Imitation Learning

ICML 2018 Tutorial

(Slides Available Online)

Yisong Yue

yyue@caltech.edu

@YisongYue

yisongyue.com

Hoang M. Le

hmle@caltech.edu

@HoangMinhLe

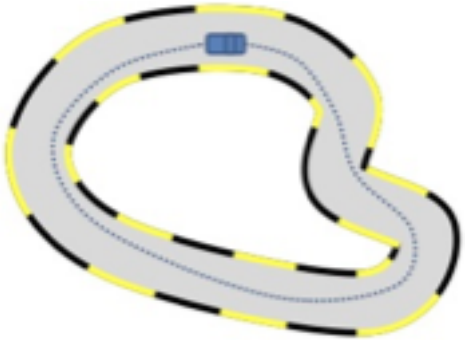
hoangle.info

Imitation Learning in a Nutshell

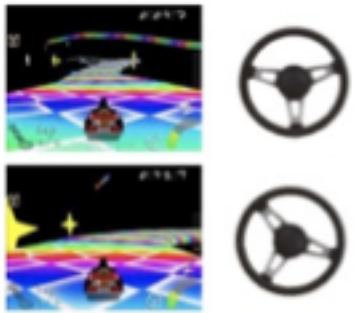
Given: demonstrations or demonstrator

Goal: train a policy to mimic demonstrations

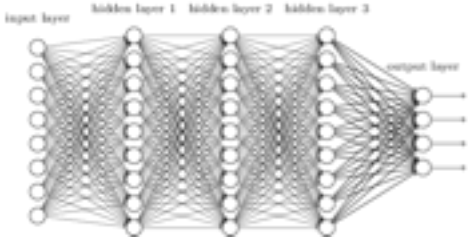
Expert Demonstrations



State/Action Pairs

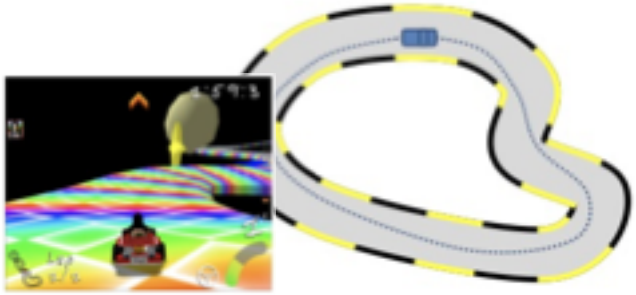


Learning



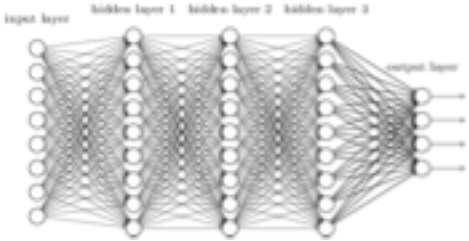
Images from Stephane Ross

Ingredients of Imitation Learning



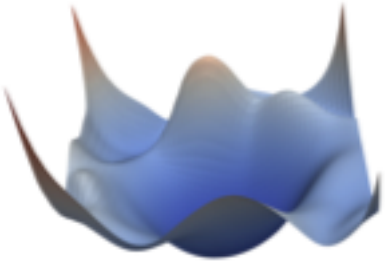
Demonstrations or Demonstrator

Environment / Simulator



Policy Class

Loss Function



Learning Algorithm

Tutorial Overview

Part 1: Introduction and Core Algorithms

- Teaser results
- Overview of ML landscape
- Types of Imitation learning
- Core algorithms of passive, interactive learning and cost learning

Part 2: Extensions and Applications

- Speech Animation
- Structured prediction and search
- Improving over expert
- Filtering and sequence modeling
- Multi-objective imitation learning
- Visual / few-shot imitation learning
- Domain Adaptation imitation learning
- Multi-agent imitation learning
- Hierarchical imitation learning
- Multi-modal imitation learning
- Weaker Feedback

Notation & Setup

State: s (sometimes x) (**state may only be partially observed)

Action: a (sometimes y)

Policy: $\pi\theta$ (sometimes h)

- Policy maps states to actions: $\pi\theta(s) \rightarrow a$
- ...or distributions over actions: $\pi\theta(s) \rightarrow P(a)$

State Dynamics: P(s'|s,a)

