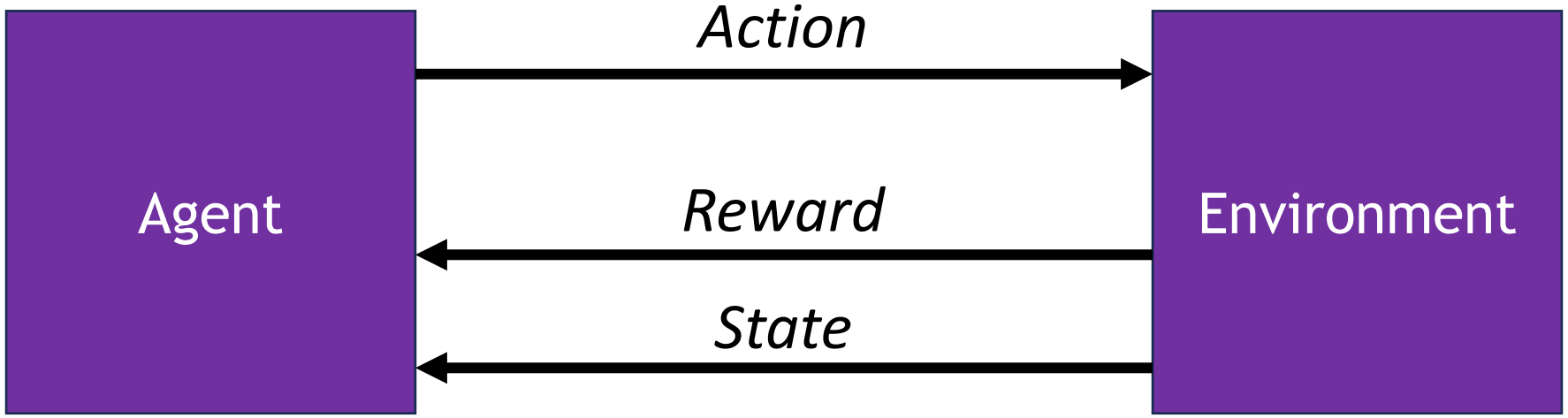
Learning from Demonstrations

Iftekharul Islam

Contents

Motivation

Learning from Rewards (Reinforcement Learning)

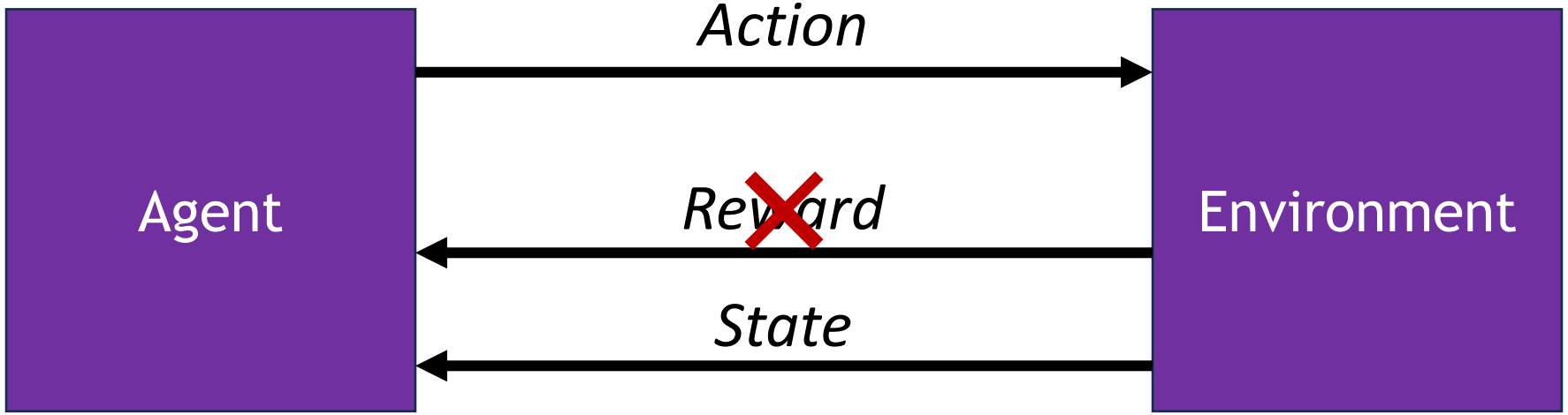


- **Goal:** Learn to choose actions that maximize agent's rewards

$$s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_n, a_n, r_n \rightarrow \pi^*(a|s)$$

This becomes ineffective when "experience" is expensive (e.g., slow action, safety concerns)

Learning from Demonstrations (Imitation Learning)



- **Goal:** Learn to choose actions that imitate an expert's policy

$$s_1, a_1^*, s_2, a_2^*, \ldots, s_n, a_n^* \rightarrow \pi^*(a|s)$$

Learning from Demonstrations (Imitation Learning)

- Useful when it's difficult to specify reward for desired behavior or desired policy.

- For example, driving has many desired features: time and speed to reach goal, comfort, traffic rules, social conventions, etc.

