Lecture 7: Imitation Learning[2]

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2018

[2]With slides from Katerina Fragkiadaki and Pieter Abbeel

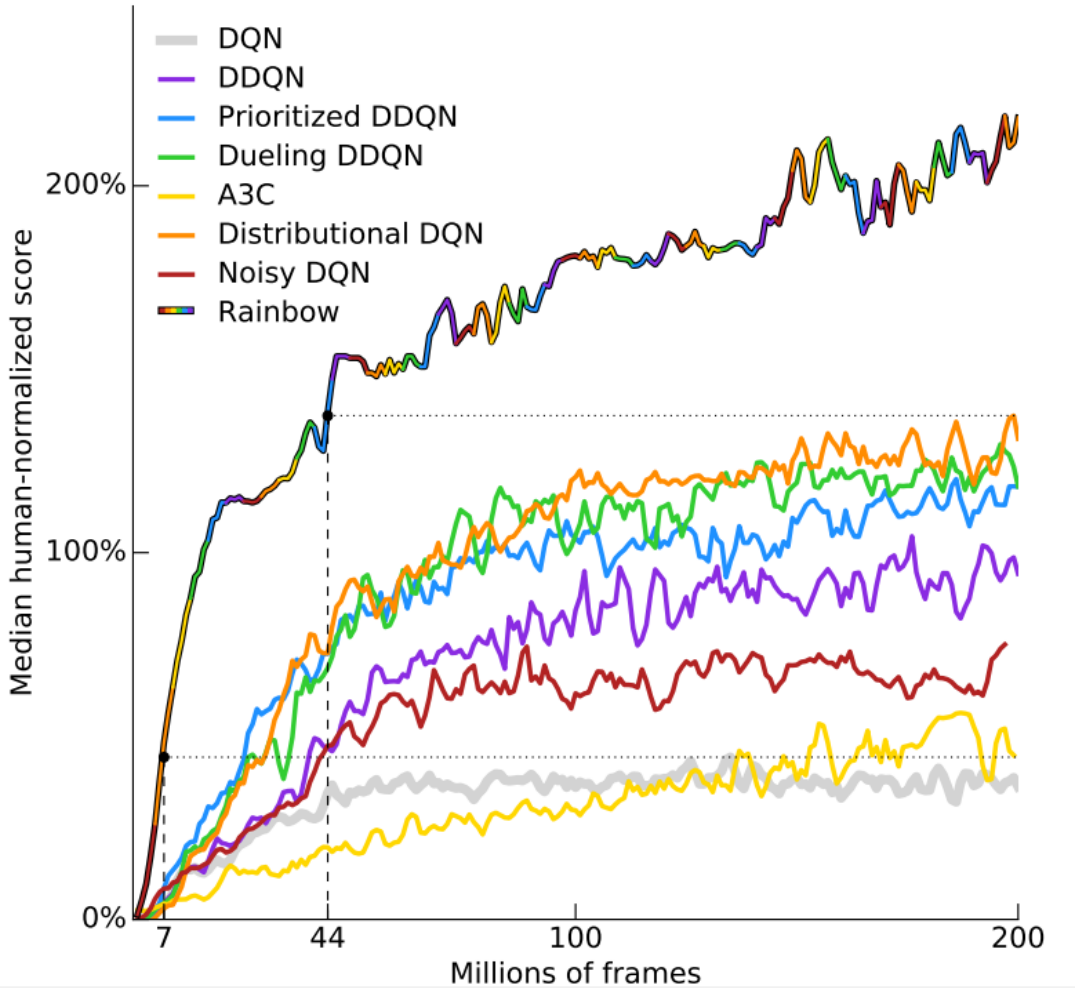
Table of Contents

1. Behavioral Cloning
2. Inverse Reinforcement Learning
3. Apprenticeship Learning
4. Max Entropy Inverse RL

Recall: Reinforcement Learning Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization

Deep Reinforcement Learning



- Hessel, Matteo, et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning."

We want RL Algorithms that Perform

- Optimization
- Delayed consequences
- Exploration
- Generalization
- And do it all statistically and computationally efficiently

Data efficiency

Generalization and Efficiency

- We will discuss efficient exploration in more depth later in the class
- But exist hardness results that if learning in a generic MDP, can require large number of samples to learn a good policy
- This number is generally infeasible
- Alternate idea: use structure and additional knowledge to help constrain and speed reinforcement learning
- Today: Imitation learning
- Later:
  - Policy search (can encode domain knowledge in the form of the policy class used)
  - Strategic exploration
  - Incorporating human help (in the form of teaching, reward specification, action specification, . . .)

