Imitation Learning

DAVIDE BACCIU – BACCIU@DI.UNIPI.IT

UNIVERSITÀ DI PISA

Outline

- ✓ Introduction
- ✓ Imitation learning challenges
  - ✓ Distribution mismatch
  - ✓ Sequential models
  - ✓ Multimodal actions
- ✓ Advanced topics
  - ✓ Generative imitation learning
  - ✓ Inverse reinforcement learning

Introduction

Limitations of learning by (physical) interaction

The agent should have the chance to try (and fail) MANY times

- ✓ Hard when safety is a concern
- ✓ Hard in general when each interaction takes time

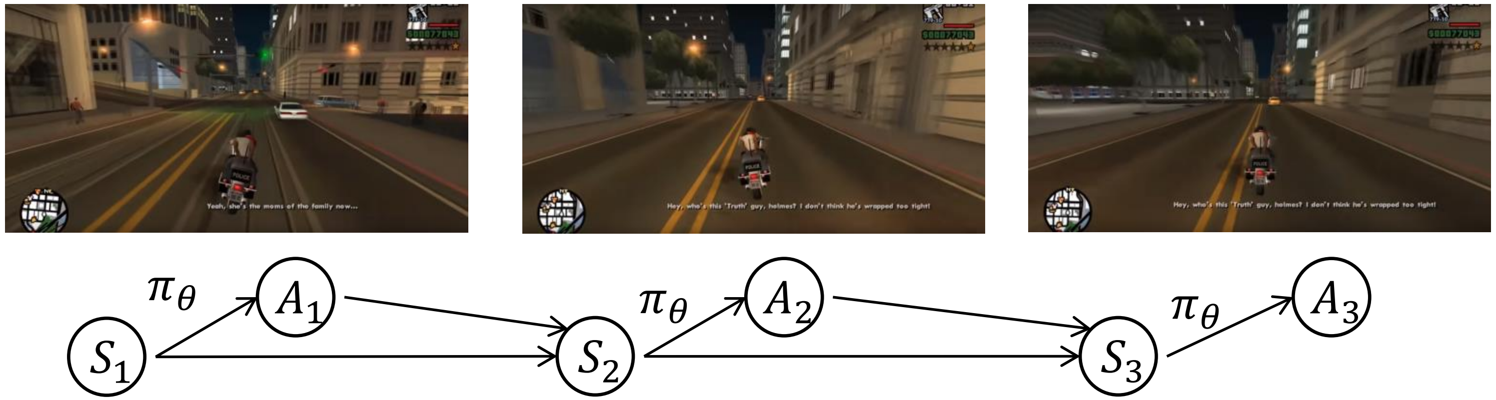
Imitation Learning

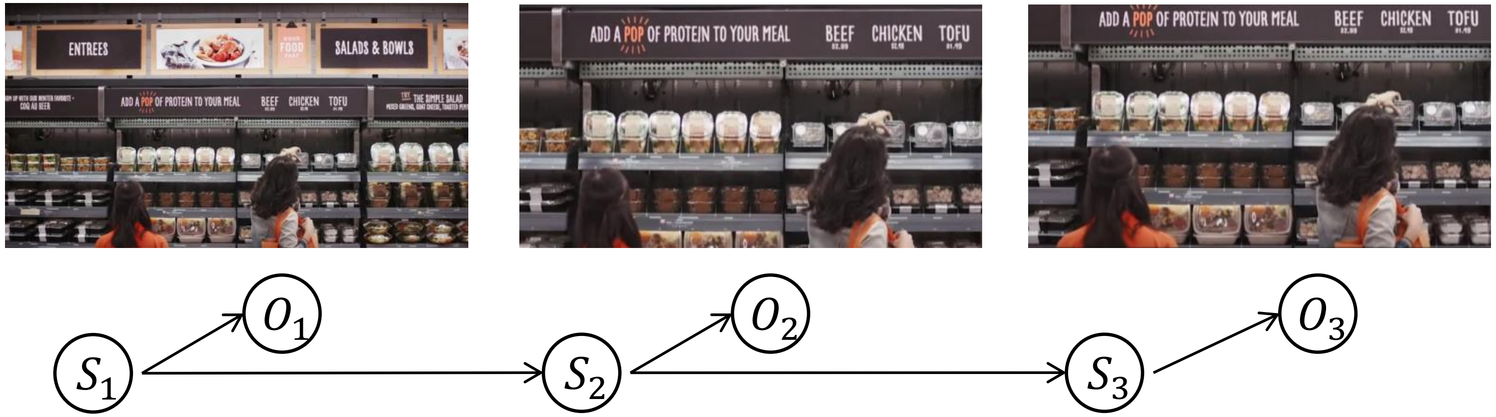
Learning from demonstrations

- ✓Kinesthetic imitation
  - ✓Teacher takes over the end effectors of the agent.
  - ✓Demonstrated actions are in the action space of the imitator

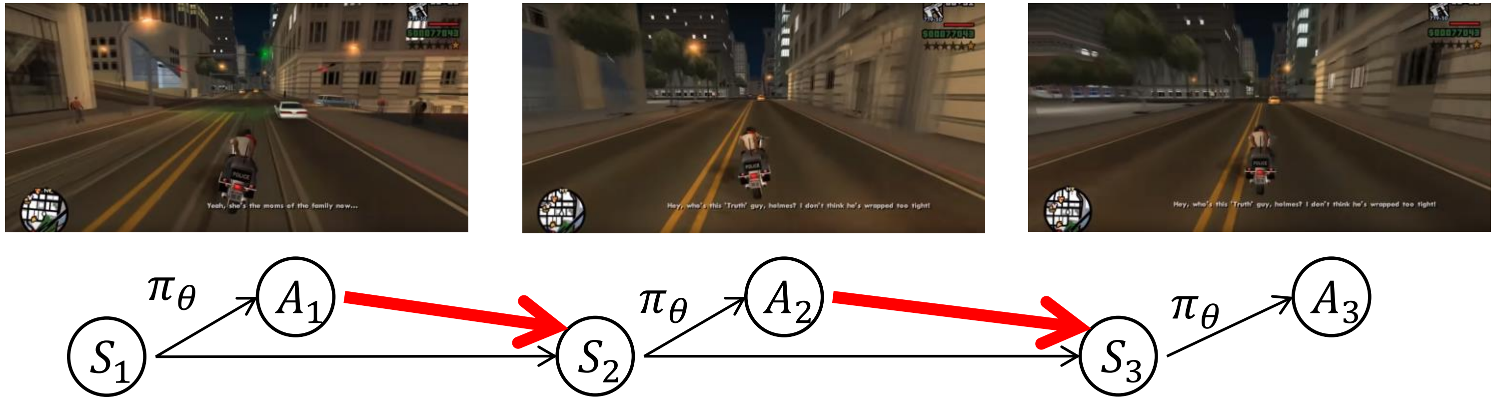
- ✓Visual imitation
  - ✓The actions of the teacher need to be inferred from visual sensory input and mapped to the action space of the agent