- It's difficult to design a numerical reward to encode all these features without producing unexpected behaviors.

Beyond Learning from Rewards

- Learning from rewards
  - effective when “experience” is cheap
  - ineffective when “experience” is expensive (e.g., slow action, safety concerns) since RL has high sample complexity

- Learning from demonstrations
  - do not need reward to begin with
  - learn from expert’s demonstrations

Behavior Cloning

Behavior Cloning

- Goal: mimic (but not overfit) the expert's policy by applying supervised learning to expert's demonstrations.
- Does not work in theory due to covariate shift, i.e., $training\ distribution \neq test\ distribution$.
- Essentially, we are addressing non-i.i.d. task using method (i.e., supervised learning) based on i.i.d. assumption.

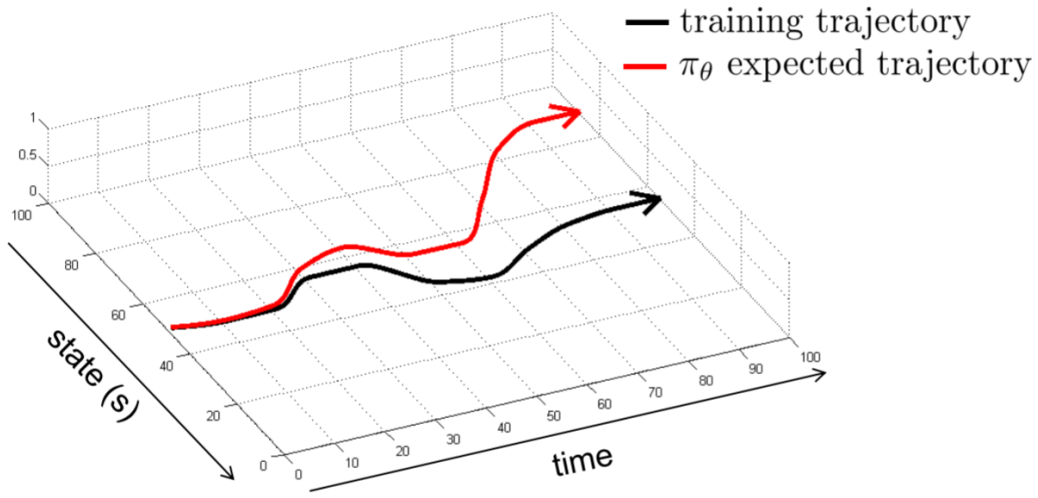


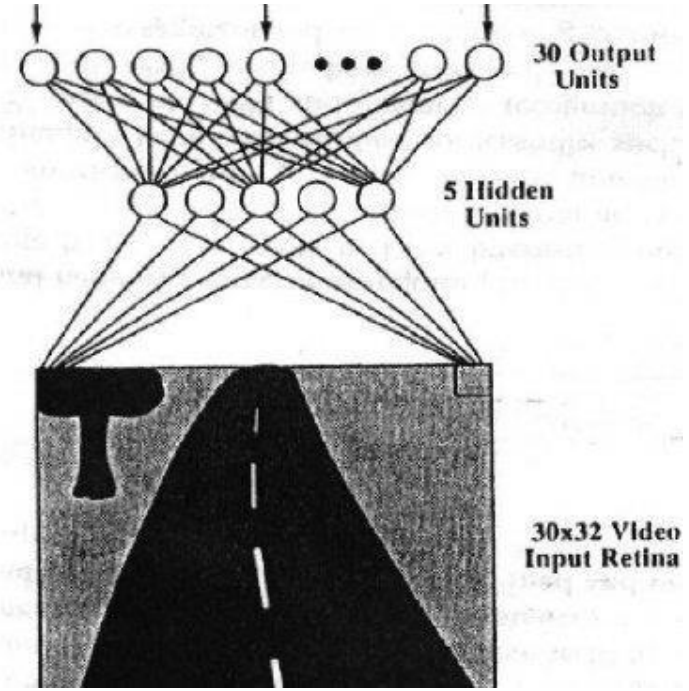
Image: Sergey Levine, CS 285 Deep RL

Supervised Learning vs. Behavior Cloning

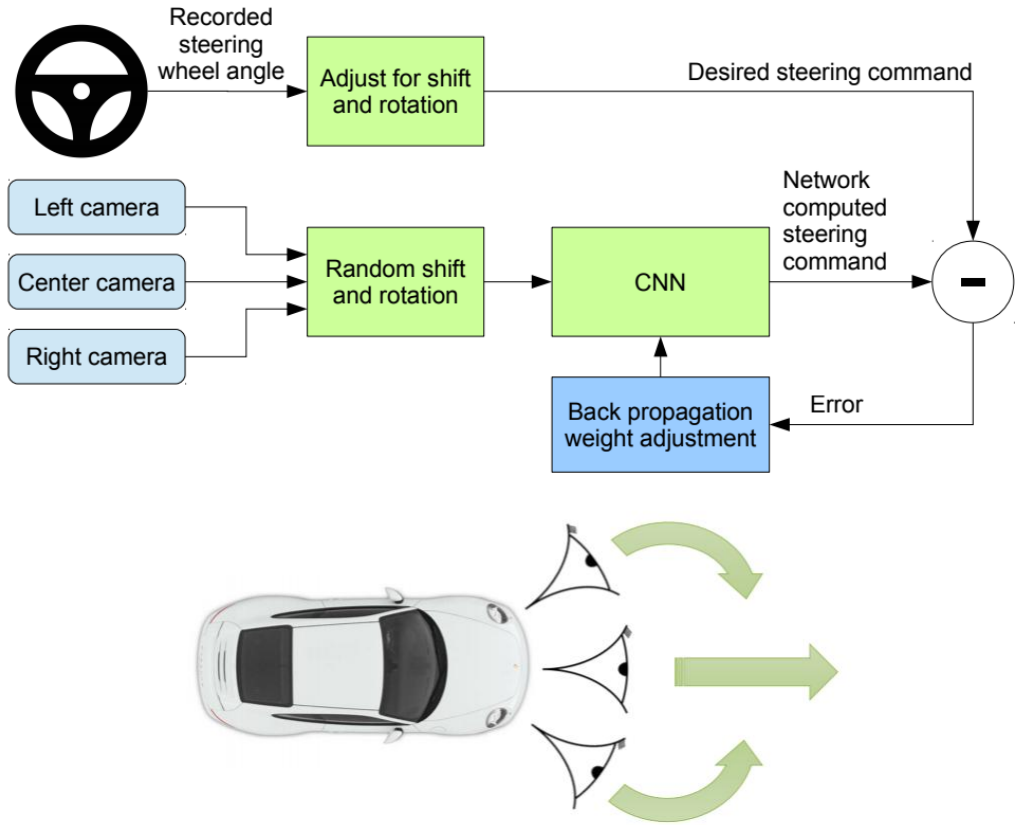
- Error of supervised learning: $O(\epsilon T)$.
- Error of behavior cloning: $O(\epsilon T^2)$.
- ϵ is learning error, T is task horizon.

