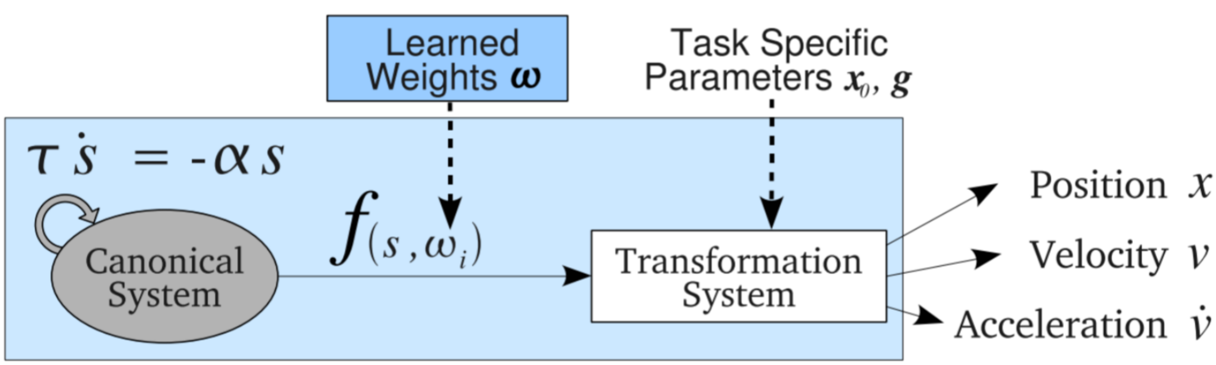
- Convergence to the goal g is guaranteed (for bounded weights) since $f(s)$ vanishes at the end of a movement
- The weights w_i can be learned to generate any desired *smooth* trajectory.
- The equations are spatial and temporal invariant, i.e., movements are self-similar for a change in goal, start point, and temporal scaling without a need to change the weights w_i
- The formulation generates movements which are robust against perturbation due to the inherent attractor dynamics of the equations.

Weighted Sum of Gaussian Basis



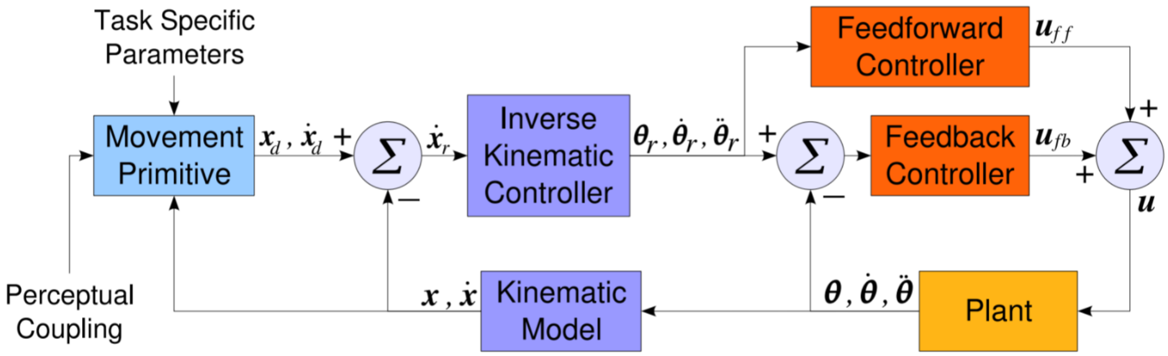
https://studywolf.wordpress.com/2013/11/16/dynamic-movement-primitives-part-1-the-basics/

Dynamic Movement Primitives



Sketch of a one dimensional DMP: the canonical system drives the nonlinear function f which perturbs the transformation system.

Pastor, Peter, et al. "Learning and generalization of motor skills by learning from demonstration." *2009 IEEE international conference on robotics and automation*. IEEE, 2009.