Class Structure

- Last time: CNNs and Deep Reinforcement learning
- **This time: Imitation Learning**
- Next time: Policy Search

TD
MC

sampling s'

$$BV = \max_a r(s,a) + \gamma \underbrace{\sum_{s'} p(s'|s,a)}_{\text{full exp.}} V^*(s')$$

Consider Montezuma's revenge

- std DQN

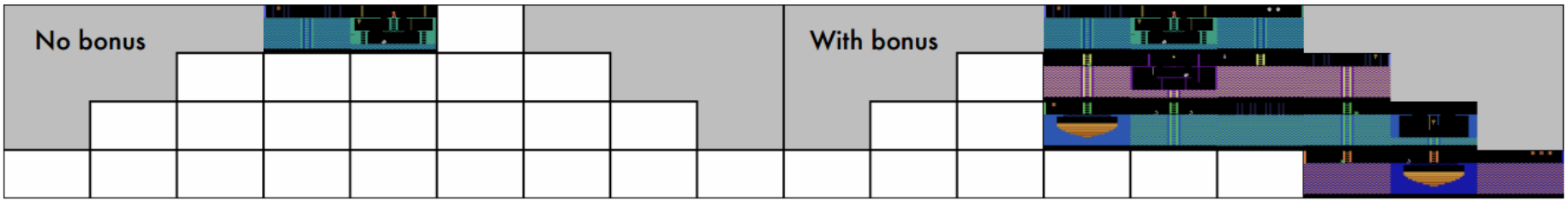


Figure 3: "Known world" of a DQN agent trained for 50 million frames with (**right**) and without (**left**) count-based exploration bonuses, in MONTEZUMA'S REVENGE.

- Bellemare et al. "Unifying Count-Based Exploration and Intrinsic Motivation"
- Vs: https://www.youtube.com/watch?v=JR6wmLaYuu4

So Far in this Course

Reinforcement Learning: Learning policies guided by (often sparse) rewards (e.g. win the game or not)

- Good: simple, cheap form of supervision
- Bad: High sample complexity

Where is it successful?

- In simulation where data is cheap and parallelization is easy
- Not when:
  - Execution of actions is slow
  - Very expensive or not tolerable to fail
  - Want to be safe

helicopter
robotics
healthcare

Reward Shaping

Rewards that are **dense in time** closely guide the agent
How can we supply these rewards?

- **Manually design them**: often brittle
- **Implicitly specify them through demonstrations**

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

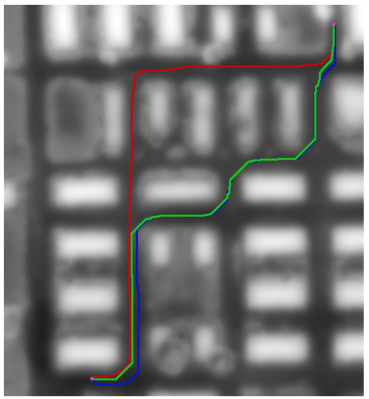
Examples

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



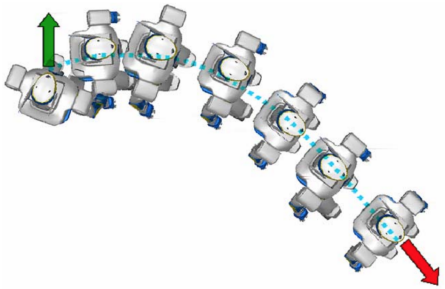
Parking lot navigation

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

Examples

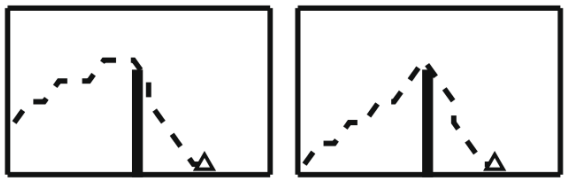
Human path planning

- 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

Problem Setup

- Input:
  - State space, action space
  - Transition model $P(s' \mid s, a)$
  - No reward function R
  - Set of one or more teacher's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (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?

