Imitation Learning in Large State Spaces[1]

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2023

[1] With slides from Katerina Fragkiadaki and Pieter Abbeel

Table of Contents

Midterm

Which of the following are valid equations for $V^\pi(s_t)$ for a state s_t in an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$? Note that s_t is the state at time t and the action at this timestep would be a_t and the next state would be s_{t+1} and so on. The policy π is stochastic and τ represents a trajectory.

1. $\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i)$ where the actions a_i are sampled from the policy
2. $\mathbb{E}_{\tau \sim \pi}[\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i)]$
3. $r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{P}}[V^\pi(s_{t+1})]$
4. $r(s_t, a_t) + \gamma V^\pi(s_{t+1})$
5. $\max_a [Q^\pi(s_t, a)]$
6. $\mathbb{E}_{a \sim \pi}[Q^\pi(s_t, a)]$

Midterm

Which of the following are true about REINFORCE? In the following options, PG stands for policy gradient.

a. Adding a baseline term can help to reduce the variance of the PG updates
b. It will converge to a global optima
c. It can be initialized with a sub-optimal, deterministic policy and still converge to a local optima, given the appropriate step sizes
d. If we take one step of PG, it is possible that the resulting policy gets worse (in terms of achieved returns) than our initial policy

Midterm

Consider an MDP $\mathcal{M}$. $\mathcal{M}$ has three states A, B, C and a terminal state and two actions a_1, a_2.
You observe the following episode from $\mathcal{M}$. The labels above the arrows are actions and below are the rewards.

$$A \xrightarrow[+5]{a_1} B \xrightarrow[-2]{a_1} A \xrightarrow[+3]{a_2} C \xrightarrow[-2]{a_1} B \xrightarrow[+4]{a_2} C \xrightarrow[-2]{a_2} \text{terminal}$$

At each time point, the greedy deterministic behaviour policy depends only on the current state. Could this trajectory be generated by Monte Carlo control reinforcement learning? Could this trajectory have been generated by SARSA? Justify your explanations.

Homework 2

Given a stream of batches of n environment interactions (s_i, a_i, r_i, s_i'), we want to learn the optimal value function using a neural network. The underlying MDP has a finite-sized action space. Your friend first suggests the following approach:

- Initialize parameters ϕ of a neural network V_ϕ
- For each batch of k tuples (s_i, a_i, r_i, s_i') (sampled at random), do stochastic gradient descent with the loss function $\sum_{i=0}^{k} |y_i - V_\phi(s_i)|^2$, where $y_i = \max_{a_i}[r_i + \gamma V_\phi(s_i')]$, where the $\max_{a_i}$ is taken over all the tuples in the batch of the form $(s_i, a_i, *, *)$.

What is the problem with this approach?

PPO Subtleties

- In class we covered some core ideas underlying modern policy gradient approaches
- There are multiple other common algorithmic changes that people often employ, including
  - Entropy regularization which can be decayed
  - Normalizing the advantages
  - Using Generalized Advantage Estimation
  - Tuning the batch size and the number of steps per batch
  - Ensuring the action range output is automatically within the desired range
- Multiple papers discuss this issue including:
  - "Implementation matters in deep RL: A case study on PPO and TRPO" https://openreview.net/forum?id=r1etN1rtPB .
  - "Revisiting Design Choices in Proximal Policy Optimization" https://arxiv.org/abs/2009.10897,
  - for actor critic algorithms "What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study" https://openreview.net/forum?id=nIAxjsniDzg

Refresh Your Understanding

- Select all that are true:
  1. In Thompson sampling for MDPs, the posterior over the dynamics can be updated after each transition
  2. When using a Beta prior for a Bernoulli reward parameter for an (s,a) pair, the posterior after N samples of that pair time steps can be the same as after N+2 samples
  3. The optimism bonuses discussed for MBIE-EB depend on the maximum reward but not on the maximum value function
  4. In class we discussed adding a bonus term to an update for a (s,a,r,s') tuple using Q-learning with function approximation. Adding this bonus term will ensure all Q estimates used to make decisions online using DQN are optimistic with respect to Q*
  5. Not sure