Imitation Learning Vs Supervised Labelling

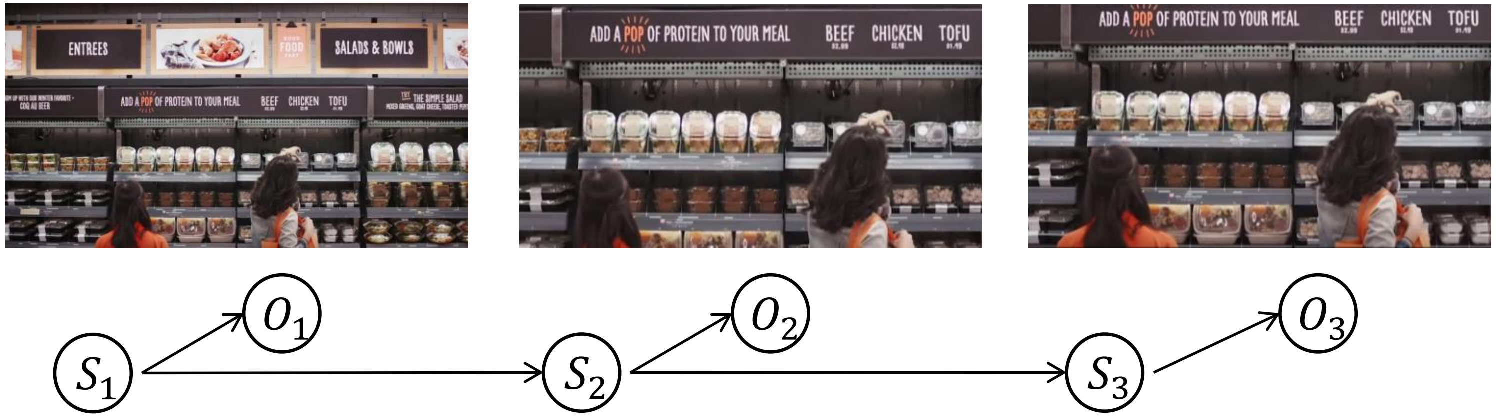




Imitation Learning Vs Supervised Labelling



Our actions influence future state and data

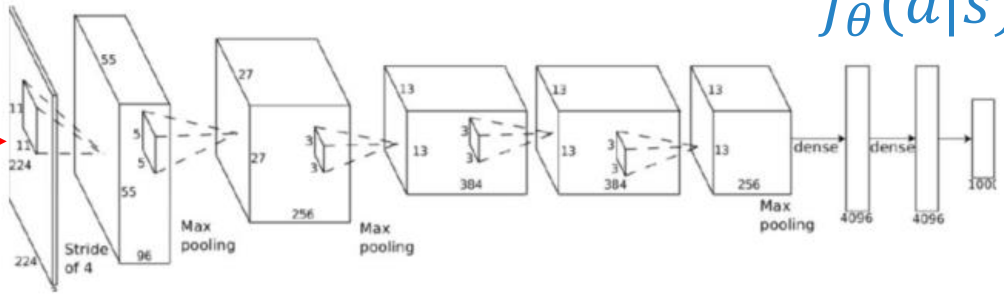


Predicted labels do not influence future

Action Labelling

A mapping from states/observations to action labels

$f_\theta(a|s)$



→ Buying poultry

Assume action labels in an annotated video are i.i.d

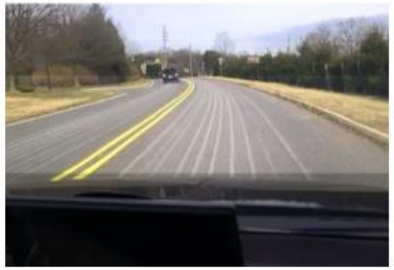
Train a classifier to map observations to labels at each time step

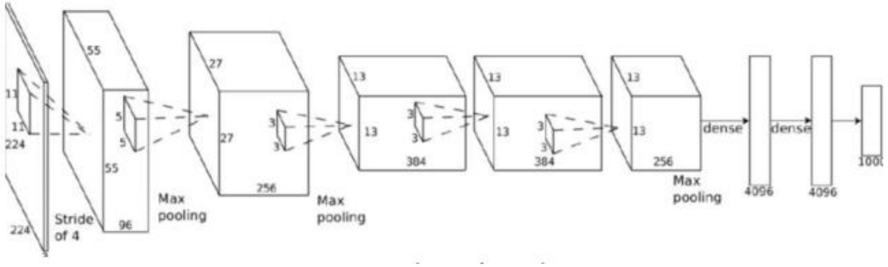
s_t a_t → Training Data → Supervised learning → $f_\theta(a|s)$

Imitation Learning (Behaviour Cloning)

Policy - A mapping from observations to actions

$\pi_\theta(a|s)$





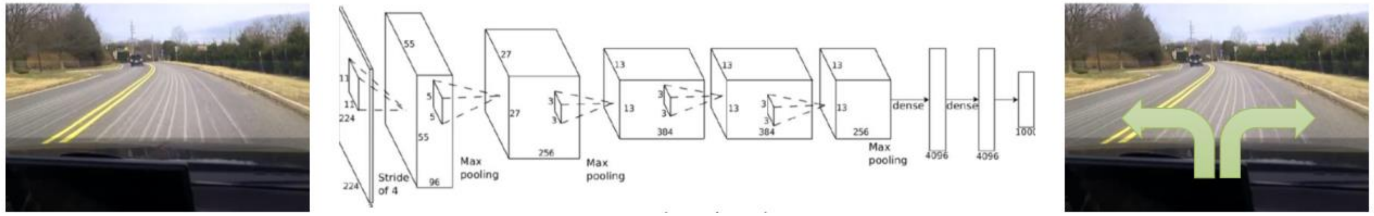
Assume actions in the expert trajectories are i.i.d

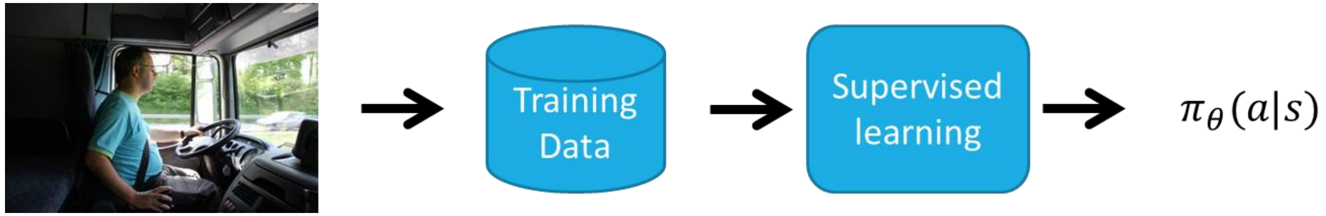
Train a function to map observations to actions at each time step

s_t, a_t → Training Data → Supervised learning → $\pi_\theta(a|s)$

What Possibly Can Go Wrong?

- ✓Compounding errors
  - ✓Data augmentation
- ✓Non-markovian observations
  - ✓Recurrent models
- ✓Stochastic expert actions
  - ✓Generative modelling

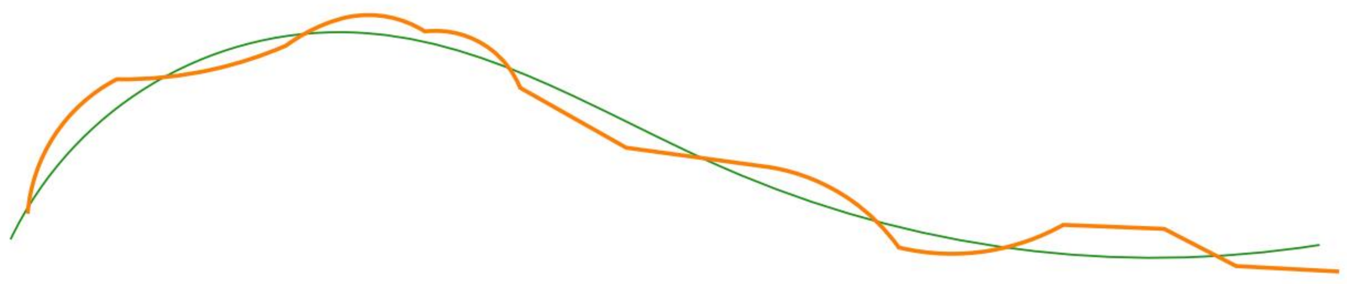




Distribution Shifts

Independent in Time Error

At each time step t, the agent wakes up on a state drawn from the state distribution of the expert trajectories, and executes an action