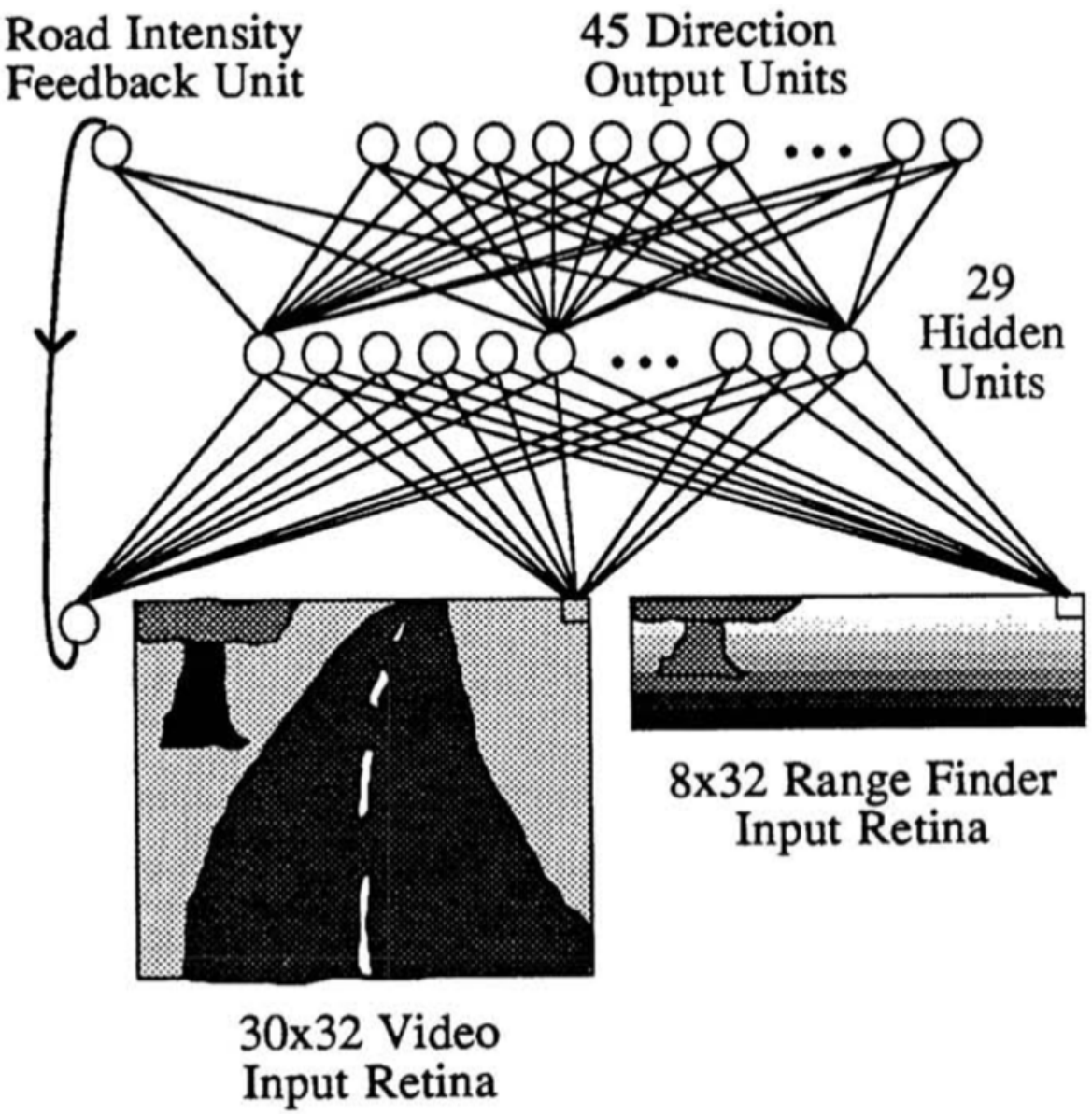
Table of Contents

1. Behavioral Cloning
2. Inverse Reinforcement Learning
3. Apprenticeship Learning
4. Max Entropy Inverse RL

Behavioral 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 $(s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots$
- Two notable success stories:
  - Pomerleau, NIPS 1989: ALVINN
  - Summut et al., ICML 1992: Learning to fly in flight simulator

ALVINN

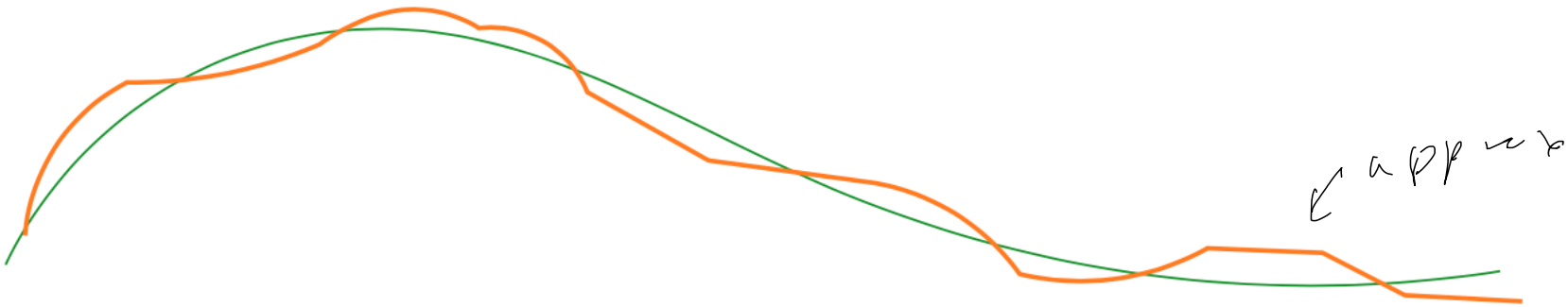

Road Intensity
Feedback Unit
45 Direction
Output Units
29
Hidden
Units
8x32 Range Finder
Input Retina
30x32 Video
Input Retina