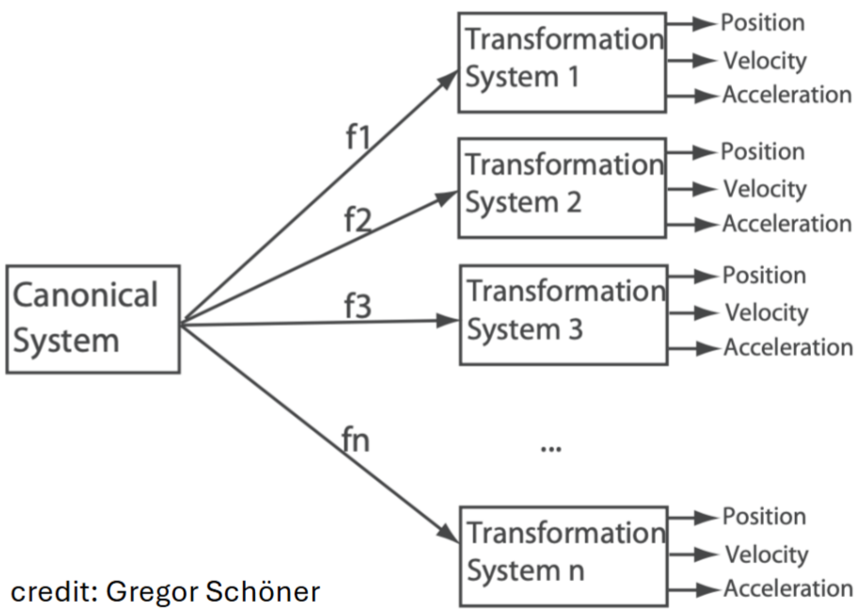


DMP control diagram: the desired task space positions and velocities are xd, x ˙ d, the reference task space velocity commands are x ˙ r , the reference joint positions, joint velocities, and joint accelerations are θr, θ ˙ r, and θ ¨ r.

Pastor, Peter, et al. "Learning and generalization of motor skills by learning from demonstration." *2009 IEEE international conference on robotics and automation*. IEEE, 2009.