Case Study I: Autonomous Driving

- ALVINN: **A**utonomous **L**and **V**ehicle **I**n a **N**eural **N**etwork (1989)

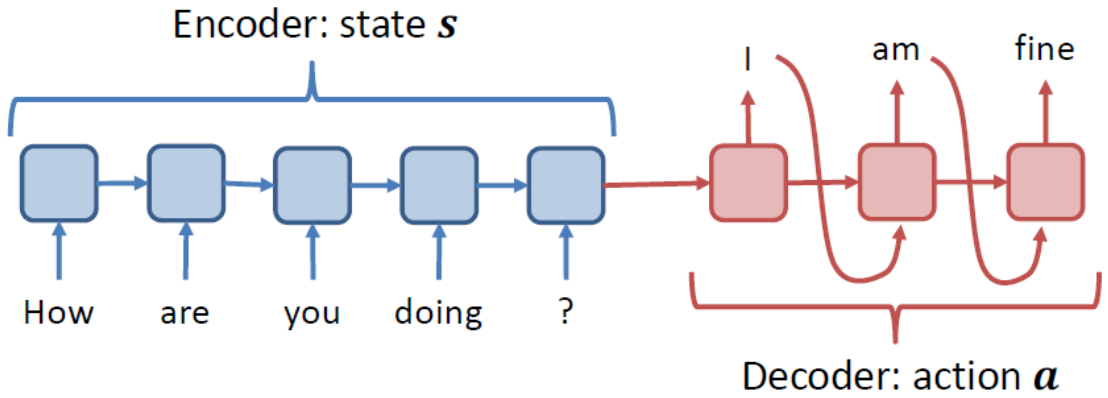


Case Study I: Autonomous Driving



Bojarski et al. '16, NVIDIA

Case Study II: Conversational Agents



Objective: $\max_{\mathbf{a}} \Pr(\boldsymbol{a}|\boldsymbol{s}) = \prod_i \Pr(a_i|a_{i-1}, \ldots, a_1, \boldsymbol{s})$

#	Context c	Message m	Generated Response r
1	noone can help this time its just i need a support system	well im here to support you. whenever you need me.	thank you, i appreciate it thank you
2	nothin much, and how's the book?!	its good but i'm only like halfway through cuz i don't feel like reading. i'm so bored ...	that's good! i have the book but i'm bored too.
3	i know! what have you been up to?	nothing really. just enjoying summer and working. what about you?	that's good! nothing much just listening to music and you?

Image: Pascal Poupart, Reinforcement Learning | Table: Sordoni et al., 2015

Behavior Cloning in Practice

- Behavior cloning works better if we have many (successful) trajectories for training since those trajectories are likely to embed error-correction data.