Refresh Your Understanding

- Select all that are true:
  1. In Thompson sampling for MDPs, the posterior over the dynamics can be updated after each transition
  2. When using a Beta prior for a Bernoulli reward parameter for an (s,a) pair, the posterior after N samples of that pair time steps can be the same as after N+2 samples
  3. The optimism bonuses discussed for MBIE-EB depend on the maximum reward but not on the maximum value function
  4. In class we discussed adding a bonus term to an update for a (s,a,r,s') tuple using Q-learning with function approximation. Adding this bonus term will ensure all Q estimates used to make decisions online using DQN are optimistic with respect to Q*
  5. Not sure

Class Structure

- Last time: Fast Learning
- This time: Imitation Learning
- Next time: Batch offline RL

Table of Contents

Learning from Past Decisions and Outcomes

In some settings there exist very good decision policies and we would like to automate them

- One idea: humans provide reward signal when RL algorithm makes decisions
- Good: simple, cheap form of supervision
- Bad: High sample complexity

Alternative: imitation learning

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]
- Parking lot navigation [Abbeel, Dolgov, Ng, and Thrun, IROS 2008]

Learning from Demonstrations

- 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

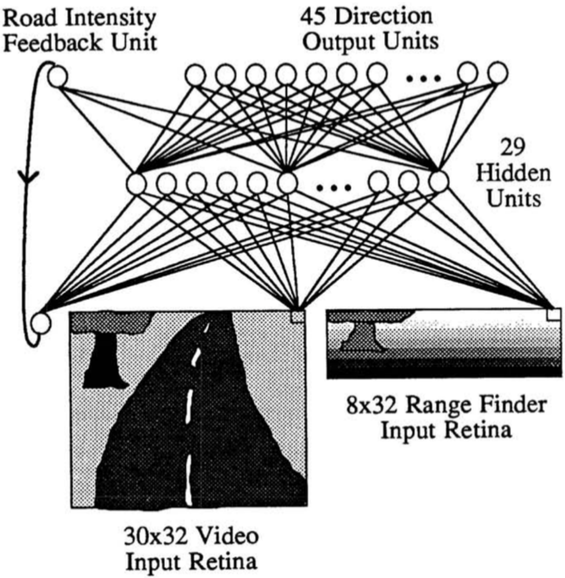
- Input:
  - State space, action space
  - Transition model $P(s' \mid s, a)$
  - No reward function R
  - Set of one or more teacher's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from teacher's policy π^*)
- Behavioral Cloning:
  - Can we directly learn the teacher's policy using supervised learning?
- Inverse RL:
  - Can we recover R?
- Apprenticeship learning via Inverse RL:
  - Can we use R to generate a good policy?

Table of Contents

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

Table of Contents

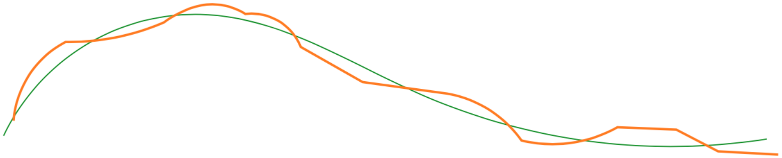
Behavioral cloning

- Often behavior cloning in practice can work very well, especially if use BCRNN
- See What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. Mandlekar et al. CORL 2021

Table of Contents

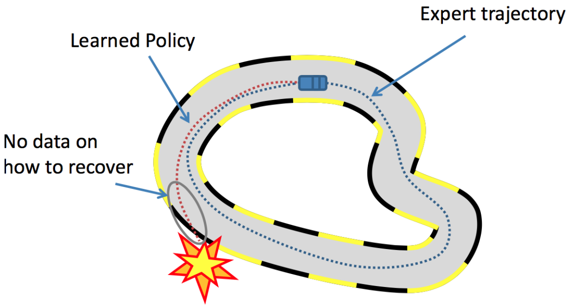
Potential Problem with Behavior Cloning: Compounding Errors

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



Error at time t with probability $<= \epsilon$
$\mathbb{E}$[Total errors] $<= \epsilon T$

Problem: Compounding Errors



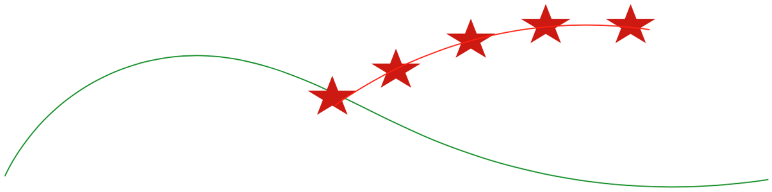
Data distribution mismatch!