Problem: Compounding Errors

supervised learning assum i.i.d

state → action

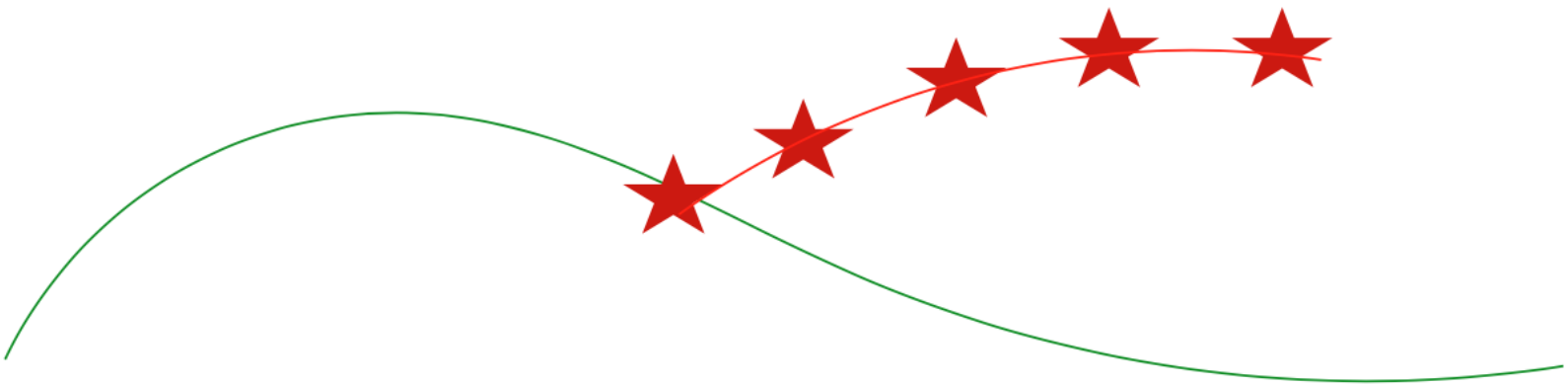
Independent in time errors:



Error at time t with probability ϵ

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

Problem: Compounding Errors

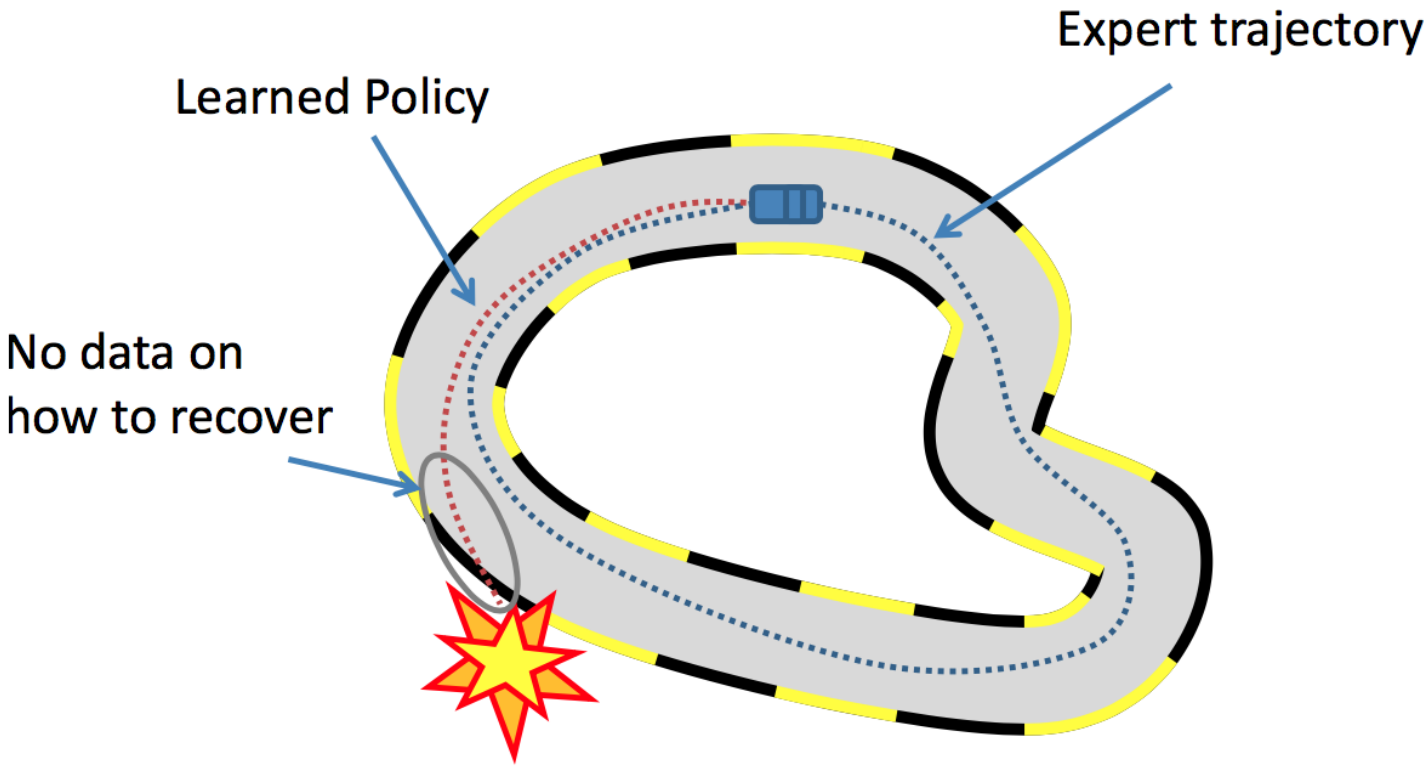


Error at time t with probability ϵ

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

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

Problem: Compounding Errors



Data distribution mismatch!

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

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

DAGGER: Dataset Aggregation

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: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

expert

- Idea: Get more labels of the right action along the path taken by the policy computed by behavior cloning
- Obtains a stationary deterministic policy with good performance under its induced state distribution

Table of Contents

1. Behavioral Cloning
2. Inverse Reinforcement Learning
3. Apprenticeship Learning
4. Max Entropy Inverse RL

Feature Based Reward Function

- Given state space, action space, transition model $P(s' \mid s, a)$
- No reward function R
- Set of one or more teacher's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from teacher's policy π)
- Goal: infer the reward function R
- With no assumptions on the optimality of the teacher's policy, what can be inferred about R?

- Now assume that the teacher's policy is optimal. What can be inferred about R?

∞ set $\quad R(s,a) = 0 \quad \forall s, a$

Linear Feature Reward Inverse RL

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^{\pi} == \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)] \tag{1}$$

Linear Feature Reward Inverse RL

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi == \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi] \quad = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{w}^T x(s_t) \mid \pi] \tag{2}$$

$$= w^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t x(s_t) \mid \pi] \tag{3}$$

$$= \boldsymbol{w}^T \mu(\pi) \tag{4}$$

- where $\mu(\pi)(s)$ is defined as the discounted weighted frequency of state s under policy π.

Table of Contents

1. Behavioral Cloning
2. Inverse Reinforcement Learning
3. Apprenticeship Learning
4. Max Entropy Inverse RL

Linear Feature Reward Inverse RL

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi = \boldsymbol{w}^T \mu(\pi) \tag{5}$$

- where $\mu(\pi)(s)$ is defined as the discounted weighted frequency of state s under policy π.
- Note that $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] \quad \forall \pi,$
- Therefore if the expert's demonstrations are from the optimal policy, to identify $\boldsymbol{w}$ it is sufficient to find w^* such that

$$w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi), \forall \pi \neq \pi^* \tag{6}$$

Feature Matching

- Want to find a reward function such that the expert policy outperforms other policies.
- For a policy π to be guaranteed to perform as well as the expert policy π^*, it suffices that we have a policy such that its discounted summed feature expectations match the expert's policy[30].
- more precisely, if

$$\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon \tag{7}$$

then for all w with $\|w\|_\infty \leq 1$:

$$|w^{*T}\mu(\pi) - w^{*T}\mu(\pi^*)| \leq \epsilon$$

[30] Abbeel and Ng, 2004

Apprenticeship Learning

- This observation leads to the following algorithm for learning a policy that is as good as the expert policy
- Assumption: $R(s) = w^T x(s)$
- Initialize policy π_0
- For $i = 1, 2 \ldots$
  - Find a reward function such that the teacher maximally outperforms all previous controllers:

$$\arg\max_{\boldsymbol{w}} \max_{\gamma} s.t. w^T \mu(\pi^*) \geq w^T \mu(\pi) + \underbrace{\gamma}_{\text{slack}} \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\} \tag{8}$$

  - s.t. $\|w\|_2 \leq 1$
  - Find optimal control policy π_i for the current $\boldsymbol{w}$
  - Exit if $\gamma \leq \epsilon/2$

involves computing π

Feature Expectation Matching

- If expert policy is suboptimal then the resulting policy is a mixture of somewhat arbitrary policies which have expert in the convex hull
- In practice: pick the best one of this set and pick the corresponding reward function.

Ambiguity

- There is an infinite number of reward functions with the same optimal policy.
- There are infinitely many stochastic policies that can match feature counts
- Which one should be chosen?