- Typically not known to policy.
- Essentially the simulator/environment

Notation & Setup Continued

Rollout: sequentially execute $\pi(s_0)$ on an initial state

- Produce trajectory $\boldsymbol{\tau}=(s_0,a_0,s_1,a_1,\ldots)$

$P(\boldsymbol{\tau}|\pi)$: distribution of trajectories induced by a policy

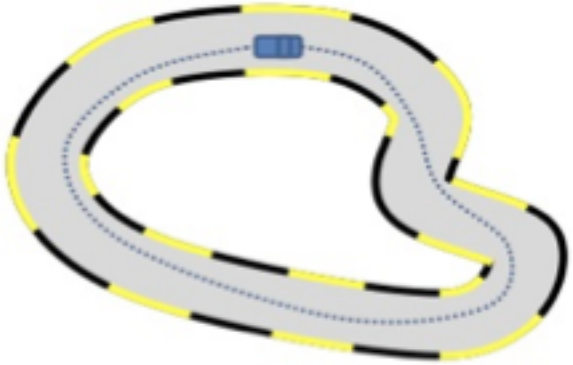
1. Sample s_0 from P_0 (distribution over initial states), initialize t = 1.
2. Sample action a_i from $\pi(s_{t-1})$
3. Sample next state s_t from applying a_t to s_{t-1} (requires access to environment)
4. Repeat from Step 2 with t=t+1

$P(s|\pi)$: distribution of states induced by a policy

- Let $P_t(s|\pi)$ denote distribution over t-th state
- $P(s|\pi) = (1/T)\sum_t P_t(s|\pi)$

Example #1: Racing Game

(Super Tux Kart)



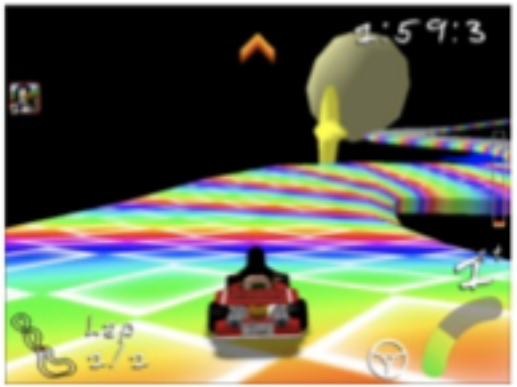
s = game screen

a = turning angle

Training set: D={$\boldsymbol{\tau}$:=(**s**,**a**)} from π^*

- **s** = sequence of s
- **a** = sequence of a

Goal: learn $\pi_\theta(s) \rightarrow a$



Images from Stephane Ross

Example #2: Basketball Trajectories

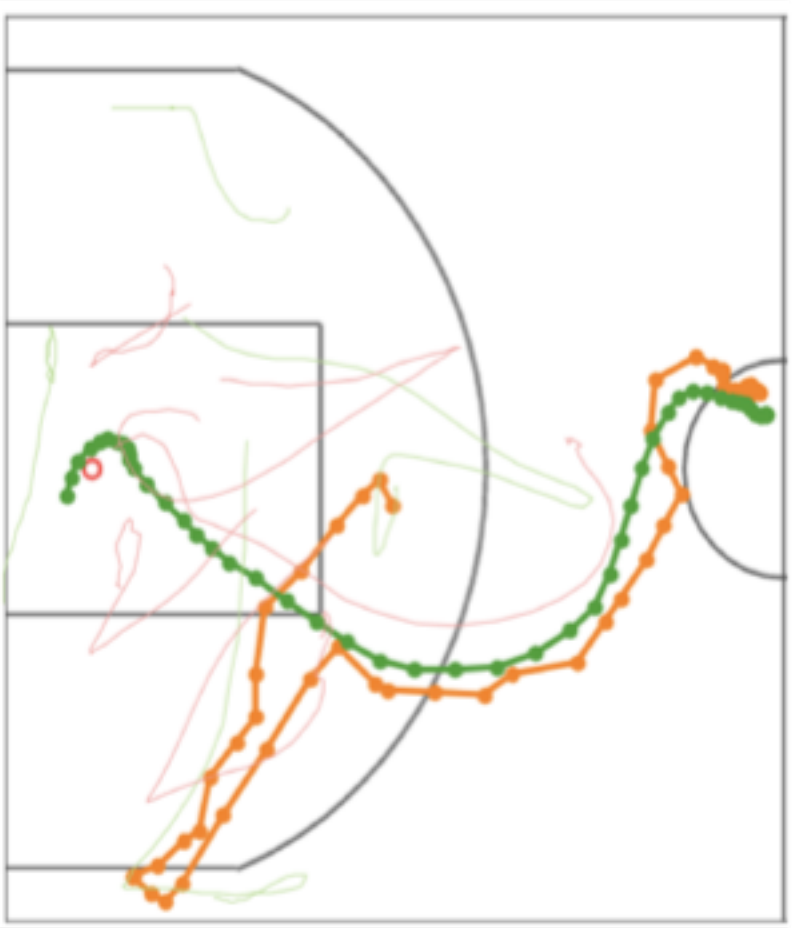
s = location of players & ball

a = next location of player

Training set: D={$\boldsymbol{\tau}$:=(**s**,**a**)} from π^*

- **s** = sequence of s
- **a** = sequence of a

Goal: learn $\pi_\theta(s) \rightarrow a$



Outline of 1st Half

Behavioral Cloning (simplest Imitation Learning setting)