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

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

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

Problem: Compounding Errors



- Error at time t with probability ϵ
- Approximate intuition: $\mathbb{E}$[Total errors] $\leq \epsilon(T + (T-1) + (T-2) \ldots + 1) \propto \epsilon T^2$
- Real result requires more formality. See Theorem 2.1 in http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats10-paper.pdf with proof in supplement: http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats10-sup.pdf

---

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

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.

- 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
- Key limitation?

Table of Contents

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 expert's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from teacher's policy π^*)
- Goal: infer the reward function R
- Assume that the teacher's policy is optimal. What can be inferred about R?

Check Your Understanding: 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 $(s_0, a_0, s_1, s_0, \dots)$ (actions drawn from teacher's policy π^*)
- Goal: infer the reward function R
- Assume that the teacher's policy is optimal.

1. There is a single unique R that makes teacher's policy optimal
2. There are many possible R that makes teacher's policy optimal
3. It depends on the MDP
4. Not sure

Check Your Understanding: 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
- Assume that the teacher's policy is optimal.

1. There is a single unique R that makes teacher's policy optimal
2. There are many possible R that makes teacher's policy optimal
3. It depends on the MDP
4. Not sure

.

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 $\boldsymbol{w} \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi(s_0) = \mathbb{E}_{s \sim \pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0]$$

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

$$
\begin{aligned}
V^\pi(s_0) = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0] \quad &= \mathbb{E}_{s\sim\pi}[\textstyle\sum_{t=0}^{\infty} \gamma^t \boldsymbol{w}^T x(s_t) \mid s_0] \\
&= w^T \mathbb{E}_{s\sim\pi}[\textstyle\sum_{t=0}^{\infty} \gamma^t x(s_t) \mid s_0] \\
&= \boldsymbol{w}^T \mu(\pi)
\end{aligned}
$$

- where $\mu(\pi)(s)$ is defined as the discounted weighted frequency of state features under policy π, starting in state s_0.

Relating Frequencies to Optimality

- Assume $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- $V^\pi = \mathbb{E}_{s \sim \pi}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] = \boldsymbol{w}^T \mu(\pi)$ where $\mu(\pi)(s) =$ discounted weighted frequency of state s under policy π.

$$V^* \geq V^\pi$$

Relating Frequencies to Optimality

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

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

- $\mu(\pi)(s) =$ discounted weighted frequency of state s under policy π.

$$\mathbb{E}_{s\sim\pi^*}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] = V^* \geq V^\pi = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] \quad \forall\pi$$

- Therefore if the expert's demonstrations are from the optimal policy, to identify $\boldsymbol{w}$ it is sufficient to find w^* such that

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

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^*, sufficient if its discounted summed feature expectations match the expert's policy [Abbeel & Ng, 2004].
- More precisely, if

$$\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon$$

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

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

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?

Learning from Demonstration / Imitation Learning Pointers

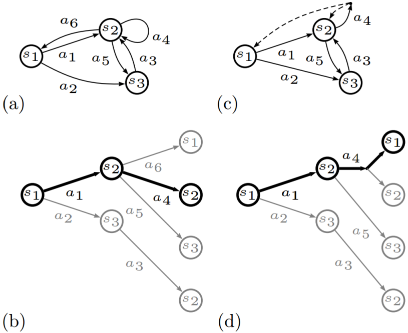
- Many different approaches
- Two of the key papers are:
  - Maximumum Entropy Inverse Reinforcement Learning (Ziebart et al. AAAI 2008)
  - Generative adversarial imitation learning (Ho and Ermon, NeurIPS 2016)

Table of Contents

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

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

- 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[1].

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

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

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

[1] 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

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

- where $D(s_i)$: state visitation frequency
- Do we need to know the transition model to compute the above?

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

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 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?

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
- For a look into some of the theory between imitation learning and RL, see Sun, Venkatraman, Gordon, Boots, Bagnell (ICML 2017)

Imitation learning: What You Should Know

- Define behavior cloning and how it differs from reinforcement learning

Class Structure

- Last time: Learning from offline data, overview and policy evaluation
- This time: Learning from offline data, policy evaluation and imitation learning
- Next time: Learning from offline data, policy optimization / learning