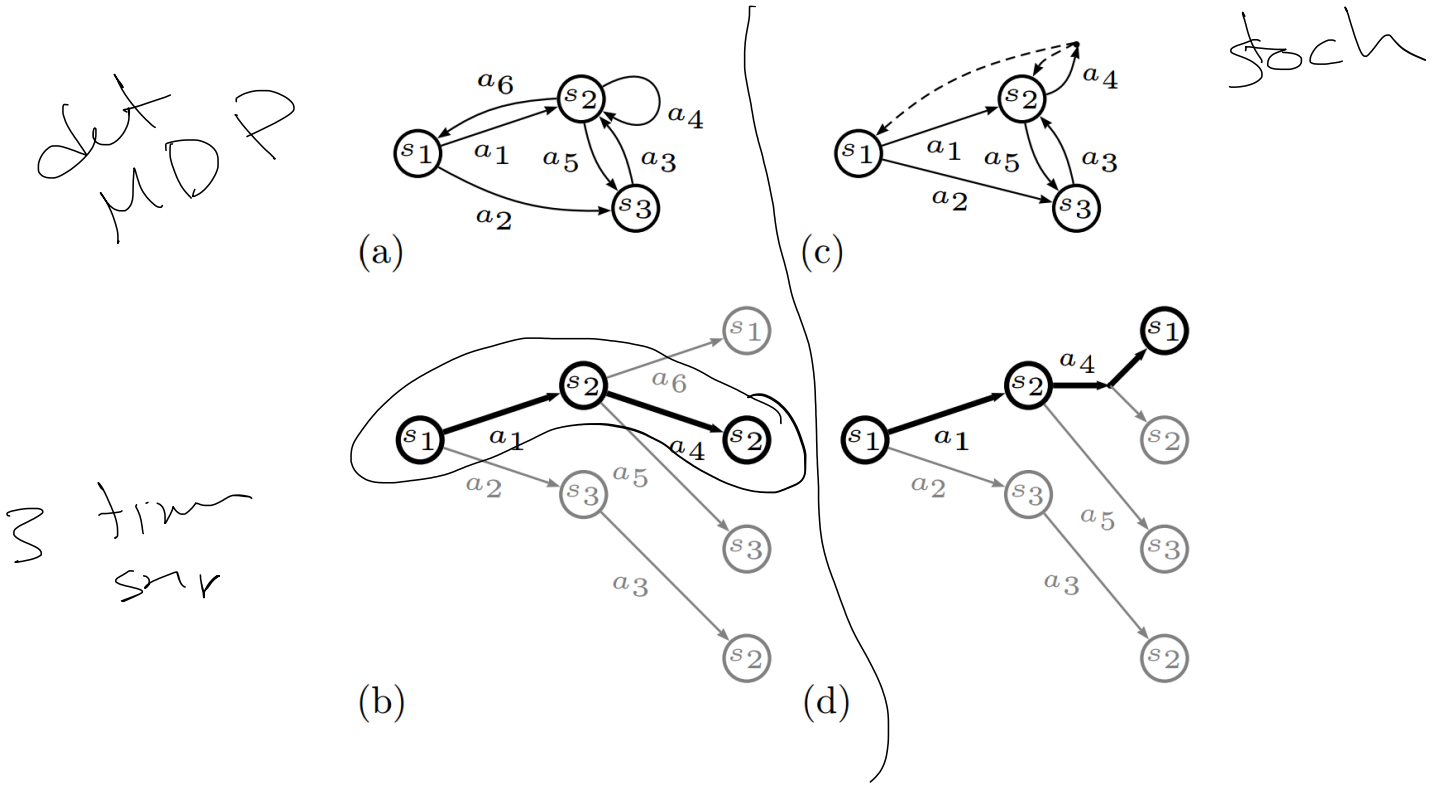
Table of Contents

1. Behavioral Cloning
2. Inverse Reinforcement Learning
3. Apprenticeship Learning
4. Max Entropy Inverse RL

Max Entropy Inverse RL

- Again assume a linear reward function $R(s) = \boldsymbol{w}^T x(s)$
- Define the total feature counts for a single trajectory τ_j as: $\mu_{\tau_j} = \sum_{s_i \in \tau_j} x(s_i)$
  - Note that this is a slightly different definition that we saw earlier
- The average feature counts over m trajectories is: $\tilde{\mu} = \frac{1}{m} \sum_{j=1}^{m} \mu_{\tau_j}$

Deterministic MDP Path Distributions



- Consider all possible H-step trajectories in a deterministic MDP
- For a linear reward model, a policy is completely specified by its distribution over trajectories
- Which policy/distribution should we choose given a set of m demonstrations?