Compare with Supervised Learning

Landscape of Imitation Learning settings

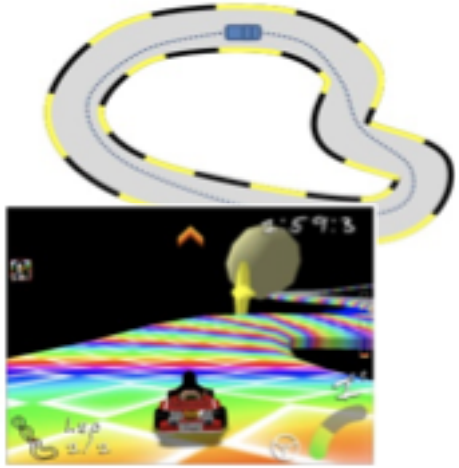
Behavioral Cloning = Reduction to Supervised Learning

(Ignoring regularization for brevity.)

Define $P^* = P(s|\pi^*)$ (distribution of states visited by expert)

(recall $P(s|\pi^*) = (1/T)\sum_t P_t(s|\pi^*)$)

(sometimes abuse notation: $P^* = P(s,a^*=\pi^*(s)|\pi^*)$)



Learning objective:

$$\text{argmin}_\theta \; E_{(s,a^*)\sim P^*} L(a^*, \pi_\theta(s))$$

Interpretations:

1. Assuming perfect imitation so far, learn to continue imitating perfectly
2. Minimize 1-step deviation error along the expert trajectories

Images from Stephane Ross

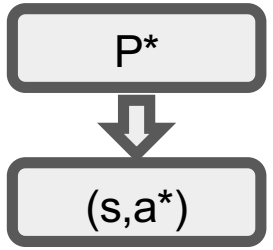
(General) Imitation Learning vs Behavioral Cloning

(Ignoring regularization for brevity.)

Behavioral Cloning (Supervised Learning):

$$\text{argmin}_\theta \, E_{(s,a^*)\sim P^*} L(a^*, \pi_\theta(s))$$

Distribution provided exogenously

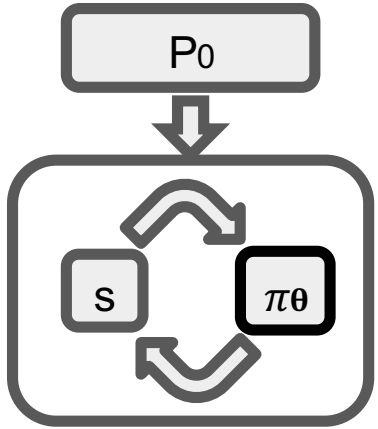


Training Loss

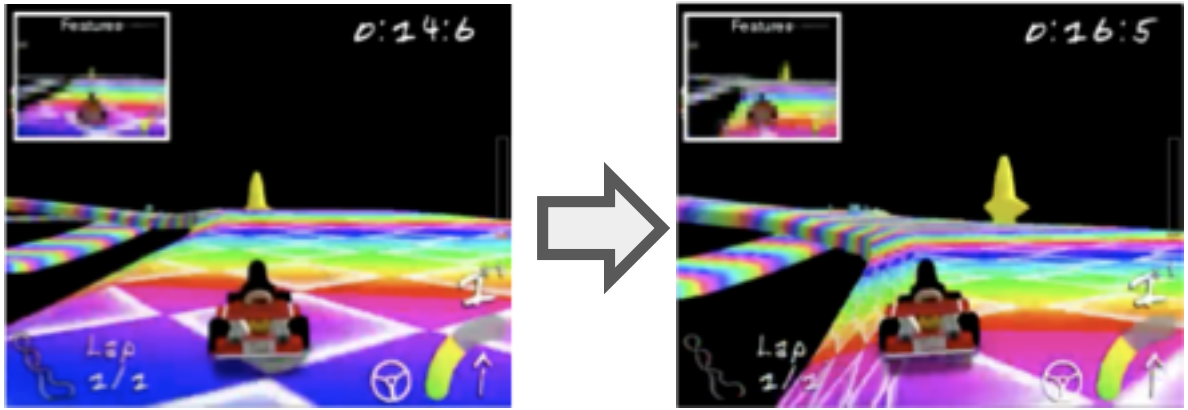
(General) Imitation Learning:

$$\text{argmin}_\theta \, E_{s\sim P(s|\theta)} L(\pi^*(s), \pi_\theta(s))$$

Distribution depends on rollout.
$P(s|\theta)$ = state distribution of π_θ



Limitations of Behavioral Cloning



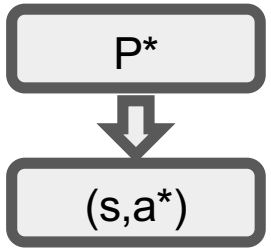
π_θ makes a mistake

New state sampled not from P*!

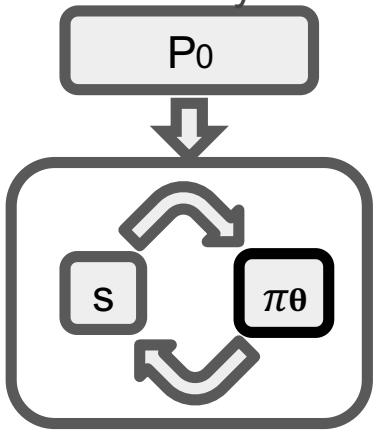
Worst case is catastrophic!

Images from Stephane Ross

IID Assumption
(Supervised Learning)



Reality



Limitations of Behavioral Cloning

Expert Trajectories
(Training Distribution)

Behavioral Cloning

Makes mistakes, enters new states

Cannot recover from new states

Videos from Stephane Ross

When to use Behavioral Cloning?

Advantages

- Simple
- Simple
- Efficient

Disadvantages

- Distribution mismatch between training and testing
- No long term planning

Use When:

- 1-step deviations not too bad
- Learning reactive behaviors
- Expert trajectories “cover” state space

Don’t Use When:

- 1-step deviations can lead to catastrophic error
- Optimizing long-term objective (at least not without a stronger model)

Outline of 1st Half

Behavioral Cloning (simplest Imitation Learning setting)

Compare with Supervised Learning

Landscape of Imitation Learning settings

Types of Imitation Learning

Behavioral Cloning

$$\text{argmin}_\theta \, E_{(s,a^*)\sim P^*} L(a^*, \pi_\theta(s))$$

Works well when P* close to Pθ

Inverse RL

Learn r such that:

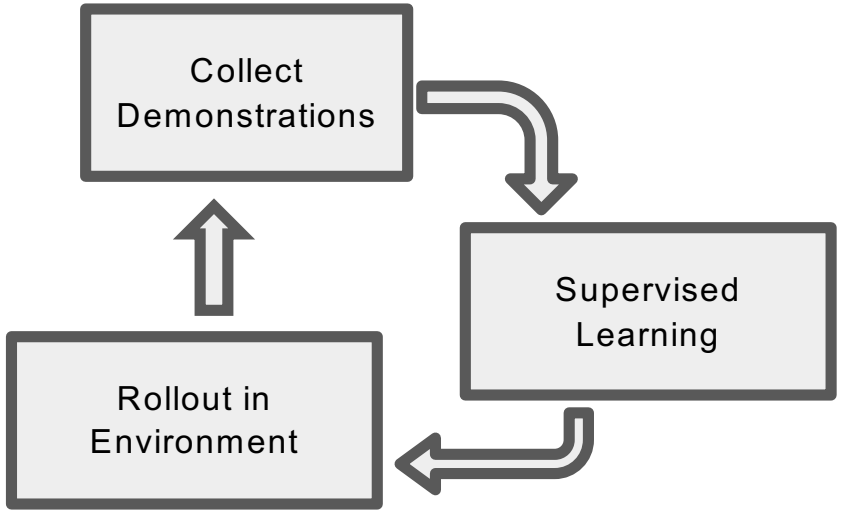
$$\pi^* = \text{argmax}_\theta \, E_{s\sim P(s|\theta)} r(s, \pi_\theta(s))$$

RL problem

Assumes learning r is statistically easier than directly learning π^*

Direct Policy Learning

via Interactive Demonstrator



Requires Interactive Demonstrator (BC is 1-step special case)

Types of Imitation Learning

	Direct Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstrations
Behavioral Cloning	Yes	No	No	No	Yes
Direct Policy Learning (Interactive IL)	Yes	No	Yes	Yes 一邊做一邊詢問專家意見	Optional
Inverse Reinforcement Learning	No	Yes	Yes	No	Yes

Interactive Direct Policy Learning

Behavioral Cloning is simplest example

Beyond BC: using interactive demonstrator

Often analyzed via **learning reductions**

- Reduce “harder” learning problem to “easier” one
- Imitation Learning → Supervised Learning
- General Overview: http://hunch.net/~il/projects/reductions/reductions.html

Learning Reductions

Behavioral Cloning:

$$\underbrace{E_{s\sim P(s|\theta)} L(a^*(s), \pi_\theta(s))}_{A} \rightarrow \underbrace{E_{(s,a^*)\sim P^*} L(a^*, \pi_\theta(s))}_{B}$$

What does learning well on B imply about A?