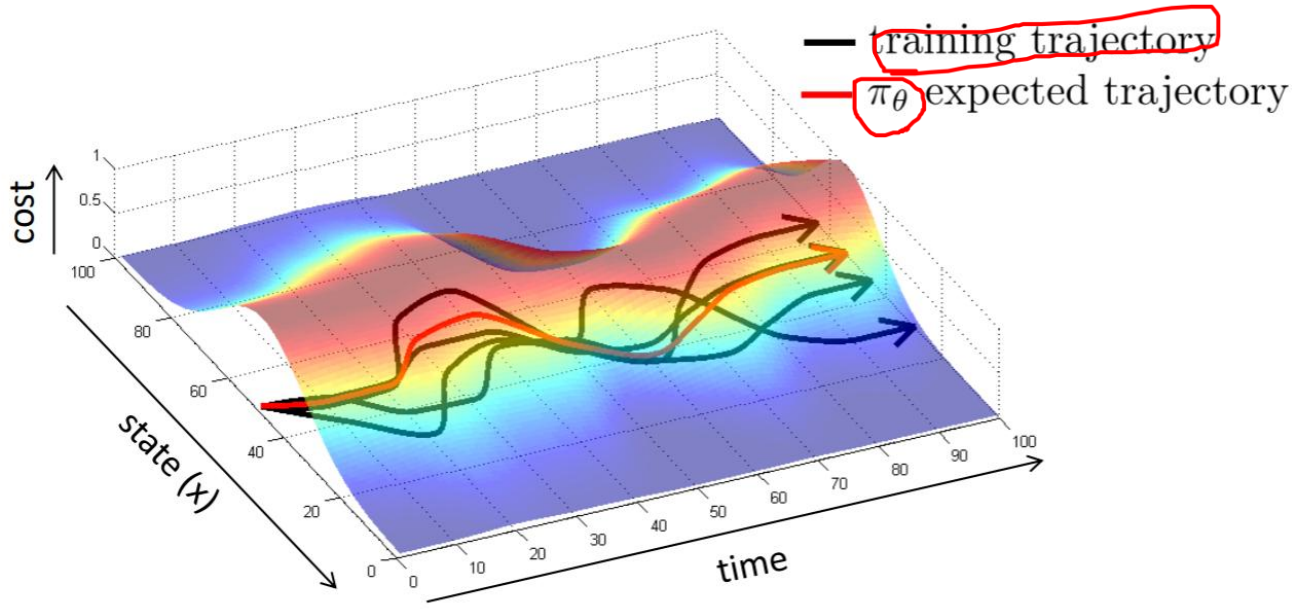


Image: Sergey Levine, CS 285 Deep RL

Behavior Cloning in Practice

- Pomerleau (1989): *the network must not solely be shown examples of accurate driving, **but also how to recover** (i.e. return to the road center) once a mistake has been made.*

DAgger: Dataset Aggregation

- DAgger: “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning” by Ross et al, 2011.

- Procedure
  - Step 1: train a policy using expert’s data
  - Step 2: run the trained policy, collect the “deviate data” (i.e., collecting observations)
  - Step 3: ask expert to label the “deviate data” (i.e., providing actions)
  - Step 4: aggregate all labelled data and re-train the policy
  - Step 5: go to Step 2 if needed

DAgger: Dataset Aggregation

- DAgger alleviates covariate shift by making training distribution ≈ test distribution. Thus, the error of DAgger is $O(\epsilon T)$.

- Essentially, DAgger achieves $O(\epsilon T)$ by converting the notion of i.i.d. data points of supervised learning to i.i.d. trajectories in online learning.

- In practice, DAgger can work well using a small number of iterations and is widely adopted because of its effectiveness and simplicity.

DAgger: Dataset Aggregation

Limitations

- Assume we can “reset” the agent to obtain i.i.d. trajectories.
- Need humans to label data
  - inefficient (deep learning needs data)
  - judgemental errors
  - unnatural for humans to provide labels for many tasks (e.g., steering angle based on an image, joint angles for a robot)

Cases Where Behavior Cloning Can Fail

- **Case 1: non-markovian behavior**—expert's action may not only depend on the current observation (e.g., single frame).
  - **Solution 1:** concatenate multiple observations as one training example (e.g., DQN on Atari)

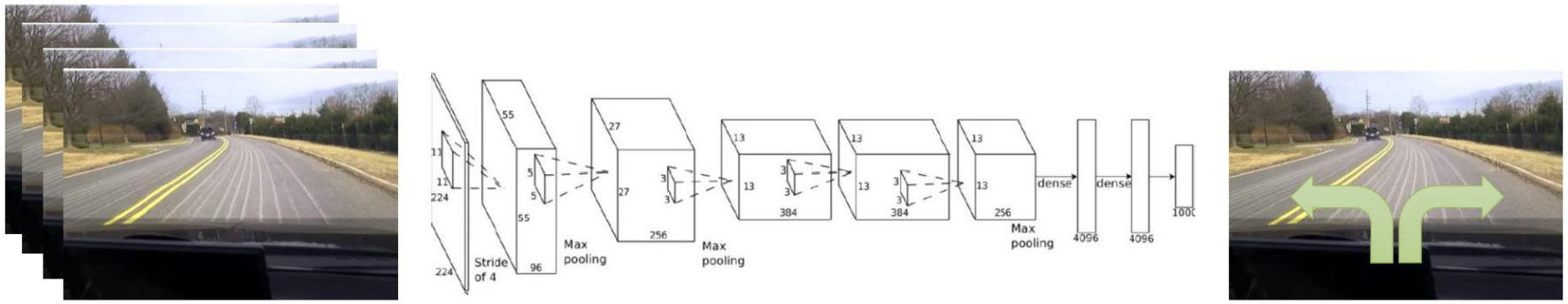


Image: Sergey Levine, CS 285 Deep RL

Cases Where Behavior Cloning Can Fail

- **Case 1: non-markovian behavior**—expert's action may not only depend on the current observation (e.g., single frame).
  - **Solution 2:** use Transformer or RNN