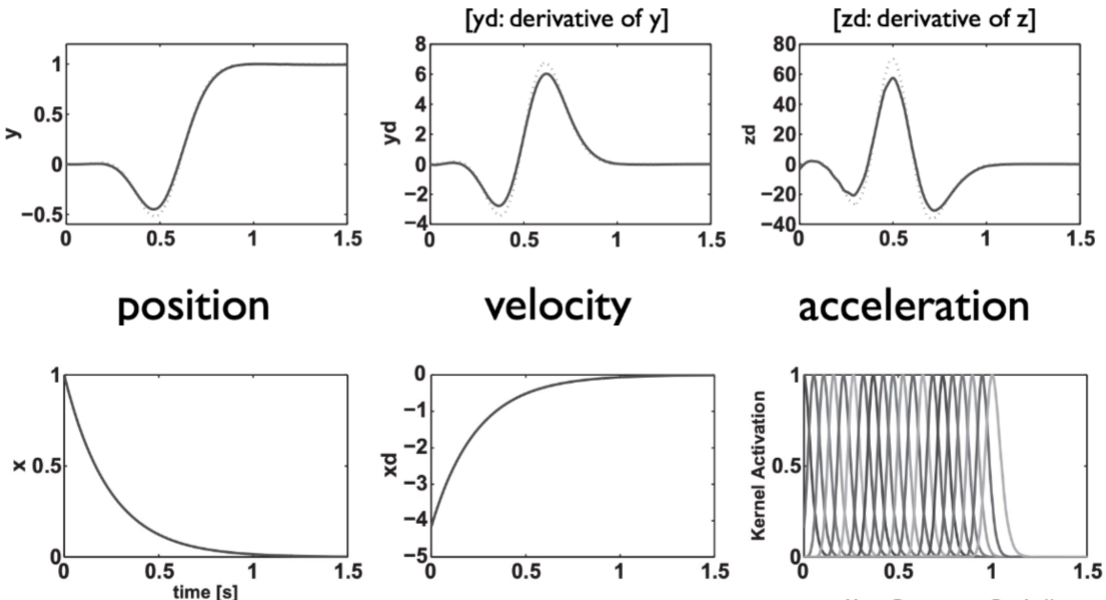
Multidimensional

- one central harmonic oscillator
- multiple transformations



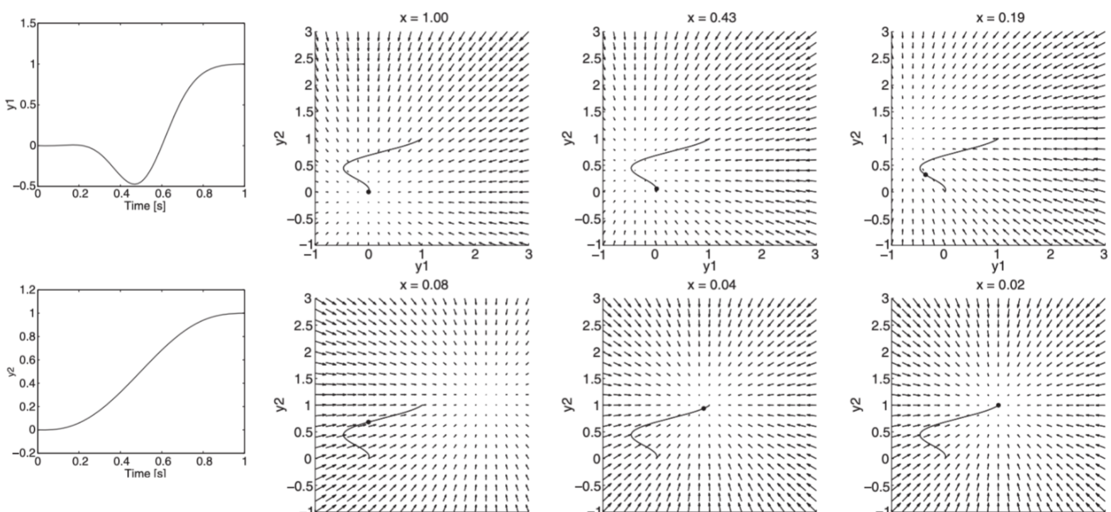
Examples:

- 1 Dimensional:



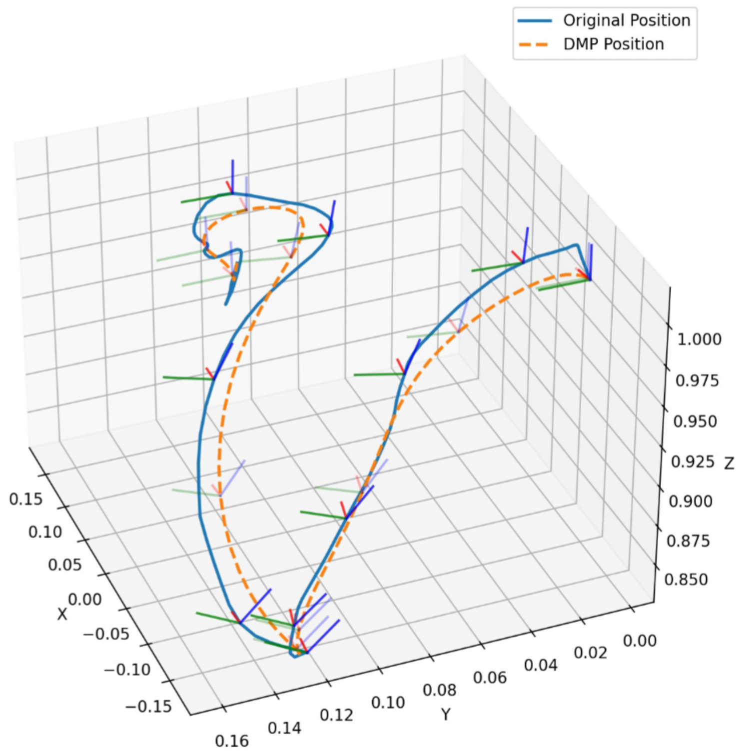
credit: Gregor Schöner

- 2 Dimensional:

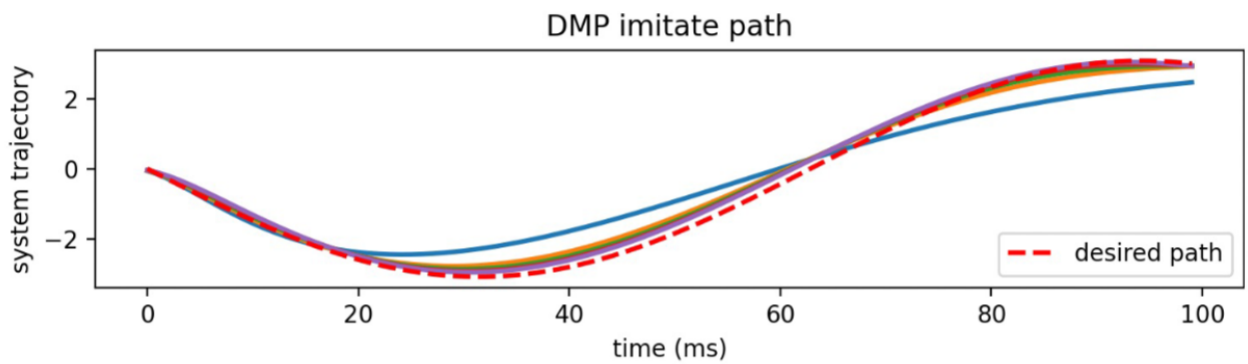


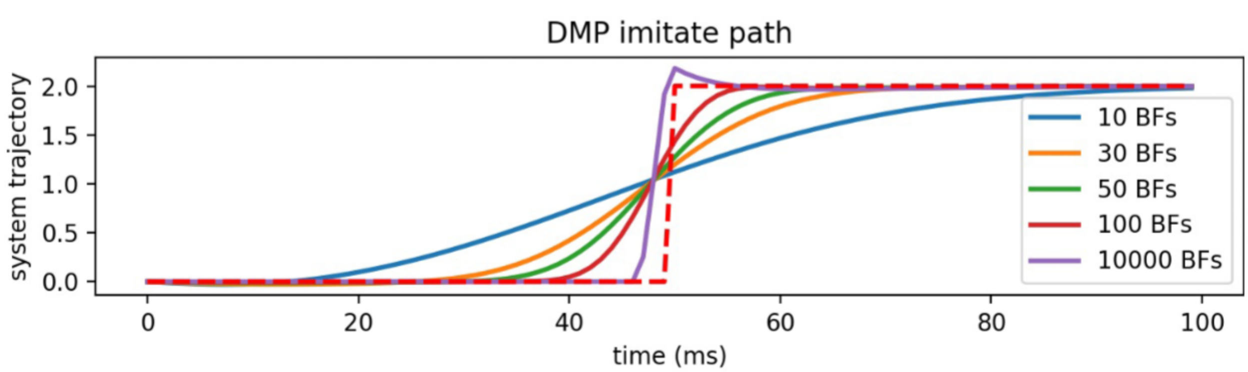
credit: Gregor Schöner

- 3 Dimensional / 6 Dimensional:



Limitations of DMPs





Summary

- DMP enable learning "movement styles" while enabling generalization to new movement targets
- DMP is a purely kinematic account ⇒ DMP is not addressing control in that respect, analogy to force-fields is misleading
- DMP addresses timing, but account of coordination is limited
- DMP for different tasks and their combination. . . ?

IID (Independent and Identically Distributed)

Imitation Learning

[FILL 7]

Supervised Learning

[FILL 9]

The i.i.d. Assumption

- "The training and testing data are **independent** and **identically** distributed."

[FILL 11] [FILL 12, incl red text]

- **Robustness** refers to the ability of a system, model, or method to maintain performance or produce reliable results **despite variations, noise, errors, or adversarial conditions** in the input or environment.

Input Data Distribution

- End-to-End Control Tasks [FILL 13]
- Markov Decision Process <S, A, P, R> [FILL 14, 15]

Behavioral Cloning

[FILL 16] Downside: The robot learns similarly to what you tell it to do, but may not be able to apply the situation. For example, if you train a robot to turn right, it may turn right spontaneously while driving on a very long, straight road.

Challenges in Imitation Learning

[FILL 18]

Quadratic Regret

- **Regret** (in decision theory) measures the difference between the reward (or outcome) one actually received and the best possible reward one *could have received* if one had made the optimal choice

$$\hat{\pi}_{sup} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{s^*}} [l(s, \pi)]$$

- Assuming $l(s, \pi)$ is the 0-1 loss (or upper bound on the 0-1 loss) implies the following performance guarantee with respect to any task cost function C bounded in [0, 1]:

- **Theorem 2.1** (Ross and Bagnell, 2010), *Let* $\mathbb{E}_{s\sim d_{s^*}[l(s,\pi)]=\epsilon}$, *then* $J(\pi) \leq K(\pi^*) + T^2\epsilon$.
- Compare to typical supervised learning loss that grows as: $O(\epsilon T)$

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell. **"A reduction of imitation learning and structured prediction to no-regret online learning."** *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 2011.