- E.g., can one lift PAC learning results from B to A?

Basic Results for Behavioral Cloning

Assume $L(a^*,a^*) = 0$

Suppose: $\varepsilon = E_{(s,a^*)\sim P^*} L(a^*,\pi_\theta(s))$

Then: $E_{s\sim P(s|\theta)} L(a^*(s),\pi_\theta(s)) = O(T\varepsilon)$

- Error on T-step trajectory is $O(T^2\varepsilon)$ 錯誤經過時間累加後放大~

*Paraphrased from

Theorem 2.1 in [Efficient Reductions for Imitation Learning - Ross & Bagnell, AISTATS 2010]

Lemma 3 in [A reduction from apprenticeship learning to classification - Syed & Schapire, NIPS 2010]

Direct Policy Learning vs Interactive Expert

Sequential Learning Reductions

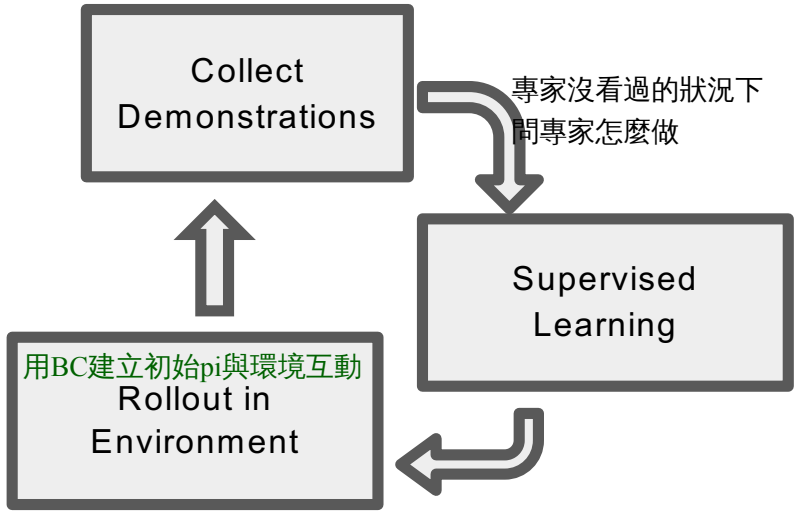
Sequence of distributions:

- $E_{s\sim P(m)} L(\pi^*(s), \pi(s))$
- Ideally converges to π_{OPT}

Best in policy class

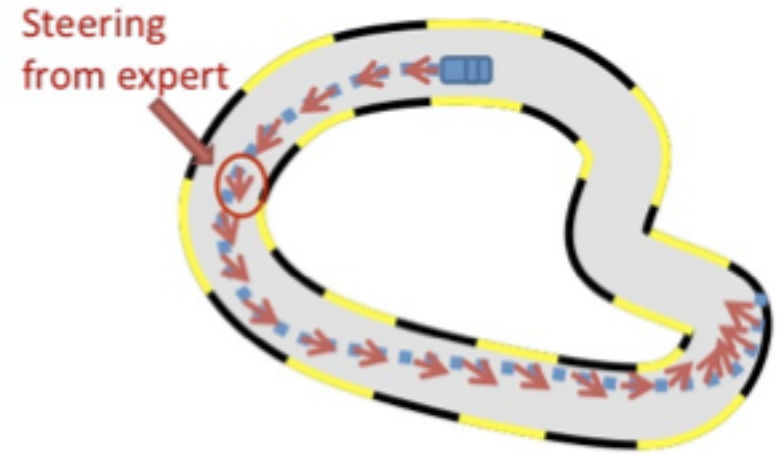
Usually starting from:

- $E_{(s)\sim P^*} L(\pi^*(s), \pi(s))$



Requires Interactive Demonstrator (BC is 1-step special case)

Interactive Expert



Example from Super Tux Kart
(Image courtesy of Stephane Ross)