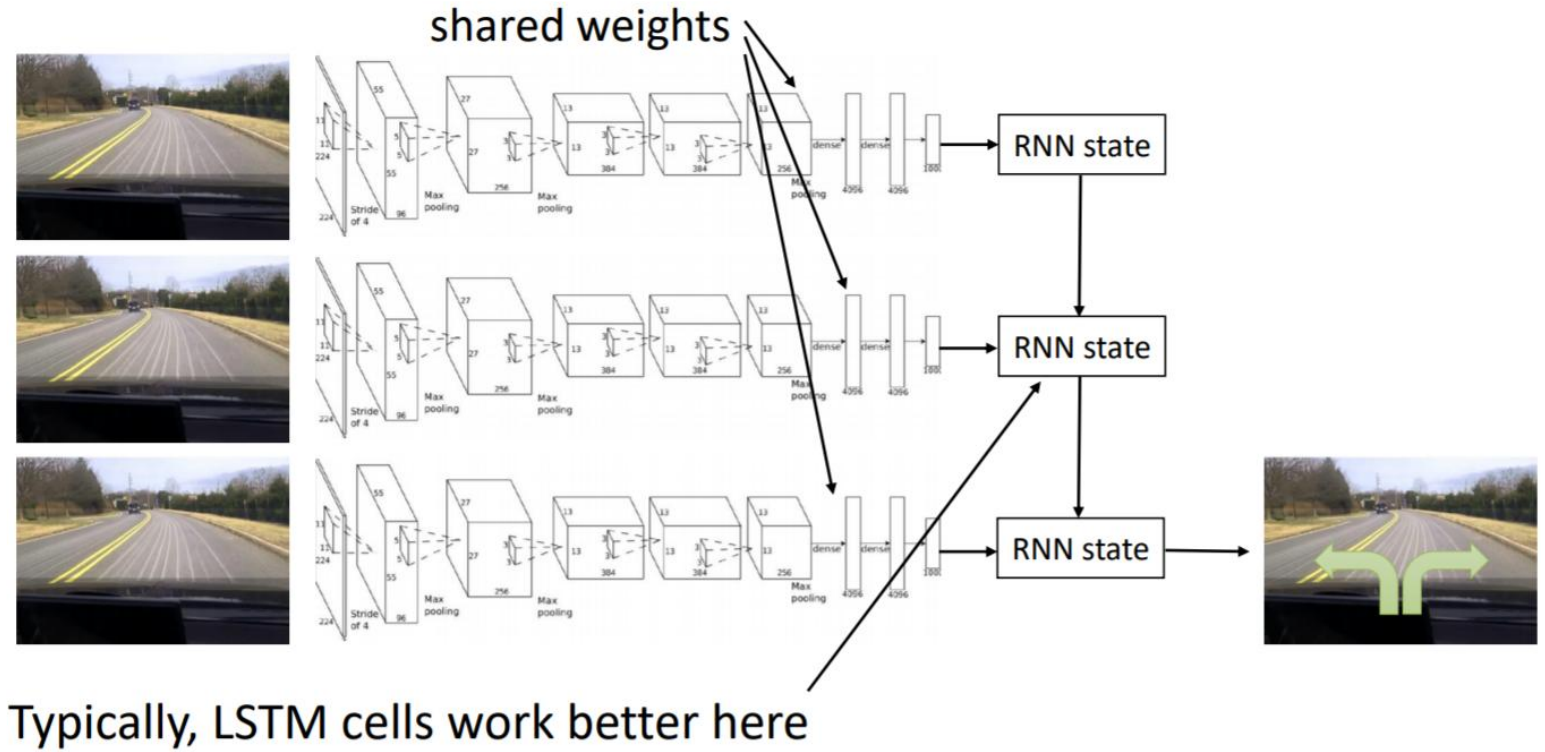


Image: Sergey Levine, CS 285 Deep RL

Cases Where Behavior Cloning Can Fail

- **Case 2: multimodal behavior**—when there are **multiple valid actions** for one observation