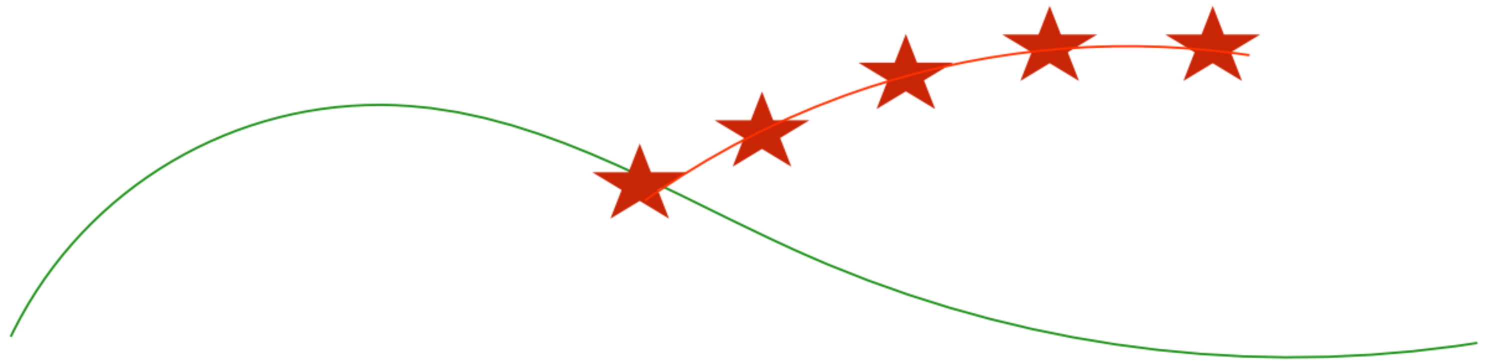


✓Error at each time t step bounded by ϵ

✓Expected total error for T steps: $\mathbb{E}[E] \leq \epsilon\, T$

Compounding Errors

At each time step t, the agent wakes up on a state drawn from the state distribution resulting from executing the action suggested by the learned policy previously

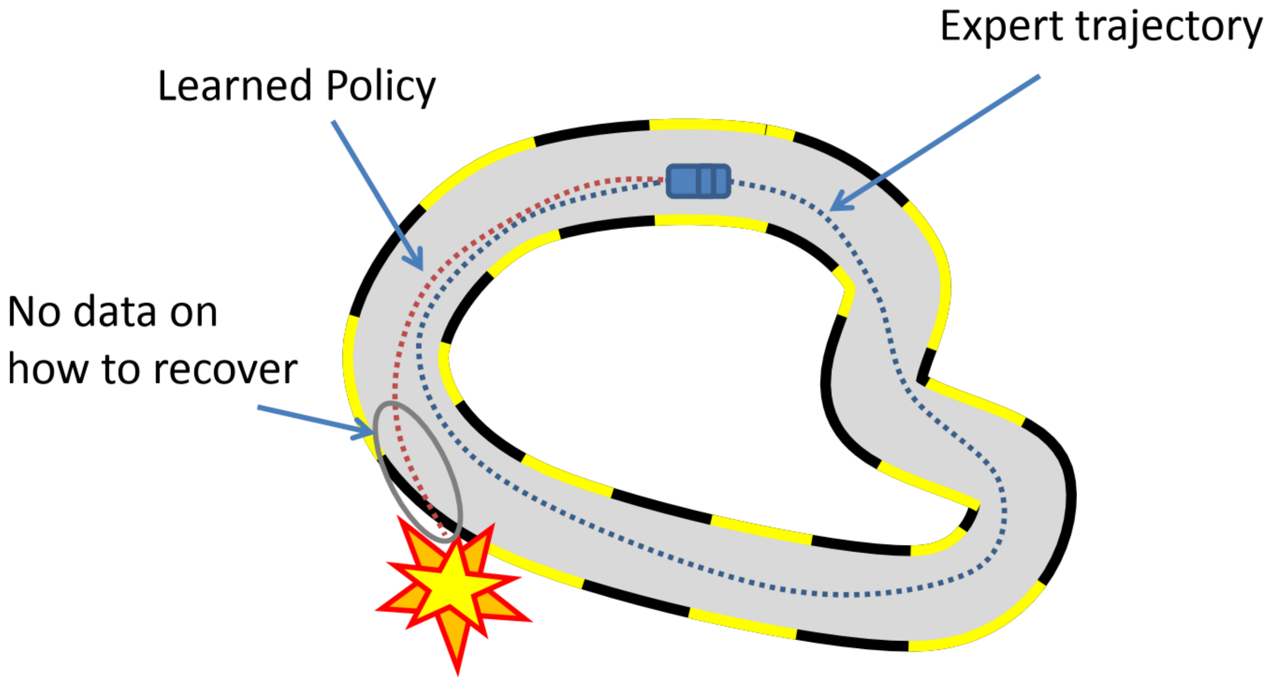


- ✓Error at each time t step bounded by ϵ
- ✓Expected total error for T steps: $\mathbb{E}[E] \leq \epsilon\,(T + (T-1) + (T-2) + \cdots) \propto \epsilon T^2$

Distribution Mismatch (Distribution Shift)

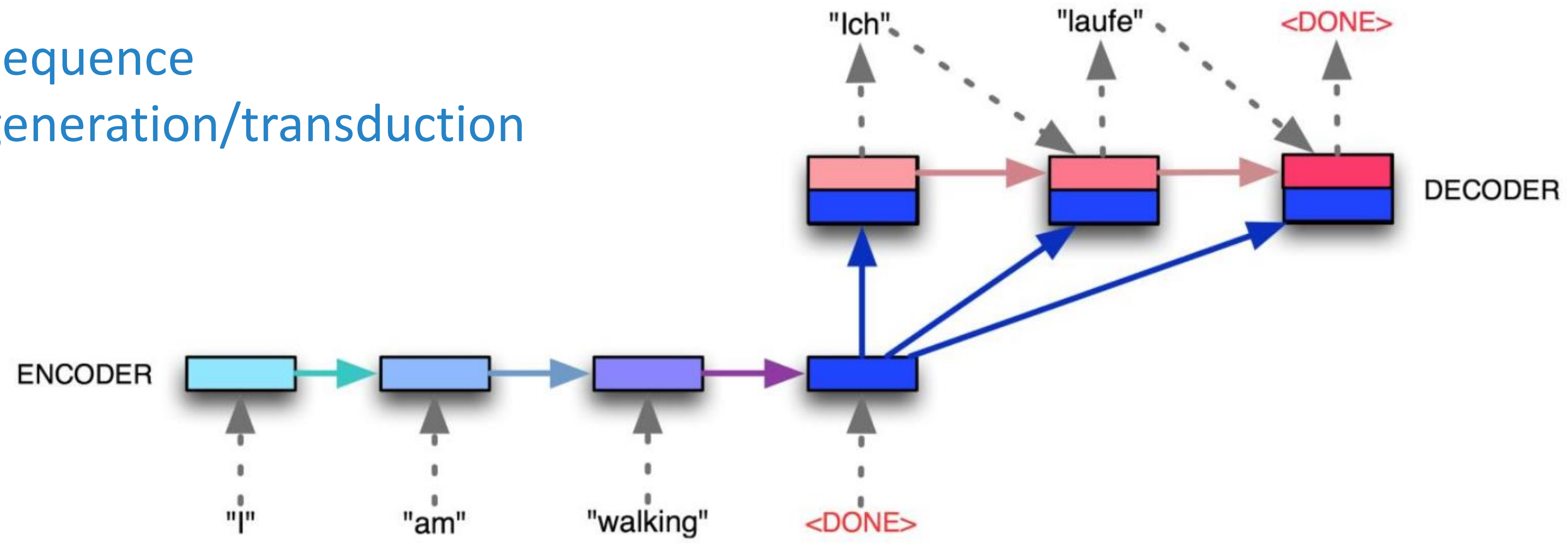
Due to the interdependence between our action at time step t and the state at $t + 1$, states seen at test time may come from a different distribution than those seen at training time

$$P_{\pi^*}(S_t) \neq P_{\pi_\theta}(S_t)$$



Something Similar Happens in..