Can query expert at any state

Construct loss function

- $L(\pi^*(s), \pi(s))$

Typically applied to rollout trajectories

- $s \sim P(s|\pi)$

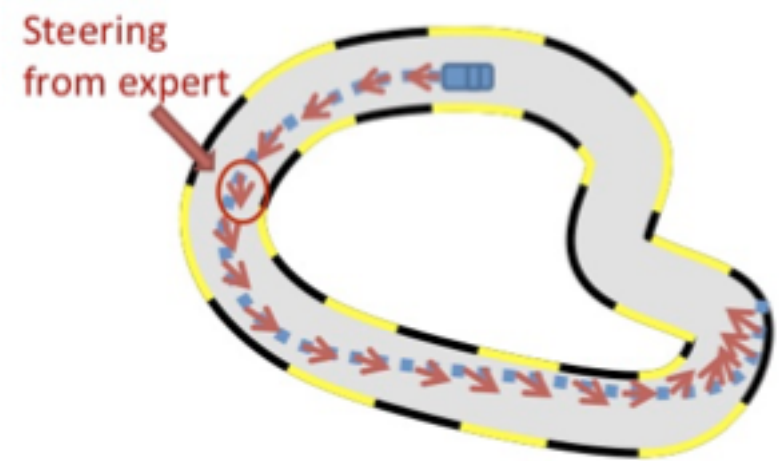
Driving example: $L(\pi^*(s), \pi(s)) = (\pi^*(s) - \pi(s))^2$

Expert provides feedback on state visited by policy

Images from Stephane Ross

Alternating Optimization (Naïve Attempt)

1. Fix P, estimate π 由收集來的feedback更新
   - Solve $\text{argmin}_\theta E_{s\sim P} L(\pi(s), \pi_\theta(s))$
2. Fix π, estimate P 蒐集新資料
   - Empirically estimate via rolling out π
3. Repeat



Not guaranteed to converge!

Images from Stephane Ross

Sequential Learning Reductions

Initial predictor: π_0 ← **Initial expert demonstrations**

For m=1

- Collect trajectories $\boldsymbol{\tau}$ via rolling out π_{m-1} ← **Typically roll out multiple times**
- Estimate state distribution P_m using $s \in \boldsymbol{\tau}$
- Collect interactive feedback $\{\pi^*(s) | s \in \boldsymbol{\tau}\}$ ← **Requires interactive expert**
- **Data Aggregation** (e.g., DAgger)
  - Train π_m on $P_1 \cup \ldots \cup P_m$
- **Policy Aggregation** (e.g., SEARN & SMILe)
  - Train π'_m on P_m
  - $\pi_m = \beta\pi'_m + (1-\beta)\pi_{m-1}$

Data Aggregation (DAgger)

類似replay buffer
把之前的s,a pair留下來
可以一直重複利用(sample)

Sequence of convex losses: $L_m(\pi) = E_{s\sim P(m)} L(\pi^*(s), \pi(s))$

Online Learning: find sequence π_m competitive with π_{OPT}:

$$R_M = \left(\frac{1}{M}\right)\sum_m L_m(\pi_m) - \left(\frac{1}{M}\right)\sum_m L_m(\pi_{OPT})$$

"Online Regret"

Best in policy class

Follow-the-Leader: $\pi_m = \min_\theta \sum_{m'=1}^{m} L_{m'}(\pi_\theta)$

$R_M = O(1/\sqrt{M})$ (M>T)

$$\exists \pi_m: \left(\frac{1}{M}\right)\sum_{m'} L_{m'}(\pi_{OPT}) \le \left(\frac{1}{M}\right)\sum_{m'} L_{m'}(\pi_m) - O\left(\frac{1}{\sqrt{M}}\right)$$

Typically π_M

A reduction of imitation learning and structured prediction to no-regret online learning
Stephane Ross, Geoff Gordon, Drew Bagnell, AISTATS 2011

Policy Aggregation (SEARN & SMILe)

Train π'_m on $P_m \longrightarrow \pi_m = \beta\pi'_m + (1-\beta)\pi_{m-1}$

透過beta去保證我能比之前更好

$$\pi_m=(1-\beta)^M\pi_0 + \beta\sum_{m'}(1-\beta)^{M-m'}\pi'_{m'}$$

π_0 is expert

(not available at test time)

π_m not much worse than π_{m-1}

- At most $\beta T L_m(\pi'_m) + \beta^2T^2/2$ for T-step rollout

π_M not much worse than π_0

π_0 negligible in π_M

- At most $2T\log(T)(\frac{1}{M})\sum_m L_m(\pi'_m) + O(1/T)$ for T-step rollout

Search-based Structured Prediction
Hal Daume et al., Machine Learning 2009

Efficient Reductions for Imitation Learning
Stephan Ross, Drew Bagnell, AISTATS 2010

