

Lecture 7: Imitation Learning in Large State Spaces²

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2019

²With slides from Katerina Fragkiadaki and Pieter Abbeel

Table of Contents

- 1 Behavioral Cloning
- 2 Inverse Reinforcement Learning
- 3 Apprenticeship Learning

Recap: DQN (Mnih et al. Nature 2015)

$$r_t + \gamma \max_a Q(s_{t+1}, a) - Q$$

$\swarrow \omega$

- DQN uses experience replay and fixed Q-targets
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory \mathcal{D}
- Sample random mini-batch of transitions (s, a, r, s') from \mathcal{D}
- Compute Q-learning targets w.r.t. old, fixed parameters ω^-
- Optimizes MSE between Q-network and Q-learning targets
- Uses stochastic gradient descent
- Achieved human-level performance on a number of Atari games

Recap: Deep Model-free RL, 3 of the Big Ideas

- Double DQN: (Deep Reinforcement Learning with Double Q-Learning, Van Hasselt et al, AAAI 2016)
- Prioritized Replay (Prioritized Experience Replay, Schaul et al, ICLR 2016)
- Dueling DQN (best paper ICML 2016) (Dueling Network Architectures for Deep Reinforcement Learning, Wang et al, ICML 2016)

Recap: Double DQN

$$\Delta \mathbf{w} = \alpha \left(\underbrace{r + \gamma \max_a}_{\text{target}} \underbrace{Q(s', a, \mathbf{w}^-)}_{\text{target}} - Q(s, a, \mathbf{w}) \right)$$

- To help avoid maximization bias, use different weights to select and evaluate actions
- Current Q-network \mathbf{w} is used to select actions
- Older Q-network \mathbf{w}^- is used to evaluate actions

ϵ -greedy

	\rightarrow	\rightarrow
	\mathbf{w}_1	\mathbf{w}_2
$t=1$	e	s
$t=2$	s	e
$t=3$	e	s

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}); \mathbf{w}^-)}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-

Note: how is this different than using a fixed target set of \mathbf{w} ?
(like in DQN)

Recap: Prioritized Experience Replay

- Let i be the index of the i -th tuple of experience (s_i, a_i, r_i, s_{i+1})
- Sample tuples for update using priority function
- Priority of a tuple i is proportional to DQN error

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \mathbf{w}^-) - Q(s_i, a_i; \mathbf{w}) \right|$$

- Update p_i every minibatch, Set $p_i = 0$ for new tuples
- One method¹: proportional (stochastic prioritization)

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad \alpha = 0$$

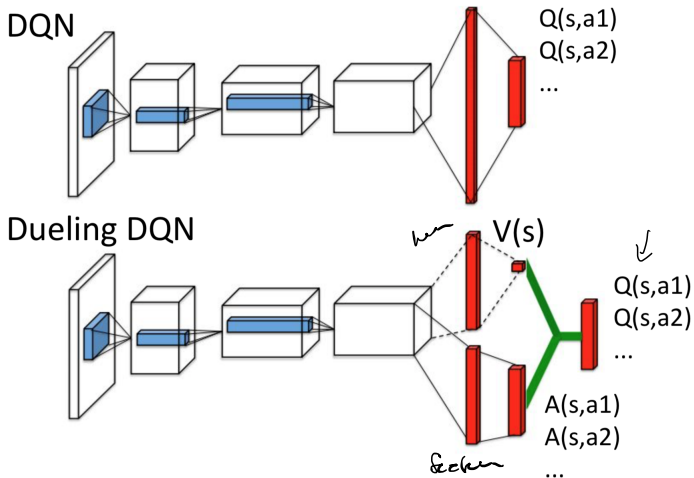
¹See paper for details and an alternative

Dueling Background: Value & Advantage Function

- Intuition: Features needed to determine the value may be different than those needed to specify the action benefit
- E.g.
 - Game score may be relevant to predicting $V(s)$
 - But not necessarily in indicating relative action values
- Advantage function (Baird 1993)

$$A^\pi(s, a) = \underbrace{Q^\pi(s, a)} - \underbrace{V^\pi(s)}_{Q^\pi(s, \pi(s))}$$

Dueling DQN



Wang et.al., ICML, 2016

- Advantage function

$$A^{\pi} = Q^{\pi} - V^{\pi}$$

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

- Identifiable? Given Q^{π} is there a unique A^{π} and V^{π} ?