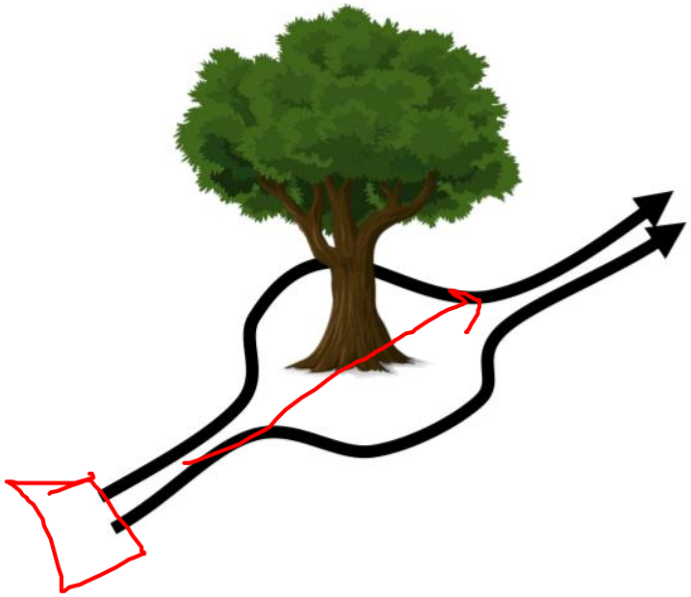


Image: Sergey Levine, CS 285 Deep RL

Cases Where Behavior Cloning Can Fail

- **Case 2: multimodal behavior**—when there are **multiple valid actions** for one observation while their "average" is bad.
  - **Solution:** output mixture of n Gaussians, n=number of valid behaviors.

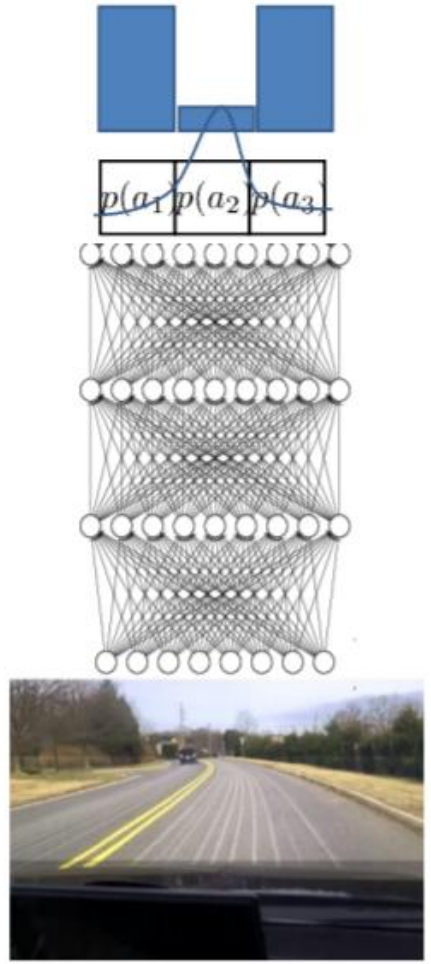


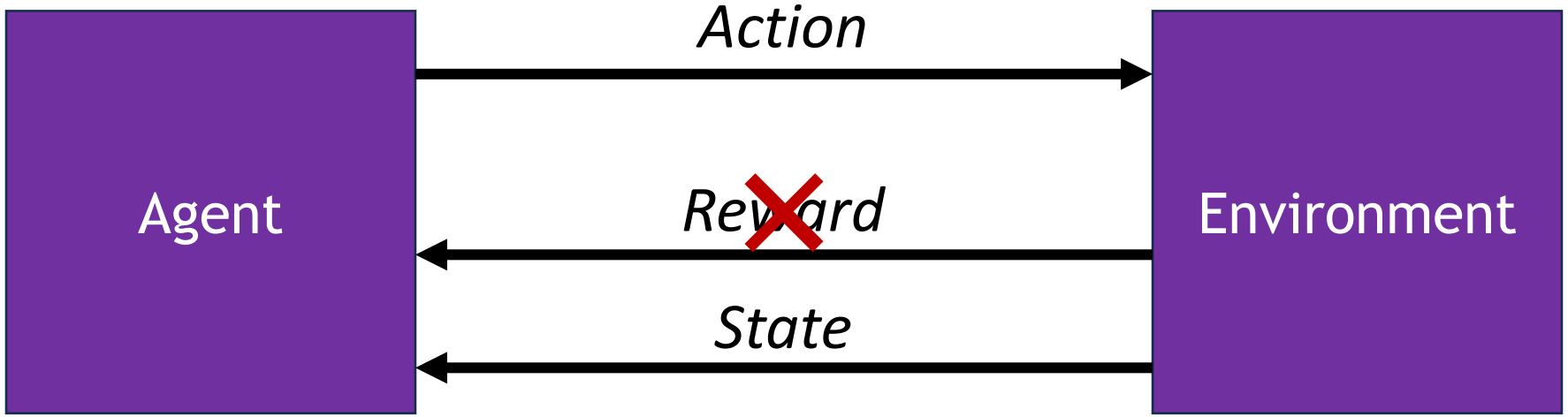
Image: Sergey Levine, CS 285 Deep RL

Other Imitation Learning Ideas

- “Learning Latent Plans from Play”
- “Learning to Reach Goals via Iterated Supervised Learning”