Sequence generation/transduction



Solutions use teacher forcing with annealed schedule

Scheduled Sampling

- ✓Initial training phase
  - ✓Conditioning states come from the Teacher (training data)
- ✓Later training phase
  - ✓States/observations are sampled from the output of the model
- ✓Ground-truth for the next time step still from the expert
  - ✓Model learns to handle its mistakes
  - ✓Pushing deviating generated sequences back to the right track

Distribution Mismatch (DM)

	Supervised Learning	Supervised learning + Control
Train	$(x, y) \sim P(D)$	$s_t \sim P_{\pi^*}\ (\mathbf{s})$
Test	$(x, y) \sim P(D)$	$s_t \sim P_{\pi_\theta}\ (\mathbf{s})$

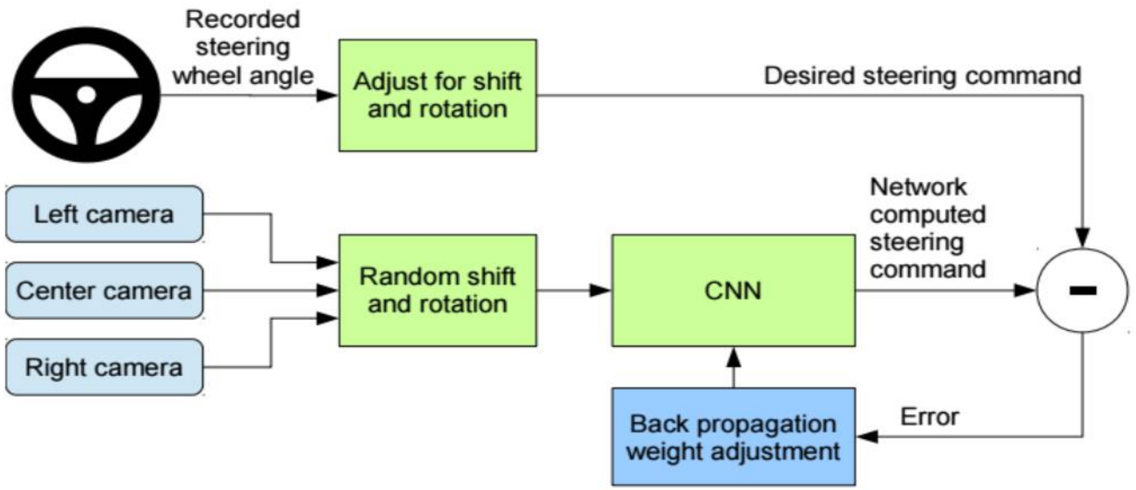
Fundamental assumption in (standard) supervised learning is that training and test data distributions match

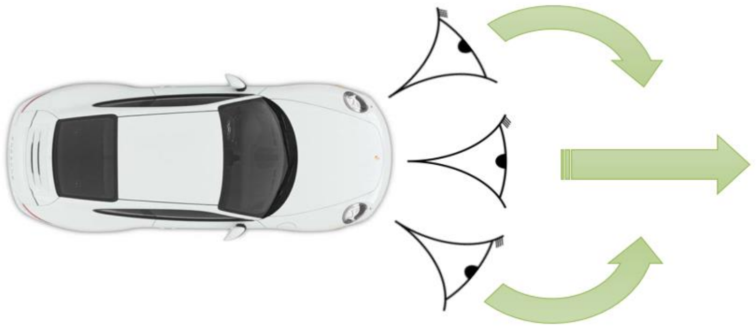
DM Solution - Augment Data

- ✓ Change $P_{\pi^*}(\boldsymbol{s})$ by augmenting the expert demonstration trajectories
  - ✓ Add examples in expert demonstration trajectories to cover the states where the agent will land when trying out its own policy
- ✓ Approaches
  - ✓ Generate synthetic data in simulation (ALVINN 1989)
  - ✓ Collecting additional data via clever tricks
  - ✓ Interactively query the experts in additional datapoints

Clever Tricks – NVIDIA 2016

End to End Learning for Self-Driving Cars , Bojarski et al. 2016





Additional, left and right cameras with automatic ground-truth labels to recover from mistakes

NVIDIA 2016 Demo

https://youtu.be/NJU9ULQUwng

Incremental Dataset Growing - DAgger

How can we make $P_{\pi^*}\,(\boldsymbol{s}) \approx P_{\pi_\theta}\,(\boldsymbol{s})$?

Key Idea - If we cannot be clever on $P_{\pi_\theta}\,(\boldsymbol{s})$ let us be clever on $P_{\pi^*}\,(\boldsymbol{s})$

DAgger – Dataset Aggregation

✓ Collect training data from $P_{\pi_\theta}\,(\boldsymbol{s})$ in place of $P_{\pi^*}\,(\boldsymbol{s})$

✓ Collect observations by running $\pi_\theta(a_t|s_t)$ and ask someone for labels a_t

Incremental Dataset Growing - DAgger

How can we make $P_{\pi^*}(\boldsymbol{s}) \approx P_{\pi_\theta}(\boldsymbol{s})$?

Key Idea - If we cannot be clever on $P_{\pi_\theta}(\boldsymbol{s})$ let us be clever on $P_{\pi^*}(\boldsymbol{s})$

DAgger – Dataset Aggregation

1. Train $\pi_\theta(a_t|s_t)$ from expert data $\mathcal{D} = \{\boldsymbol{s}_1, \boldsymbol{a}_1, \dots, \boldsymbol{s}_N, \boldsymbol{a}_N\}$
2. Run $\pi_\theta(a_t|s_t)$ to get data $\mathcal{D}_\pi = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_N\}$
3. Ask expert to label $\mathcal{D}_\pi$ with action $\boldsymbol{a}_n$
4. Aggregate $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Potential issues

- Execute an unsafe/partially trained policy
- Repeatedly query the expert
- Expert is queried for an action without experiencing the state

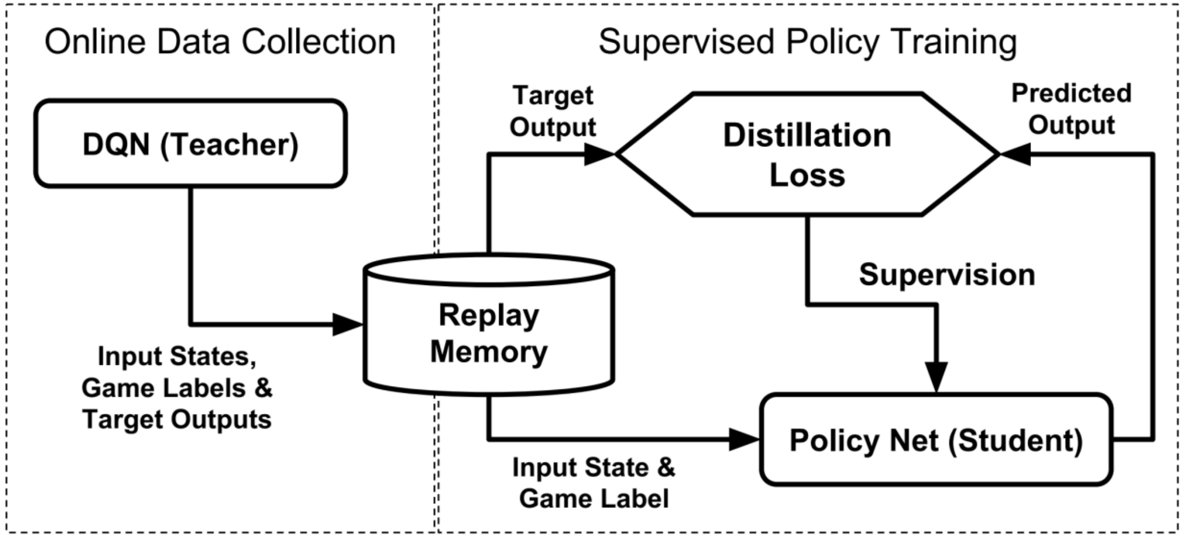
Dagger Demo

Ross et al, Learning monocular reactive UAV control in cluttered natural environments, 2013

https://youtu.be/hNsP6-K3Hn4

Beyond Vanilla Dagger

- ✓Experts do not need to be humans
  - ✓Generative learning can be used for imitating expert policies
  - ✓Solving simpler optimization in a constrained part of the state space
- ✓Imitation then means distilling knowledge of constrained policies into a general policy that can do well in all scenarios



