1 Behavioral Cloning

2 Inverse Reinforcement Learning

3 Apprenticeship Learning

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

\[ r_t + \gamma \max_a Q(s_{t+1}, a) - Q \]
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)
Recap: Double DQN

\[ \Delta w = \alpha (r + \gamma \max_a \hat{Q}(s', a; w^-) - \hat{Q}(s, a; w)) \]

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

Action evaluation: \( w^- \)

\[ \Delta w = \alpha (r + \gamma \hat{Q}(\text{arg max}_{a'} \hat{Q}(s', a'; w); w^-) - \hat{Q}(s, a; w)) \]

Action selection: \( w \)

Note: how is this different than using a fixed target set of \( w \)? (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
  \[ p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; w^-) - Q(s_i, a_i; w) \right| \]
- Update $p_i$ every minibatch, Set $p_i = 0$ for new tuples
- One method\(^1\): proportional (stochastic prioritization)
  \[ P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad \alpha = 0 \]

\(^1\)See paper for details and an alternative
**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)**

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

Wang et al., ICML, 2016
**Advantage function**

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

**Identifiable? Given** \( Q^\pi \) **is there a unique** \( A^\pi \) **and** \( V^\pi \) **?**
Identifiability

- Advantage function

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

- Unidentifiable: given \( Q^\pi \) not a unique \( A^\pi \) and \( V^\pi \)

- Option 1: Force \( A(s, a) = 0 \) if \( a \) is action taken

\[ \hat{Q}(s, a; \mathbf{w}) = \hat{V}(s; \mathbf{w}) + \left( \hat{A}(s, a; \mathbf{w}) - \max_{a' \in \mathcal{A}} \hat{A}(s, a'; \mathbf{w}) \right) \]

- 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.
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.
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, ...).
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
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

Inversion 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 $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from teacher’s policy $\pi^*$)  

- **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?

**Inverse RL + IRC**
Table of Contents

1 Behavioral Cloning

2 Inverse Reinforcement Learning

3 Apprenticeship Learning
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
Problem: Compounding Errors

Supervised learning assumes iid. \((s, a)\) pairs and ignores temporal structure

Independent in time errors:

Error at time \(t\) with probability \(\epsilon\)

\[\mathbb{E}[\text{Total errors}] \leq \epsilon T\]
Problem: Compounding Errors

Error at time $t$ with probability $\epsilon$

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

---

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

for $i = 1$ to $N$ do

Let $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$.
Sample $T$-step trajectories using $\pi_i$.
Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by $\pi_i$ 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 $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from teacher’s policy $\pi^*$)
- 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 and Stuart Russell 2000
Recall linear value function approximation

Similarly, here consider when reward is linear over features

\[ R(s) = w^T x(s) \text{ where } w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n \]

Goal: identify the weight vector \( w \) given a set of demonstrations

The resulting value function for a policy \( \pi \) can be expressed as

\[
V^\pi = \mathbb{E}_{s \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \right]
\]
Linear Feature Reward Inverse RL

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

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

where \( \mu(\pi)(s) \) is defined as the discounted weighted frequency of state features under policy \( \pi \).
Recall linear value function approximation

Similarly, here consider when reward is linear over features

\[ R(s) = w^T x(s) \text{ where } w \in \mathbb{R}^n, x: S \rightarrow \mathbb{R}^n \]

Goal: identify the weight vector \( w \) given a set of demonstrations

The resulting value function for a policy \( \pi \) can be expressed as

\[ V^\pi = w^T \mu(\pi) \quad (5) \]

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

Note

\[
\mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^* \right] = V^* \geq V^\pi = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi \right] \quad \forall \pi,
\]

Therefore if the expert’s demonstrations are from the optimal policy, to identify \( w \) it is sufficient to find \( w^* \) such that

\[ w^*^T \mu(\pi^*) \geq w^*^T \mu(\pi), \forall \pi \neq \pi^* \quad (6) \]
Want to find a reward function such that the expert policy outperforms other policies.

For a policy $\pi$ to be guaranteed to perform as well as the expert policy $\pi^*$, it suffices that we have a policy such that its discounted summed feature expectations match the expert’s policy\(^{42}\).

More precisely, if
\[
\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon
\]
then for all $w$ with $\|w\|_\infty \leq 1$:
\[
|w^T \mu(\pi) - w^T \mu(\pi^*)| \leq \epsilon
\]

\(^{42}\) Abbeel and Ng, 2004
This observation leads to the following algorithm for learning a policy that is as good as the expert policy:

**Assumption:** \( R(s) = w^T x(s) \)

1. Initialize policy \( \pi_0 \)
2. For \( i = 1, 2 \ldots \)
   
   - Find a reward function such that the teacher maximally outperforms all previous controllers:
     
     \[
     \arg \max_w \max_{\gamma} s.t. \ w^T \mu(\pi^*) \geq w^T \mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\} 
     \]
     
     (8)
   
   - s.t. \( \|w\|_2 \leq 1 \)
   
   - Find optimal control policy \( \pi_i \) for the current \( w \)
   
   - 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. AAAI 2008)
- Generative adversarial imitation learning (Ho and Ermon, NeurIPS 2016)

\[ \text{GAIL} \quad \text{DNN} \]
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