- Advantage function

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

- Unidentifiable: given Q^π not a unique A^π and V^π
- Option 1: Force $A(s, a) = 0$ if a is action taken

$$\hat{Q}(s, a; \mathbf{w}) = \hat{V}(s; \mathbf{w}) + \left(\hat{A}(s, a; \mathbf{w}) - \underbrace{\max_{a' \in \mathcal{A}} \hat{A}(s, a'; \mathbf{w})}_{\text{max over actions}} \right)$$

- Option 2: Use mean as baseline (more stable)

$$\hat{Q}(s, a; \mathbf{w}) = \hat{V}(s; \mathbf{w}) + \left(\hat{A}(s, a; \mathbf{w}) - \frac{1}{|\mathcal{A}|} \sum_{a'} \hat{A}(s, a'; \mathbf{w}) \right)$$

Dueling DQN V.S. Double DQN with Prioritized Replay

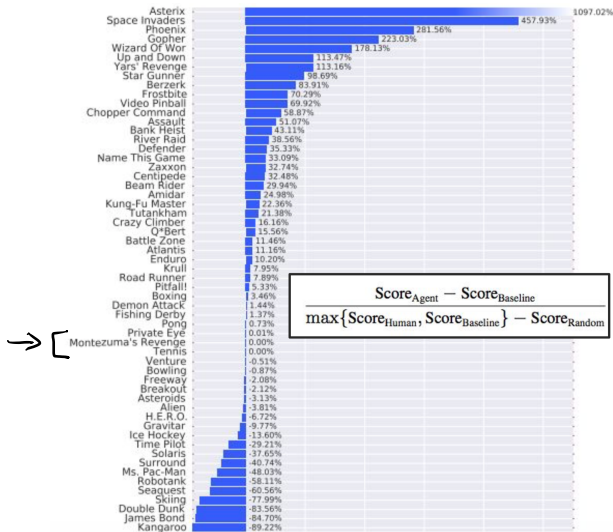


Figure: Wang et al, ICML 2016

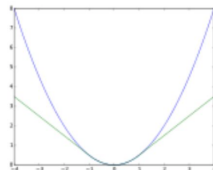
Practical Tips for DQN on Atari (from J. Schulman)

- DQN is more reliable on some Atari tasks than others. Pong is a reliable task: if it doesn't achieve good scores, something is wrong
- Large replay buffers improve robustness of DQN, and memory efficiency is key
 - Use uint8 images, don't duplicate data
- Be patient. DQN converges slowly—for ATARI it's often necessary to wait for 10-40M frames (couple of hours to a day of training on GPU) to see results significantly better than random policy
- In our Stanford class: Debug implementation on small test environment

Practical Tips for DQN on Atari (from J. Schulman) cont.

- Try Huber loss on Bellman error

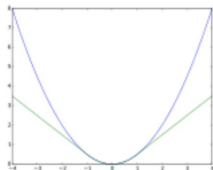
$$L(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq \delta \\ \delta|x| - \frac{\delta^2}{2} & \text{otherwise} \end{cases}$$



Practical Tips for DQN on Atari (from J. Schulman) cont.

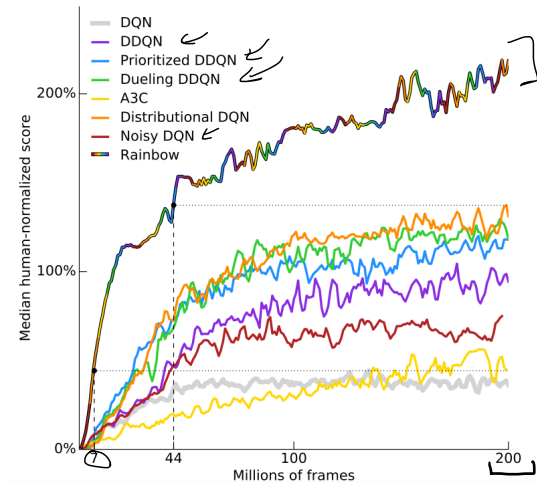
- Try Huber loss on Bellman error

$$L(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq \delta \\ \delta|x| - \frac{\delta^2}{2} & \text{otherwise} \end{cases}$$