Rusu, Policy distillation, ICLR 2016

Meeting the Expert Expectations

Non-Markovian Behaviour

$$\pi_\theta(a_t|s_t)$$

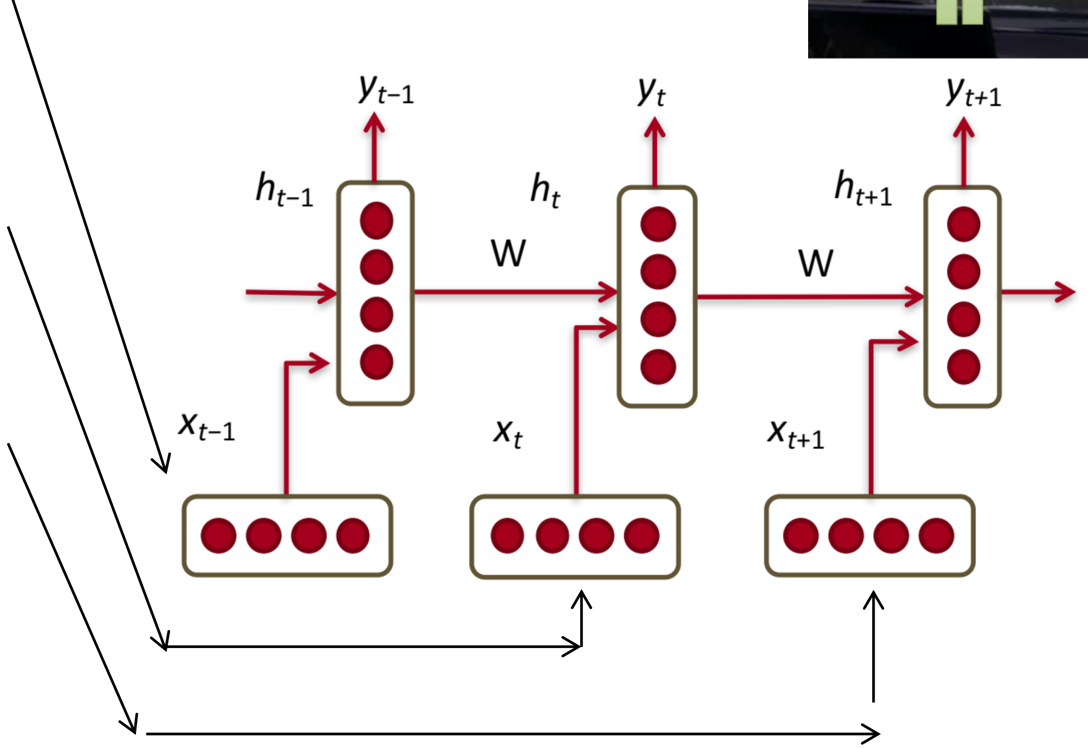
If we see the same thing twice, we do the same thing twice, regardless of what happened before

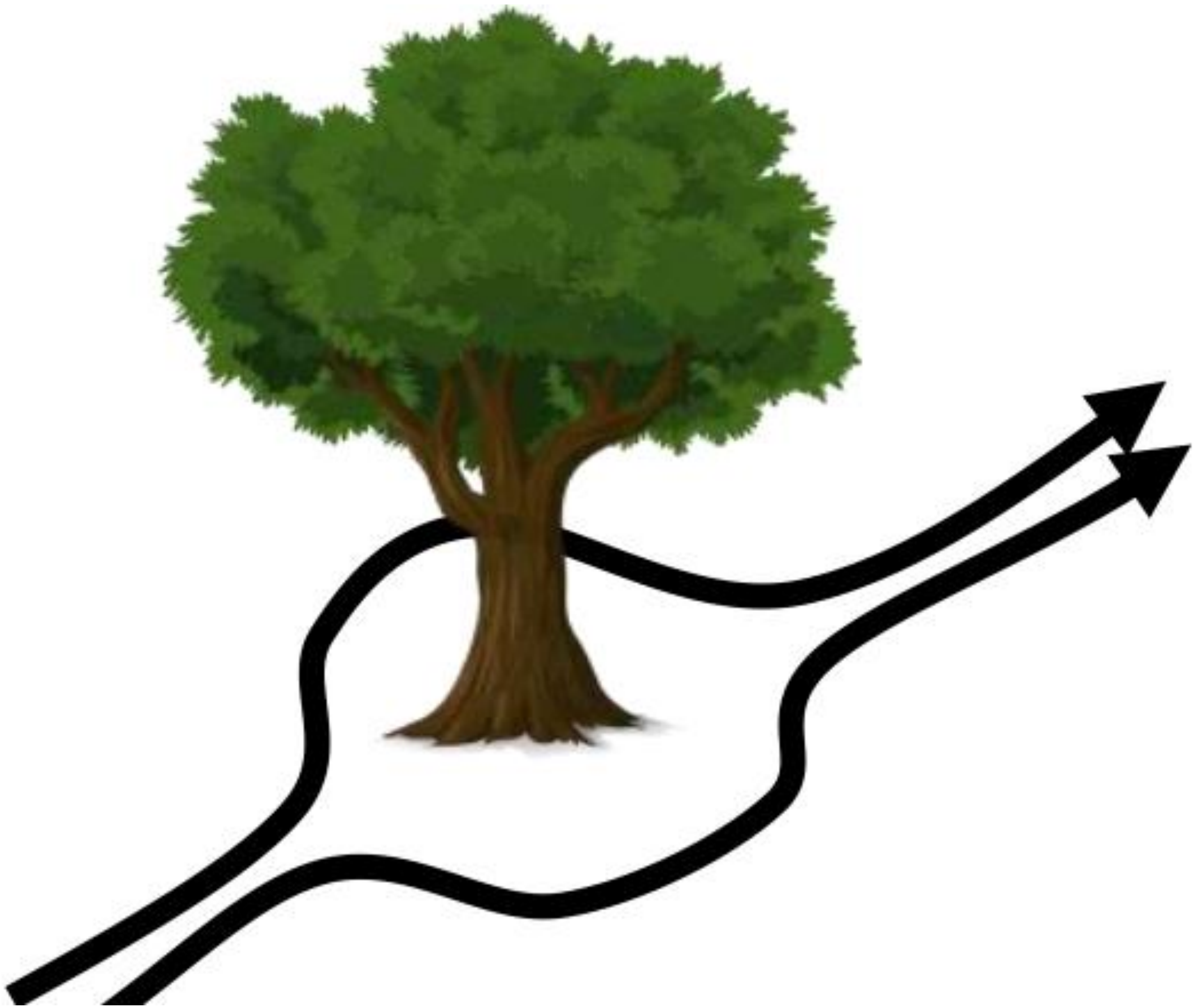
$$\pi_\theta(a_t|s_t, s_{t-1}, \ldots, s_0)$$

Behavior depends on the history of past observations

Non-Markovian Behaviour – How to?

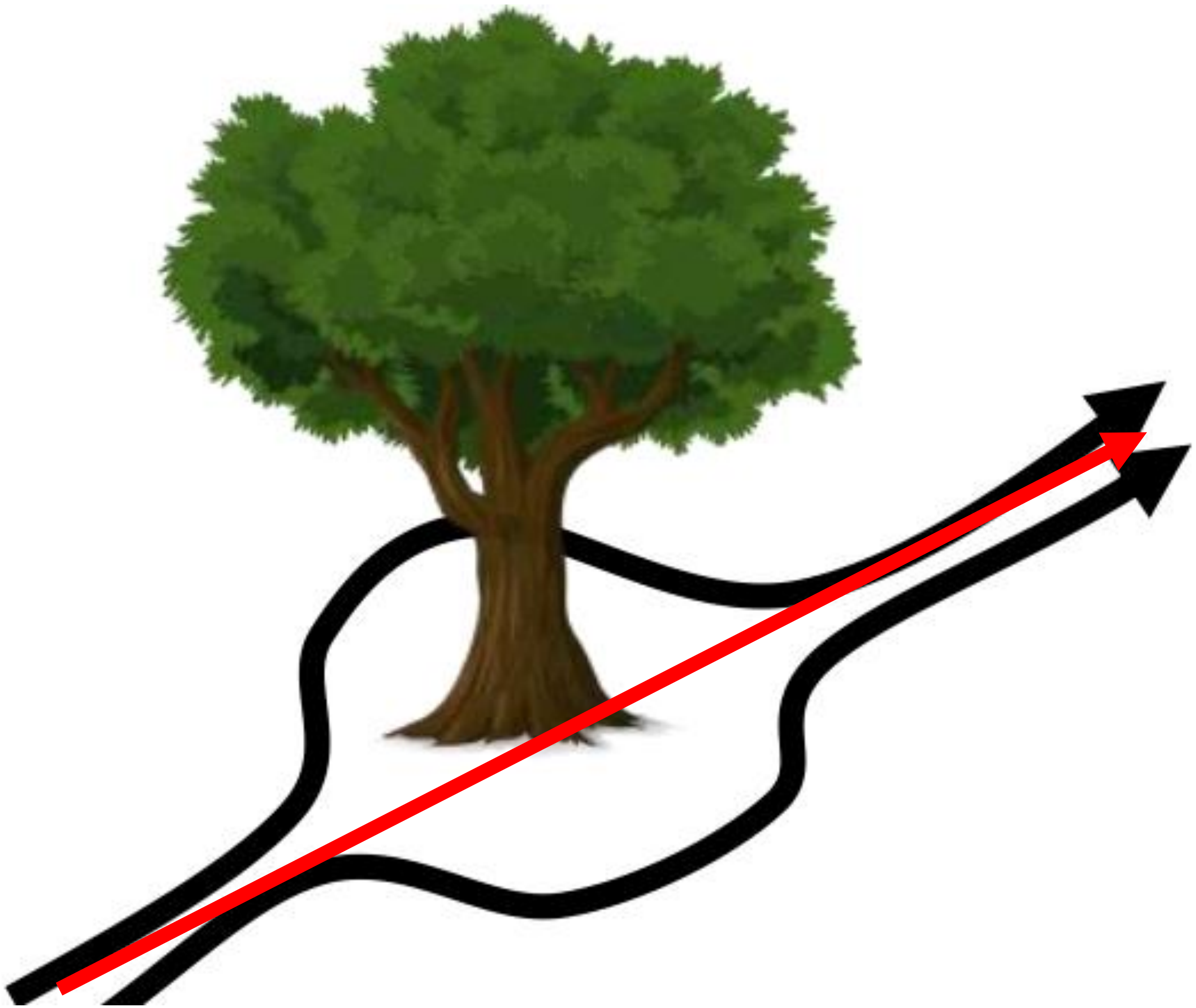
Recurrent neural network
(gated to say the least)