Inverse Reinforcement Learning

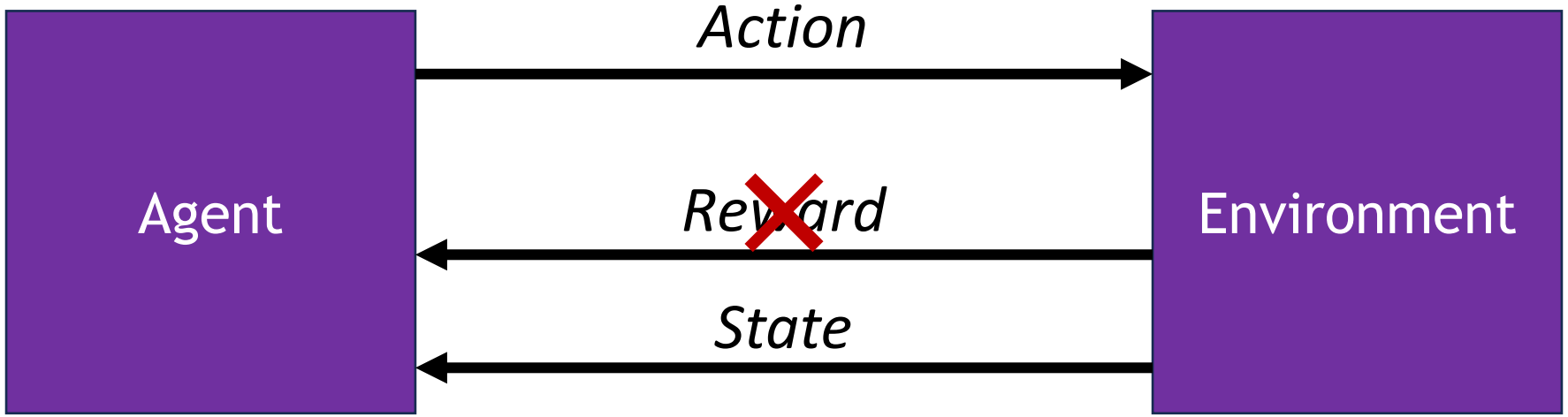
Imitation Learning



- **Goal:** Learn to choose actions that imitate an expert's policy

$$s_1, a_1^*, s_2, a_2^*, \dots, s_n, a_n^* \rightarrow \pi^*(a|s)$$

Inverse Reinforcement Learning



- **Goal:** Find the reward function that the expert is implicitly optimizing

Inverse Reinforcement Learning

- Definition
  - States: $s \in S$
  - (Near) optimal actions: $a^* \in A$
  - Rewards: $r \in \mathbb{R}$
  - Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - Reward model: $R(s,a) = E[r|s,a]$

  } unknown model

  - Discount factor: $0 \leq \gamma \leq 1$
    - discounted: $\gamma < 1$ undiscounted: $\gamma = 1$
  - Horizon (i.e., # of time steps): h
    - Finite horizon: $h \in \mathbb{N}$ infinite horizon: $h = \infty$

- Goal: find reward model $R(s,a) = E[r|s,a]$ such that

$$\pi^* = argmax_\pi \sum_{t=0}^{h} \gamma^t E_\pi[E[r_t|s_t, a_t]]$$

Transfer Learning

- Why recover R when we have π^*?

- Think about navigation planning (multiple competing goals: travel time, distance covered, and fuel efficiency)
  - Suppose we know the route (policy) followed by taxi drivers between any pair of locations in city A
  - An autonomous car can follow the same policy in city A
  - What about city B?

- If we can identify what objectives the taxi drivers were optimizing in City A, we can apply those same objectives to find appropriate routes in City B that share similar characteristics.

General Approach

- Use IRL to learn reward function
- Then use reward function to learn policy
- **Advantages:**
  - No assumption that state-action pairs are i.i.d.
  - Transfer reward function to new environments

Feature Expectation Matching

- Assume that reward model $R(s,a)$ is a linear combination of some features $\phi_i(s,a)$:

$$R(s,a) = \sum_i w_i \phi_i(s,a) = \boldsymbol{w}^T \boldsymbol{\phi}(s,a)$$

- Value function:

$$\begin{aligned} V^\pi(s) &= \sum_t \gamma^t E_\pi[R(s_t,a_t)] \\ &= \sum_t \gamma^t E_\pi[\boldsymbol{w}^T \boldsymbol{\phi}(s_t,a_t)] \\ &= \boldsymbol{w}^T [\sum_t \gamma^t E_\pi[\boldsymbol{\phi}(s_t,a_t)]] \\ &= \boldsymbol{w}^T \bar{\boldsymbol{\phi}}^\pi \end{aligned}$$

Feature Expectation Matching

- Idea: find weights $\boldsymbol{w}$ that define a reward model R such that the optimal policy $\pi^{\boldsymbol{w}}$ (with respect to R based on $\boldsymbol{w}$) matches expert feature expectation.
  - Let $\overline{\boldsymbol{\phi}^e}$ be the feature expectation of expert e
  - Let $\pi^{\boldsymbol{w}}$ be an optimal policy for $R = \boldsymbol{w}^T \overline{\boldsymbol{\phi}^e}$
  - Find $\boldsymbol{w}$ such that $\overline{\boldsymbol{\phi}^e} = \overline{\boldsymbol{\phi}^{\pi^{\boldsymbol{w}}}}$

- Problem: infinitely many $\boldsymbol{w}$ satisfy the feature expectation matching equality

Maximum Margin IRL

Idea: find unique weights $\boldsymbol{w}$ that lead to the largest margin (value gap) possible between expert actions and non-expert actions.