Principle of Max Entropy

- Principle of max entropy: choose distribution with no additional preferences beyond matching the feature expectations in the demonstration dataset

$$\max_P -\sum_\tau P(\tau)\log P(\tau) s.t. \quad \sum_\tau P(\tau)\mu_\tau = \tilde{\mu} \qquad \sum_\tau P(\tau) = 1 \tag{9}$$

- In the linear reward case, this is equivalent to specifying the weights $\boldsymbol{w}$ that yield a policy with the max entropy constrained to matching the feature expectations

Ziebart et al., 2008

Max Entropy Principle

- Maximizing the entropy of the distribution over the paths subject to the feature constraints from observed data implies we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution[39].

$$P(\tau_j \mid w) = \frac{1}{Z(w)} \exp\left(w^T \mu_{\tau_j}\right) = \frac{1}{Z(w)} \exp\left(\sum_{s_i \in \tau_j} w^T x(s_i)\right)$$

$$Z(w, s) = \sum_{\tau_s} \exp\left(w^T \mu_{\tau_s}\right)$$

- Strong preference for low cost paths, equal cost paths are equally probable.

[39] Jaynes 1957

Stochastic MDPs

- Many MDPs of interest are stochastic
- For these the distribution over paths depends both on the reward weights and on the stochastic dynamics

$$P(\tau_j \mid w, P(s'|s,a)) \approx \frac{\exp\left(w^T \mu_{\tau_j}\right)}{Z(w, P(s'|s,a))} \prod_{s_i, a_i \in \tau_j} P(s_{i+1}|s_i, a_i)$$

Learning w

- Select $\boldsymbol{w}$ to maximize likelihood of data:

$$w^* = \arg\max_w L(w) = \arg\max_w \sum_{\text{examples}} \log P(\tau \mid w)$$

- The gradient is the difference between expected empirical feature counts and the learner's expected feature counts, which can be expressed in terms of expected state visitation frequencies

$$\nabla L(w) = \tilde{\mu} - \sum_{\tau} P(\tau \mid w)\mu_\tau = \tilde{\mu} - \sum_{s_i} D(s_i)x(s_i)$$

$D(s_i)$: state visitation frequency

Do we need to know the transition model to compute the above? yes

MaxEnt IRL Algorithm

Backward pass

1. Set $Z_{s_i,0} = 1$
2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5. Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_{i,t}}$

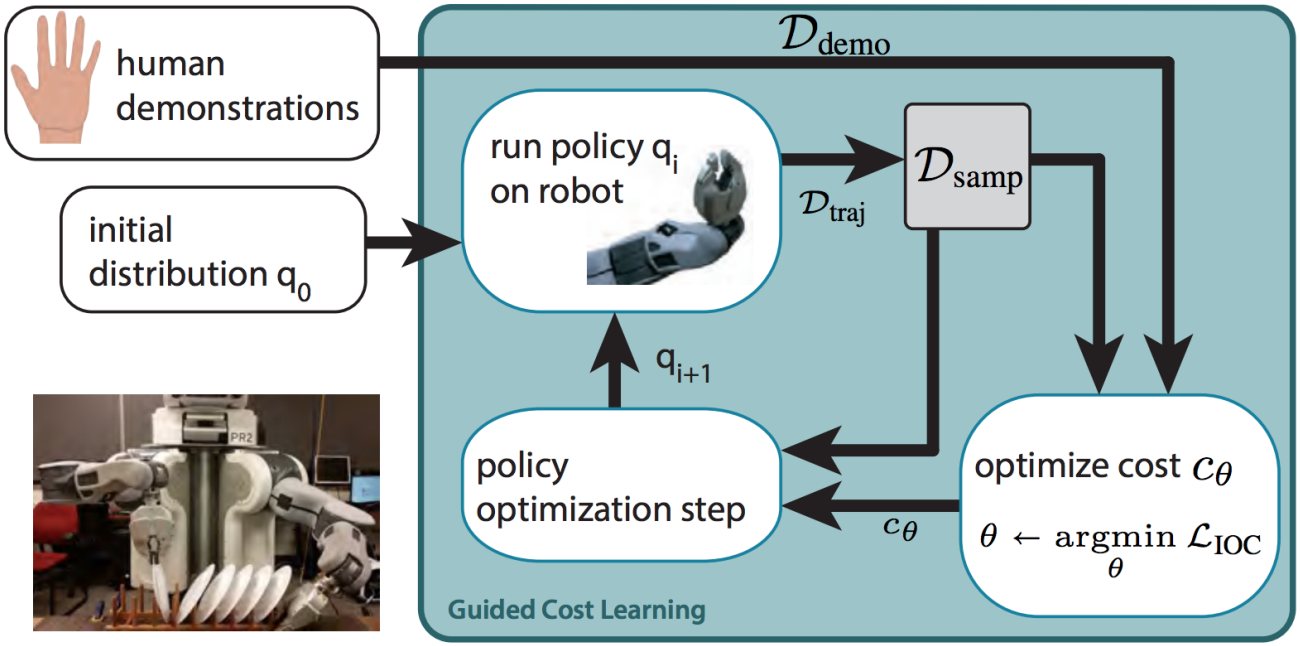
Max Entropy IRL

- Max entropy approach has been hugely influential
- Provides a principled way for selecting among the (many) possible reward functions
- The original formulation requires knowledge of the transition model or the ability to simulate/act in the world to gather samples of the transition model
  - Check your understanding: was this needed in behavioral cloning?