- Consider trying Double DQN—significant improvement from small code change in Tensorflow.
- To test out your data pre-processing, try your own skills at navigating the environment based on processed frames
- Always run at least two different seeds when experimenting
- Learning rate scheduling is beneficial: Note in Assignment 2 this is not needed, because we are using the Adam optimizer
- Try non-standard exploration schedules

Deep Reinforcement Learning



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

Summary of Model Free Value Function Approximation with DNN

- DNN are very expressive function approximators
- Can use to represent the Q function and do MC or TD style methods
- Should be able to implement DQN (assignment 2)
- Be able to list a few extensions that help performance beyond DQN

We want RL Algorithms that Perform

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

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 with Large State Spaces**
- Next time: Policy Search

Consider Montezuma's revenge

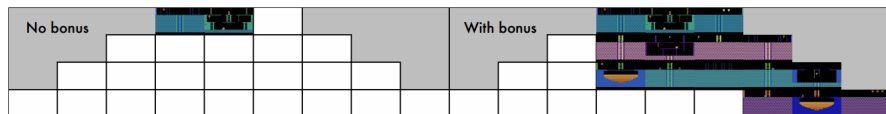


Figure 3: “Known world” of a DQN agent trained for 50 million frames with (**right**) and without (**left**) count-based exploration bonuses, in MONTEZUMA’S REVENGE.

- 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

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

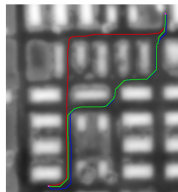
Simulated highway driving

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



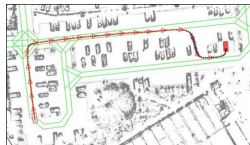
Aerial imagery-based navigation

- Ratliff, Bagnell, and Zinkevich, ICML 2006



Parking lot navigation

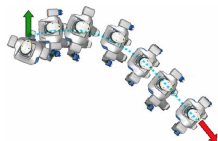
- Abbeel, Dolgov, Ng, and Thrun, IROS 2008



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

*Inverse RL
Imitation learning*

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

Problem Setup

- 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: $\hat{\pi}^*$
 - 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?

Inverse RL + IRL

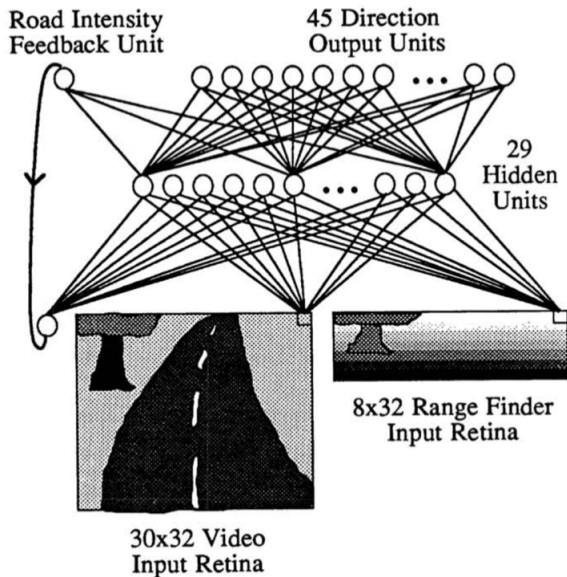
Table of Contents

1 Behavioral Cloning

2 Inverse Reinforcement Learning

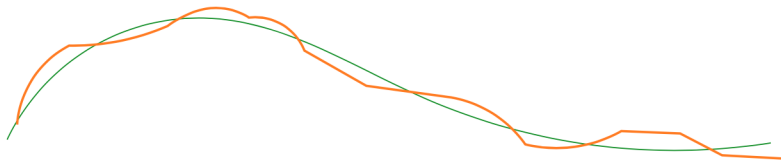
3 Apprenticeship Learning

- Formulate problem as a standard machine learning problem: $s \rightarrow a$
 - 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