Multimodal Behaviour

- ✓Avoiding an obstacle for an expert is easy
- ✓Take one of the two steering angles



Multimodal Behaviour in Regression

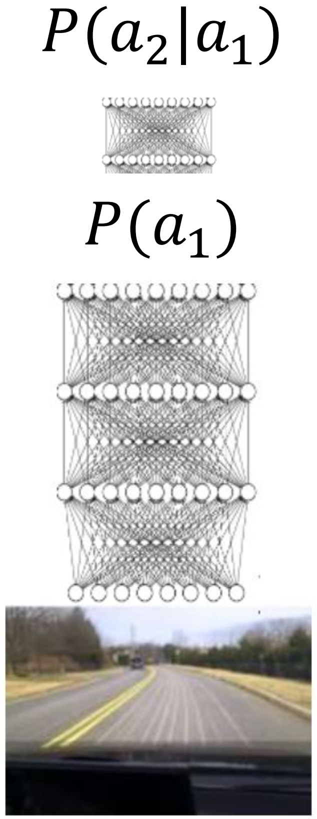
- ✓Regression fails in multimodality
- ✓MSE minimum is the average
- ✓Which might not be advisable…

Multimodal Behaviour in Regression

- ✓ ….unless you know how to do this

Multimodality - Can we fix it?

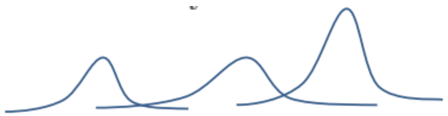
- Autoregressive discretization
  - Discretize the action space and train a classifier that predicts a categorical distribution it
  - Do it progressively on the original features to avoid curse of dimensionality (e.g. PixelRNN)
- Gaussian mixture model output
  - Predict mixture components weights, means and variances
  - Need to guess number of modes
- Density model
  - Density networks
  - Variational Autoencoders
  - Generative Adversarial Network

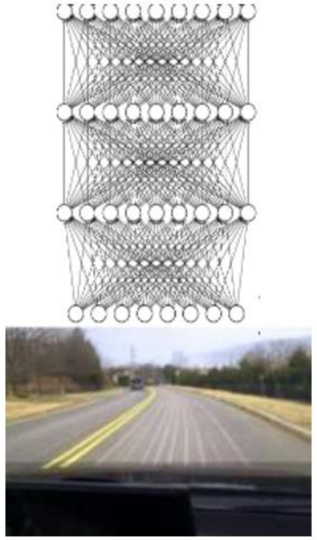


Multimodality - Can we fix it?

- Autoregressive discretization
  - Discretize the action space and train a classifier that predicts a categorical distribution it
  - Do it progressively on the original features to avoid curse of dimensionality (e.g. PixelRNN)
- Gaussian mixture model output
  - Predict mixture components weights, means and variances
  - Need to guess number of modes
- Density model
  - Density networks
  - Variational Autoencoders
  - Generative Adversarial Network

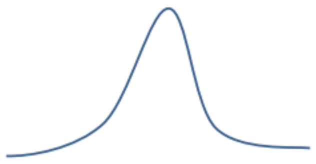
$$\pi_\theta(a|s) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$$



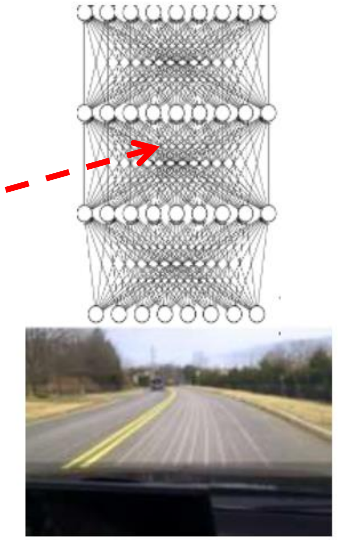


Multimodality - Can we fix it?

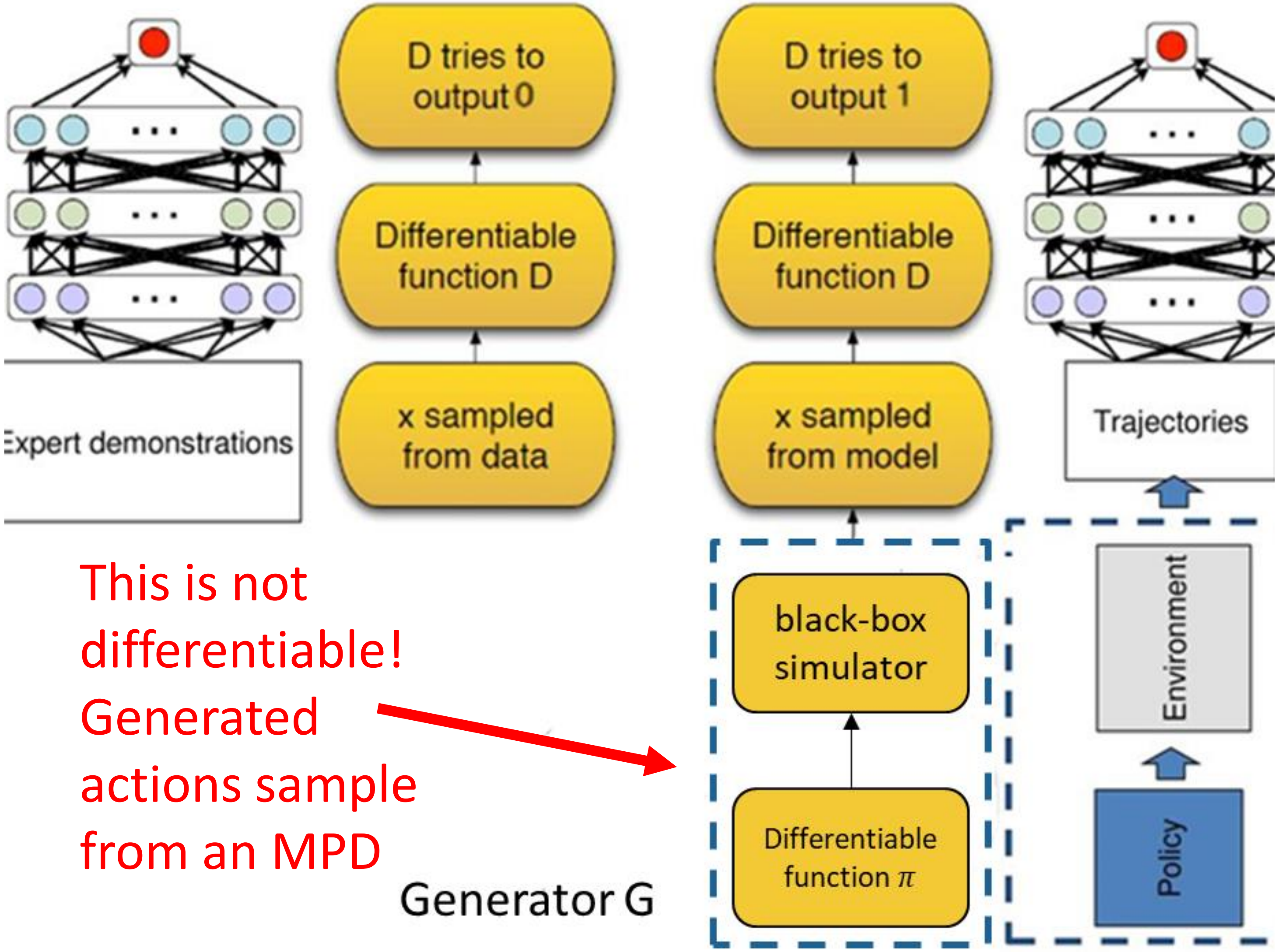
- Autoregressive discretization
  - Discretize the action space and train a classifier that predicts a categorical distribution it
  - Do it progressively on the original features to avoid curse of dimensionality (e.g. PixelRNN)
- Gaussian mixture model output
  - Predict mixture components weights, means and variances
  - Need to guess number of modes
- Density model
  - Density networks
  - Variational Autoencoders
  - Generative Adversarial Network