DAgger: Dataset Aggregation

End-to-End Control Tasks [FILL 21, im10]

`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` $\mathcal{D}$

`return best` $\hat{\pi}_i$ `on validation`

Key idea: keep collecting demonstration data that is on-distribution for current policy, and reduce dependence on expert over time

- **Theorem 2.2** *Let* π *be such that* $\mathbb{E}_{s\sim d_s}[l(s,\pi)] = \epsilon$, *and* $Q^{\pi^*}_{T-t+1}(s,a) - Q^{\pi^*}_{T-t+1}(s,\pi^*) \leq u$ *for all action* $a, t \in \{1, 2, \ldots, T\}, d^t_\pi(s) > 0$, *then* $J(\pi) \leq J(\pi^*) + uT\epsilon$
- If difference between optimal t-step Q and any other action is u (e.g., the worst single action regret): The end cost is (no) worse than optimal plus number of mistakes times u, the worst possible regret of each mistake.

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell. **"A reduction of imitation learning and structured prediction to no-regret online learning."** *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 2011.

What are some other ways to improve robustness?

$$J(\pi) \leq J(\pi^*) + T^2\epsilon$$

- Modify the T^2!
- Increasing T is basically increasing the number of actions.

[FILL 24] In this image, the yellow line represents the trainer's performance. The orange line represents the robot cloning the behavior. [FILL 26, 25, remove red/green text, combine]

- Unintended low-level motions constitute noise in demonstrations
- The demonstrator's **high-level actions** are optimal!
- These actions can be categorized into **two general modes.**

HYDRA

[FILL 27, im13, 28, 30]

Challenges in Imitiation Learning #2

[FILL 31]

Generative Modeling

[FILL 32, 33]

Diffusion

[FILL 34, 35, im20, 36, 37, im22]

How to learn from human data?

- Behavioral Cloning
  - Quadratic regret in worst case; bad performance out of expert distribution
  - Can't learn from additional data collected by the agent
- DAgger
  - Provides data/feedback on-policy
  - still can't learn from additional data collected by the agent
- Inverse reinforcement learning
  - Infers reward function from demonstrations so that RL can be used

Inverse Reinforcement Learning

[FILL 40, full]

1. Collect user demonstration $(s_0, a_0), (s_1, a_1), \ldots, (s_n, a_n)$ and assume it is sampled from the expert's policy, π^E
2. Explain expert demos by finding R^* such that:

$$E[\sum_{t=0^\infty} \gamma^t R^*(s_t)|\pi^E] \geq E[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi]\forall\pi$$
$$E_{s_0 \sim D}[V^{\pi^E}(s_0)] \geq E_{s_0 \sim D}[V^{\pi}(s_0)]\forall\pi$$

How can search be made tractable?

Linear reward functions

Define R^* as a linear combination of features:

$$R^*(s) = w^T \phi(s), \text{ where } \phi : S \to \mathbb{R}^n$$

Then,

$$\begin{aligned} E[\sum_{t=0}^{\infty}\gamma^t R^*(s_t)|\pi] &= E[\sum_{t=0}^{\infty}\gamma^t w^T\phi(s_t)|\pi] \\ &= w^T E[\sum_{t=0}^{\infty}\gamma^t \phi(s_t)|\pi] \\ &= w^T\mu(\pi) \end{aligned}$$

Thus, the expected value of a policy can be expressed as a weighted sum of the expected features $\mu(\pi)$

A simplified optimization problem

Originally, Explain expert demos by finding R^* such that:

$$E[\sum_{t=0}^{\infty}\gamma^t R^*(s_t)|\pi^E] \geq E[\sum_{t=0}^{\infty}\gamma^t R^*(s_t)|\pi]\forall\pi$$

Use expected features:

$$E[\sum_{t=0}^{\infty}\gamma^t R^*(s_t)|\pi] = w^T\mu(\pi)$$

Restated - find w^* such that:

$$w^*\mu(\pi^E) \geq w^*\mu(\pi)\forall\pi$$

Iterative Reward Search

Goal: Find w^* such that $w^*\mu(\pi^E) \geq w^*\mu(\pi)\forall\pi$

- Initialize π_0 to any policy.
- Iterate for $i = 1, 2, \ldots$:
  - Find w^* such that expert maximally outperforms all previously examined policies $\pi_{0,\ldots,i-1}$

    $$\max_{\epsilon, w^* : \|w^*\|_2 \leq 1} \epsilon \text{ s.t. } w^*\mu(\pi^E) \geq w^*\mu(\pi_j) + \epsilon$$