From IRL to Policies

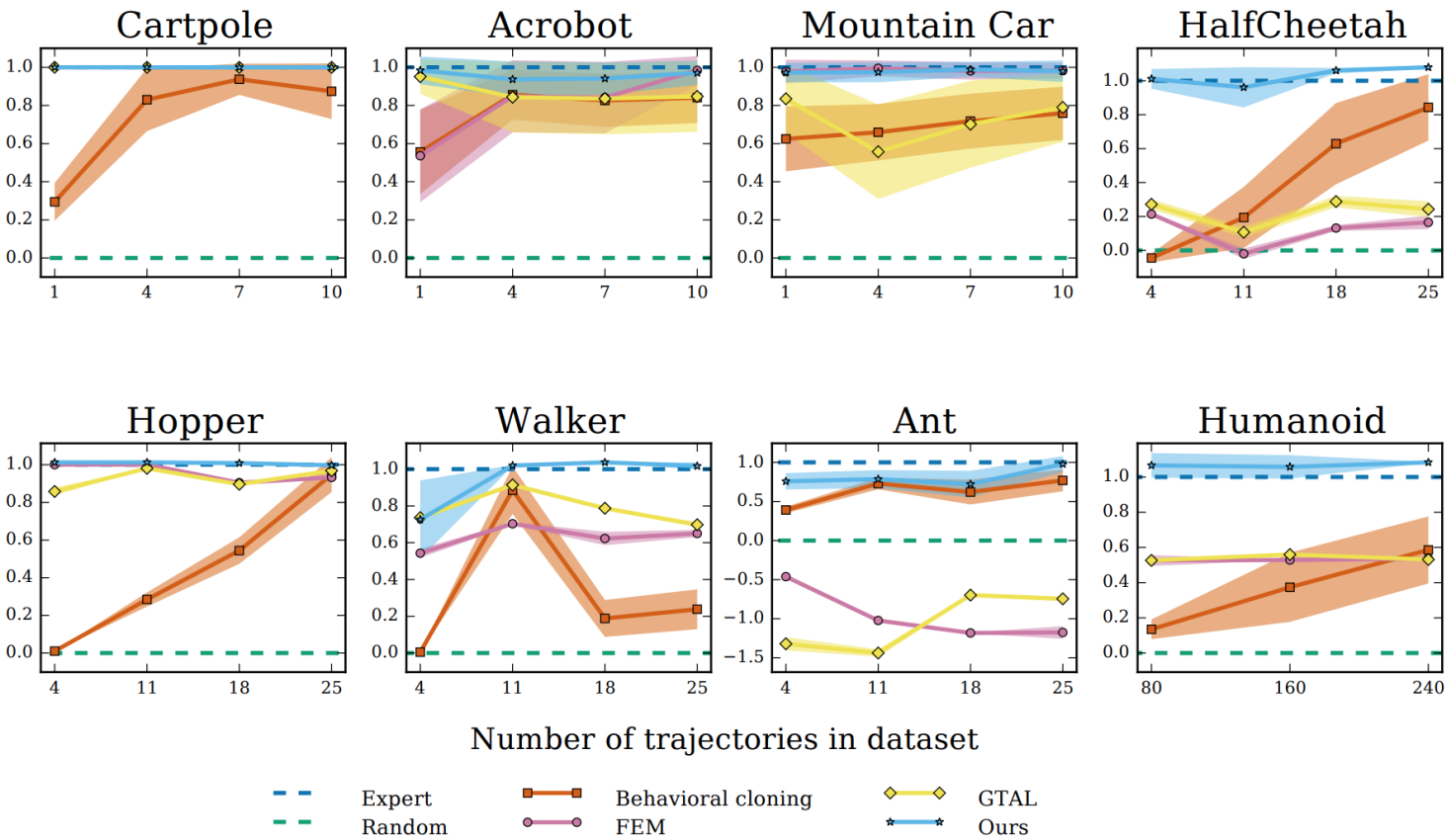
- Inverse RL approaches provide a way to learn a reward function
- Generally interested in using this reward function to compute a policy whose performance equals or exceeds the expert policy
- One approach: given learned reward function, use with regular RL
- Can we more directly learn the desired policy?

Guided Cost Learning



Finn et al., 2016

Generative Adversarial Imitation Learning



Formulate Imitation Learning as Generative Adversarial Network, use TRPO to provide demonstrations that can be compared with expert.

Ho and Ermon, 2016

Table of Contents

1. Behavioral Cloning
2. Inverse Reinforcement Learning
3. Apprenticeship Learning
4. Max Entropy Inverse RL

Class Structure

- Last time: Deep reinforcement learning
- This time: Imitation Learning
- Next time: Policy Search