- Let $\overline{\boldsymbol{\phi}^e}$ be the feature expectation of expert e
- Let π^w be an optimal policy for $R = \boldsymbol{w}^T \overline{\boldsymbol{\phi}^e}$
- Find $\boldsymbol{w}^* = \text{argmax}_{\boldsymbol{w}} \min_{\pi} \boldsymbol{w}^T (\overline{\boldsymbol{\phi}^e} - \overline{\boldsymbol{\phi}^\pi})$

Maximum Margin IRL Pseudocode

Input: expert trajectories $\tau^e \sim \pi^{expert}$ where $\tau^e = (s_1, a_1, s_2, a_2, \dots)$

Estimate $\overline{\boldsymbol{\phi}^e}$ from τ^e and learn transition model T from τ^e

Initialize policy π at random, sample $\tau \sim \pi$

Estimate $\bar{\boldsymbol{\phi}}$ from τ and initialize $\Phi = \{\bar{\boldsymbol{\phi}}\}$

Repeat

  Compute weights that maximize margin

$$\boldsymbol{w}^* = argmax_{\{\boldsymbol{w}: ||\boldsymbol{w}||_2 = 1\}} \; margin \;\; s.t. \;\; margin \leq \boldsymbol{w}^T(\overline{\boldsymbol{\phi}_e} - \bar{\boldsymbol{\phi}}) \quad \forall \boldsymbol{\phi} \in \Phi$$

  Compute optimal policy for w^* :

  $\pi^* = solveMDP(T, R, \gamma, h)$ where $R(s,a) = (\boldsymbol{w}^*)^T \bar{\boldsymbol{\phi}}$

  Sample $\tau \sim \pi^*$, estimate $\bar{\boldsymbol{\phi}}$ from τ and update $\Phi \leftarrow \Phi \cup \{\boldsymbol{\phi}^*\}$

Until $margin \leq \epsilon$

Return $\boldsymbol{w}^*$ and π^*

Further Reading (Some Seminal Papers)

- "Classic IRL"
  - Maximum Entropy Inverse Reinforcement Learning
  - Hierarchical Imitation and Reinforcement Learning

- "Deep IRL"
  - Guided cost learning: Deep inverse optimal control via policy optimization
  - Generative Adversarial Imitation Learning
  - Maximum entropy deep inverse reinforcement learning
  - One-Shot Imitation Learning
  - Learning Robust Rewards With Adversarial Inverse Reinforcement Learning

Combining Behavior Cloning and RL

Behavior Cloning and RL

- **Behavior cloning**
  - Pros
    - use supervised learning, simple and stable
  - Cons
    - needs demonstrations
    - covariate shift
    - can't outperform demonstrations
- **Reinforcement learning**
  - Pros
    - potential superhuman performance
  - Cons
    - needs reward function
    - needs to balance exploration and exploitation
    - potentially non-convergent

Behavior Cloning and RL

- If we have both demonstrations and rewards, can we get the best of both approaches?
- Idea: initialize with behavior cloning, then finetune with RL.

> - Step 1: collect demonstration data $\{(s, a)\}_N$
> - Step 2: apply behavior cloning on $\{(s, a)\}_N$ and learn $\pi_\theta(a|s)$.
> - Step 3: run $\pi_\theta(a|s)$ to collect experience
> - Step 4: improve $\pi_\theta(a|s)$ using RL

- Success example: "Learning to select and generalize striking movements in robot table tennis" by Mülling et al., 2013.

Off-policy RL

- The procedure introduced in the last slide represents on-policy learning.
- Off-policy RL can use data from arbitrary policies.
- Approach: use demonstrate data as off-policy samples, thus they are never forgotten.
- Available method: Q-learning, off-policy policy gradient, ...

Q-learning with Demonstrations

- Recall in DQN we have experience replay where we keep a buffer of experience collected so far and sample from it to update our policy.

- Approach: **feed expert demonstrations to the replay buffer and run the algorithm as usual.**

- Success example: “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards” by Vecerik et al., 2017.

Hybrid Loss Function

- Hybrid objective: Loss = behavior cloning objective + λ RL objective
- Similar to off-policy approaches, ensure the use of demonstrations.

- Examples
  - "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations" by Rajeswaran et al., 2017.
  - "Deep q-learning from demonstrations" by Hester et al., 2018.

- Issues: need to choose λ carefully and the algorithm becomes task-dependent especially in deciding which demonstrations to use.

Summary

- LfD is used when rewards are hard to define or RL's trial-and-error is unsafe/expensive.

- Behavioral Cloning (BC): Simply mimics expert. Suffers from Covariate Shift (compounding error), which can be fixed by DAgger (iterative expert labeling).

- Inverse RL (IRL): Recovers the expert's hidden reward function ("why"). Its main advantage over BC is Transfer Learning—the reward is general, the policy is not.