    * The above is a support vector machine (SVM) solver
  - Use RL to calculate optimal policy π_i associated with w^*
  - Stop if $\epsilon \leq$ threshold

A (rough) illustration

[FILL 45-47, combined, im24]

Naive IRL Challenges

- RL in the inner loop
- Where do linear features come from?
- Underspecified inference problem: infinite reward functions that explain behavior equally well. Which one to choose?
- Policies are underspecified too: many policies lead to the same expected features counts. Which one to choose?
- What if demonstrated behavior was actually suboptimal?

Suboptimality and Policy Mixtures

- If a demonstrator acts optimally, then there trivially is some reward function for which **at least one** optimal policy exists that matches the demonstrator's expected feature counts exactly
- But if the demonstrations are sometimes suboptimal, then here may be no single reward function with this property (aside from degenerate ones e.g. all zeros, under which everything is optimal)
- This can be thought of as the demonstrator sometimes acting optimally under a different reward function, so a mixture of reward functions (and their corresponding optimal policies) would be needed to match the feature counts
- Instead, we will now consider policies that are not strictly optimal, but produce trajectories in proportion to their return:
$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\zeta_i}}$$

Principle of Maximum Entropy

- **Definition**: the probability distribution which best represents the current state of knowledge about a system is the one with largest entropy, subject to your constraints.
- **Intuitively**: Don't overcommit in ways that aren't supported by the data — e.g. don't prefer one trajectory over another if they have the same return.
- **Practical consequence for IRL**: Tells us how to tiebreak between reward functions that explain the data equally well.
- **How?**: Find a reward function that matches expert feature counts under a specific trajectory distribution:
$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\zeta_i}}$$
- **Why this distribution?**: If you have specific expected feature counts f that you wish to match, it is known that the maximum entropy trajectory distribution that matches f is of the above form for some θ

[FILL 51]

Trajectory vs. Action-based Reasoning

[FILL 52, full, incl text]

Trajectory Probabilities

- Deterministic: $P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\zeta_i}}$
- Stochastic: $P(\zeta|\theta, T) \approx \frac{e^{\theta^T f_\zeta}}{Z(\theta, T)} \prod_{s_{t+1}, a_t, s_t \in \zeta} P_t(s_{t+1}|a_t, s_t)$

Learning a Reward Function

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_{\text{examples}} \log P(\bar{\zeta}|\theta, T)$$

$$\nabla L(\theta) = \bar{f} - \sum_\zeta P(\zeta|\theta, T) f_\zeta = \bar{f} - \sum_{s_i} D_{s_i} f_{s_i}$$

- How do we compute the D_{s_i}?

Calculating state visitation frequencies

[FILL 55]

Driver Route Modeling

[FILL 57]

- Collected driving data of 100,000 miles spanning 3,000 driving hours for Pittsburgh
- Fitted GPS data to the road network, to generate $\sim 13,000$ road trips
- Discarded noisy trips, or trips that were too short (less than 10 road segments)

Four different road aspects considered:

- Road type: interstate to local road
- Speed: high speed to low speed
- Lanes: multi-lane or single lane
- Transitions: straight, left, right, hard left, hard right

There was a total of 22 features used to represent this state.

Model	% Matching	% >90% Match	Log Prob	Reference
Time-Based	72.38	43.12	N/A	N/A
Max Margin	75.29	46.56	N/A	[Ratliff, Bagnell, and Zinkevich, 2006]
Action	77.30	50.37	-7.91	[Ramchandran and Amir 2007]
Action (Cost)	77.74	50.75	N/A	[Ramchandran and Amir 2007]
MaxEnt	**78.79**	**52.98**	**-6.85**	**[Zeibart et al. 2008]**

What about Larger Problems?

- MaxEnt IRL: probabilistic framework for learning reward functions:
  - Computing gradient requires enumerating state-action visitations for all states and actions
  - Only really viable for small, discrete state and action spaces
  - Amounts to a dynamic programming algorithm (exact forward-backward inference)
- For deep IRL, we want two things:
  - Large and continuous state and action spaces
  - Effectinve learning under unknown dynamics

Guided Cost Learning

[FILL 60, im29]