$\epsilon \sim \mathcal{N}(0, \sigma)$



Generative Imitation Learning



This is not differentiable! Generated actions sample from an MPD

Generative Adversarial Imitation Learning (GAIL)

- ✓Use a policy network as generator (state conditioned)
- ✓Find a policy that makes it impossible for a discriminator network to distinguish between state-actions from the expert demonstrations and state-actions visited by the learnt policy
- ✓Generator needs to be trained using RL

Ho and Ermon, Generative Adversarial Imitation Learning, NIPS 2016

GAIL Algorithm

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0

2: **for** $i = 0, 1, 2, \ldots$ **do**

3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$

4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \quad (17)$$

Discriminator provides reward function

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta),$$
$$\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \quad (18)$$

Trust-Region policy gradient

Entropy-based policy regularizer

6: **end for**

Rewards are not always as explicit

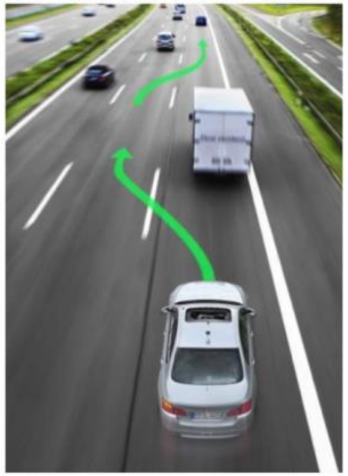
reward

Mnih et al. '15

robotics

dialog

autonomous driving



what is the reward?
often use a proxy

Inverse Reinforcement Learning (IRL)

- ✓ An alternative to imitation learning
  - ✓ Use demonstrations to learn a reward function
  - ✓ Train a policy using learnt reward function
- ✓ Least expensive form of supervision
  - ✓ Does not need full demonstrations
  - ✓ RL phase can “fill in” missing behavior given partial demonstrations
- ✓ Argued to be a more comprehensive model of expert behavior
  - ✓ Learning why the expert did something instead of mapping states to actions
  - ✓ Can potentially generalize better

GAIL is a particular form of inverse RL that learns a reward function that tries to match state distributions between the expert and imitator

More Formally

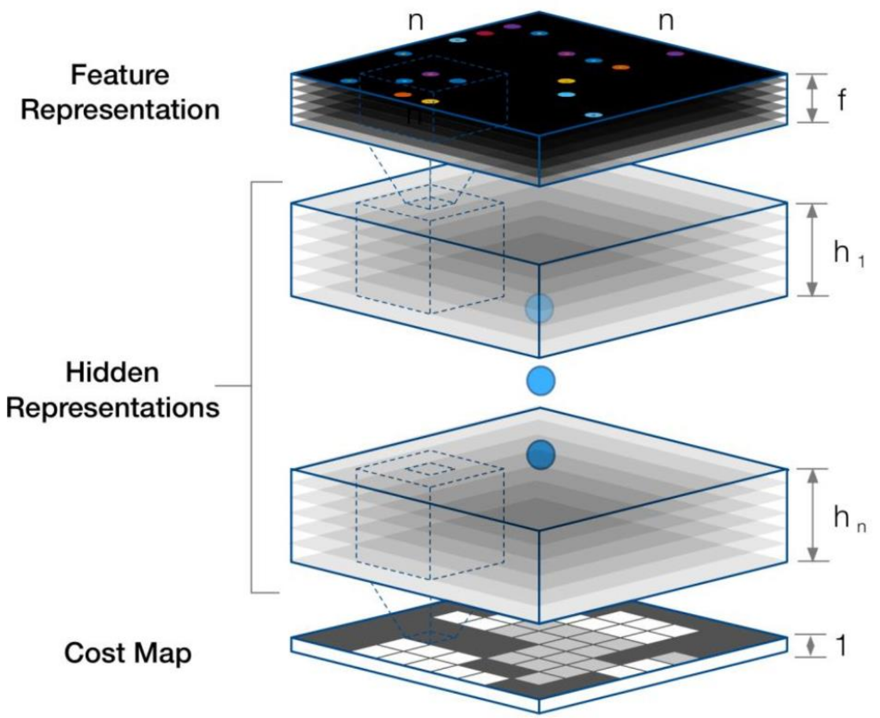
Forward Reinforcement Learning

- ✓ Given
  - ✓ States $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$
  - ✓ Transitions $P_{ss'}^{a}$, (sometimes)
  - ✓ Reward function $\mathcal{R}_{s}^{a}$
- ✓ Learn or infer policy $\pi^{*}(a|s)$

Inverse Reinforcement Learning

- ✓ Given
  - ✓ States $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$
  - ✓ Transitions $P_{ss'}^{a}$, (sometimes)
  - ✓ Sampled episodes from expert $(\mathrm{s}, \mathrm{a}) \sim \pi^{*}(a|s)$
- ✓ Learn a reward function $\mathrm{r}_{\phi}(\mathrm{a}, \mathrm{s})$, with ϕ being adaptive parameters
- ✓ ...and use $\mathrm{r}_{\phi}(\mathrm{a}, \mathrm{s})$ to learn/infer $\pi^{*}(a|s)$

Solving IRL – MaxEntropy Deep IRL



Wulfmeier et al, Maximum Entropy Deep Inverse Reinforcement Learning, 2015

Algorithm 1 Maximum Entropy Deep IRL

Input: $\mu_D^a, f, S, A, T, \gamma$

Output: optimal weights θ^*

$\theta^1 = \text{initialise\_weights}()$

Iterative model refinement

for n = 1 : N **do**

$r^n = \text{nn\_forward}(f, \theta^n)$

Solution of MDP with current reward

$\pi^n = \text{approx\_value\_iteration}(r^n, S, A, T, \gamma)$

$\mathbb{E}[\mu^n] = \text{propagate\_policy}(\pi^n, S, A, T)$

Determine Maximum Entropy loss and gradients

$\mathcal{L}_D^n = \log(\pi^n) \times \mu_D^a$

$\frac{\partial \mathcal{L}_D^n}{\partial r^n} = \mu_D - \mathbb{E}[\mu^n]$

Compute network gradients

$\frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n} = \text{nn\_backprop}(f, \theta^n, \frac{\partial \mathcal{L}_D^n}{\partial r^n})$