Direct Policy Learning via Interactive Expert

Reduction to sequence of supervised learning problems

- Constructed from roll-outs of previous policies
- Requires interactive expert feedback

避免忘記過去的錯誤

過去policy權重相加(現在+以前)

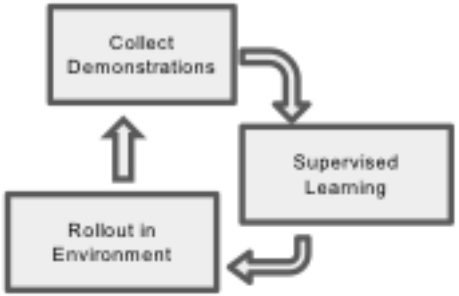
Two approaches: Data Aggregation & Policy Aggregation

- Ensures convergence
- Motivated by different theory

Not covered:

- What is expert feedback & loss function?

Depends on application



Types of Imitation Learning

Behavioral Cloning

$$\text{argmin}_\theta \, E_{(s,a^*)\sim P^*} L(a^*, \pi_\theta(s))$$

Works well when P* close to Pθ

Inverse RL

Learn r such that:

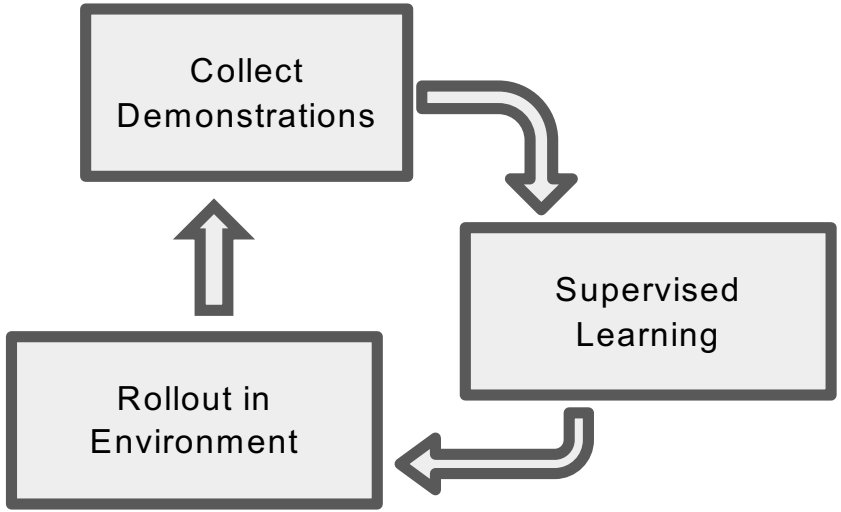
$$\pi^* = \text{argmax}_\theta \, E_{s\sim P(s|\theta)} r(s, \pi_\theta(s))$$

RL problem

Assumes learning r is statistically easier than directly learning π^*

Direct Policy Learning

via Interactive Demonstrator



Requires Interactive Demonstrator (BC is 1-step special case)

Learn Policy w/o Expert: Reinforcement Learning

MDP Formulation: $(S, A, P, r, \gamma, p_0)$

Reward function Starting state distribution

Transition model $P(s_{t+1}|s_t, a_t)$

Goal: maximize cumulative rewards

$$\max_{\pi \in \Pi} V(\pi) \triangleq \max_{\pi \in \Pi} \mathbb{E}_{\pi}\left[r(s,a)\right] = \max_{\pi \in \Pi} \mathbb{E}_{s_0 \sim p_0}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| \pi\right]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$$

Fully specified MDP: value & policy iteration

Learn Policy w/o Expert: Reinforcement Learning

MDP Formulation: $(S, A, \cancel{P}, r, \gamma, p_0)$

Optimize Value Function wrt Parameterized Policy π_θ

$$V(\pi_\theta) = \mathbb{E}_{s_0 \sim p_0}\left[\texttt{cumulative rewards} \,\middle|\, \pi_\theta\right]$$

$$Q^{\pi_\theta}(s, a) = \mathbb{E}\left[\texttt{cumulative rewards} \,|\, \pi_\theta, s_0 = s, a_0 = a\right]$$

Policy Gradient Theorem (Sutton et al., ICML 1999)

$$\nabla_\theta V(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)\right]$$

Learn Policy w/o Expert: Reinforcement Learning

MDP Formulation: $(S, A, \cancel{P}, r, \gamma, p_0)$

REINFORCE algorithm:

for each trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_T, a_T\} \sim \pi_\theta$:

  for $t = 0, \ldots, T-1$:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) \widehat{Q}^{\pi_\theta}(s_t, a_t)$$