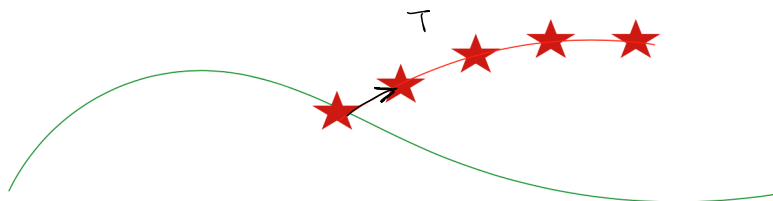
Problem: Compounding Errors

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



Error at time t with probability ϵ
 $\mathbb{E}[\text{Total errors}] \leq \epsilon T$

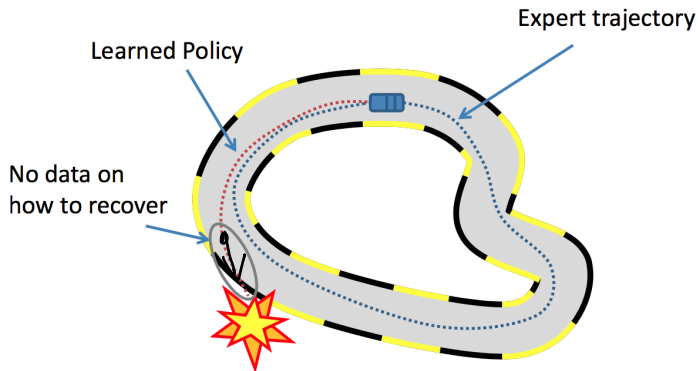
Problem: Compounding Errors



Error at time t with probability ϵ ϵT
 $\mathbb{E}[\text{Total errors}] \leq \epsilon(T + (T - 1) + (T - 2) \dots + 1) \propto \underline{\epsilon T^2}$

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** *expert behavior*

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.

- Idea: Get more labels of the expert 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

Feature Based Reward Function

- Given 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^*)
- 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 ?

○

not unique

Andrew Ng
Stanford Research
2000

Linear Feature Reward Inverse RL

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

$$V^\pi = \mathbb{E}_{s \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \underbrace{R(s_t)}_{\mathbf{w}^T \mathbf{x}(s_t)} \right] \quad (1)$$

Linear Feature Reward Inverse RL

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

$$V^\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \underbrace{\mathbf{w}^T}_{\text{weight}} \underbrace{\mathbf{x}(s_t)}_{\text{feature}} \mid \pi\right] \quad (2)$$

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

$$= \mathbf{w}^T \mu(\pi) \quad (4)$$

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

Table of Contents

- 1 Behavioral Cloning
- 2 Inverse Reinforcement Learning
- 3 Apprenticeship Learning**

Linear Feature Reward Inverse RL

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

$$V^\pi = \mathbf{w}^T \mu(\pi) \quad (5)$$

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

- Note $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] = V^* \geq V^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] \quad \forall \pi$,
where arrows indicate that V^* is the value of the optimal policy and V^π is the value of any other policy.
- Therefore if the expert's demonstrations are from the optimal policy, to identify \mathbf{w} it is sufficient to find \mathbf{w}^* such that

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

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 policy⁴².
- More precisely, if

$$\left. \begin{array}{l} \overbrace{\|\mu(\pi) - \mu(\pi^*)\|_1}^{(s)} \leq \epsilon \\ \swarrow \text{demonstrates} \end{array} \right\} \quad (7)$$

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

$$\left. \begin{array}{l} \swarrow \\ |w^T \mu(\pi) - w^T \mu(\pi^*)| \leq \epsilon \end{array} \right\}$$

⁴²Abbeel 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 $i = 1, 2, \dots$ \mathbf{w}
 - Find a reward function such that the teacher maximally outperforms all previous controllers:

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

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

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?

- Many different approaches
- Two of the key papers are:
 - Maximum Entropy Inverse Reinforcement Learning (Ziebart et al. AAI 2008)
 - Generative adversarial imitation learning (Ho and Ermon, NeurIPS 2016)

GAIL

DNN

Transition model unknown

Summary

- Imitation learning can greatly reduce the amount of data need to learn a good policy
- Challenges remain and one exciting area is combining inverse RL / learning from demonstration and online reinforcement learning

Class Structure

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