$\theta^{n+1} = \text{update\_weights}(\theta^n, \frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n})$

end for

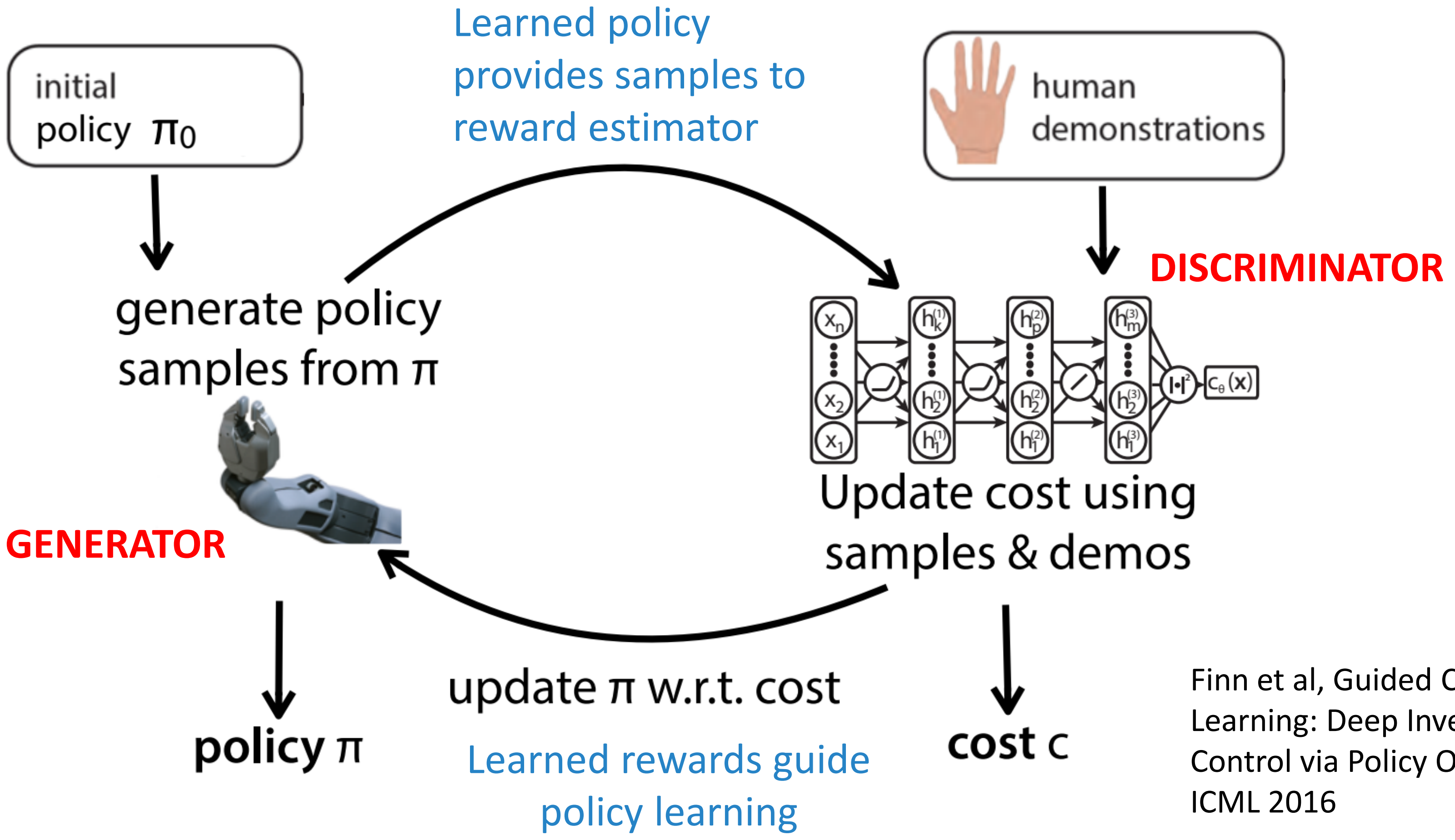
Expert state-action frequencies

Finds Q and V and infers π^n

Expected state visiting frequencies by sampling with π^n

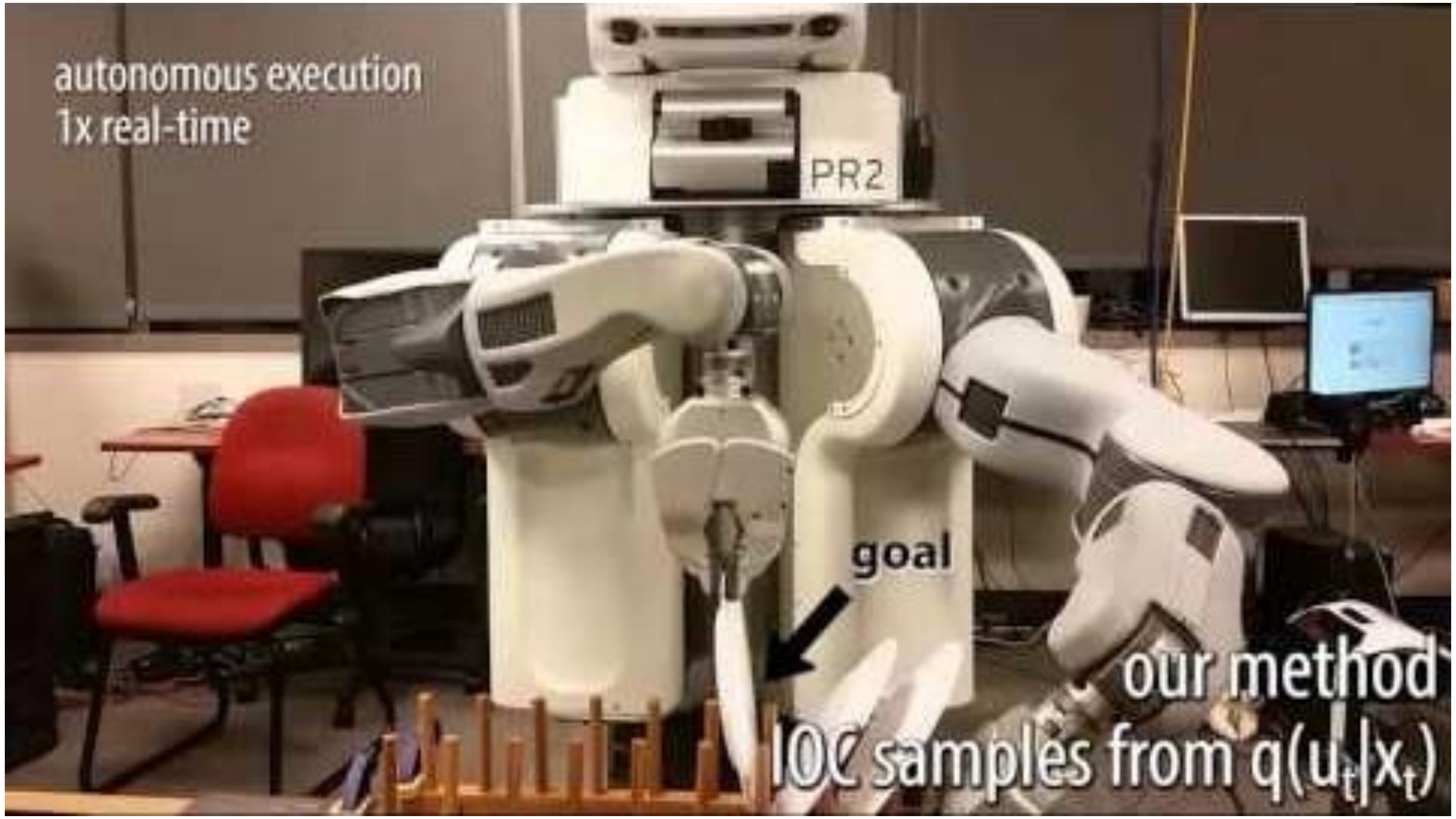
MAP of observing expert samples

Guided Cost Learning



Finn et al, Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization, ICML 2016

GCL Demo



https://youtu.be/hXxaepw0zAw

Wrap-up

Take (stay) home messages

- ✓Effective imitation learning is much about managing distribution shift and relaying less on human demonstration
- ✓Using a learnable model of reward can serve the purpose of reducing the extent of human labelling (inverse reinforcement learning)
- ✓Much left unsaid
  - ✓Off-policy learning from imitation policy
  - ✓Q-learning as natural off-policy imitation learner
    - ✓Just drop demonstrations into the replay buffer
  - ✓Inverse reinforcement learning Vs generative adversarial learning

Coming up

Final lecture in 2 weeks (sorry!)

- Friday 03rd July h. 16-18

About

- Course wrap-up
- Research topics and interesting applications
- Planning exams