Challenges with Reward Engineering

MDP Formulation: $(S, A, P, r, \gamma, p_0)$ assumed given

https://blog.openai.com/faulty-reward-functions/

Inverse Reinforcement Learning

MDP Formulation: $(S, A, P, \cancel{r}, \gamma, p_0)$

Given: $\mathcal{D} = \{\tau_1, \ldots, \tau_m\} = \{(s_0^i, a_0^i, s_1^i, a_1^i \ldots)\} \sim \pi^*$

Goal: Learn a reward function r^* so that

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \, \mathbb{E}_\pi \left[r^*(s, a) \right]$$

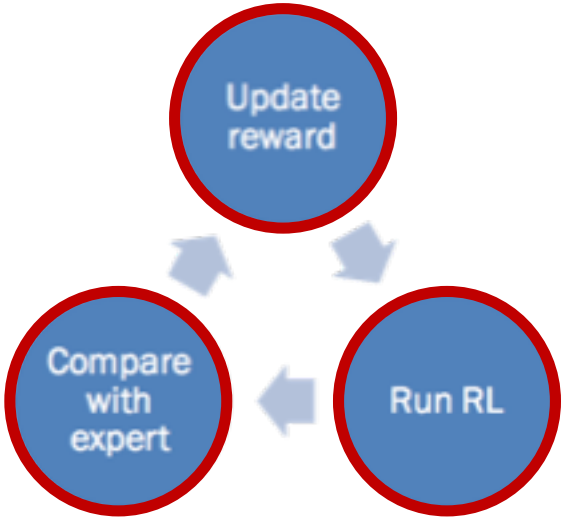
can also be $\mathbb{E}_\pi \left[r^*(s) \right]$

Inverse Reinforcement Learning

- Inverse RL high-level recipe:

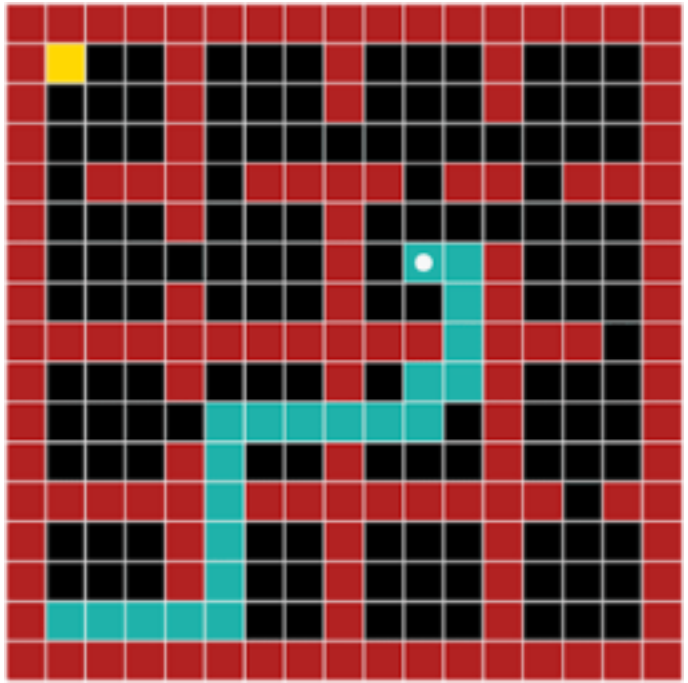
  π_E or π^*

  - Expert demonstrations: $\mathcal{D} = \{\tau_1, \ldots, \tau_m\}$
  - Learn reward function: $r_\theta(s_t, a_t)$ $r_\theta(s,a) = \theta^\top \phi(s,a)$
  - Learn policy given reward function (RL)
  - Compare learned policy with expert



Reward Learning is Ambiguous

1. Many reward functions correspond to the same policy



Imitation Learning via Inverse RL (model-given)

Abbeel & Ng, ICML '04

Goal: find reward function r

expert 的reward一定是最大的，如果被超過，就會更新reward function

$$\max_{\pi \in \Pi} \mathbb{E}_{\pi} \left[r(s,a) \right] \geq \mathbb{E}_{\pi^*} \left[r^*(s,a) \right] - \epsilon$$

差距在於某個給定的值

Different from "idealized" IRL

Game-Theoretic Inverse RL (model-given)

Syed & Schapire, NIPS '07

Goal: find π performing better than π^* over a class of rewards

$$\max_{\pi \in \Pi} \min_{r \in \mathcal{R}} \mathbb{E}_{\pi} \left[r(s,a) \right] - \mathbb{E}_{\pi^*} \left[r(s,a) \right]$$

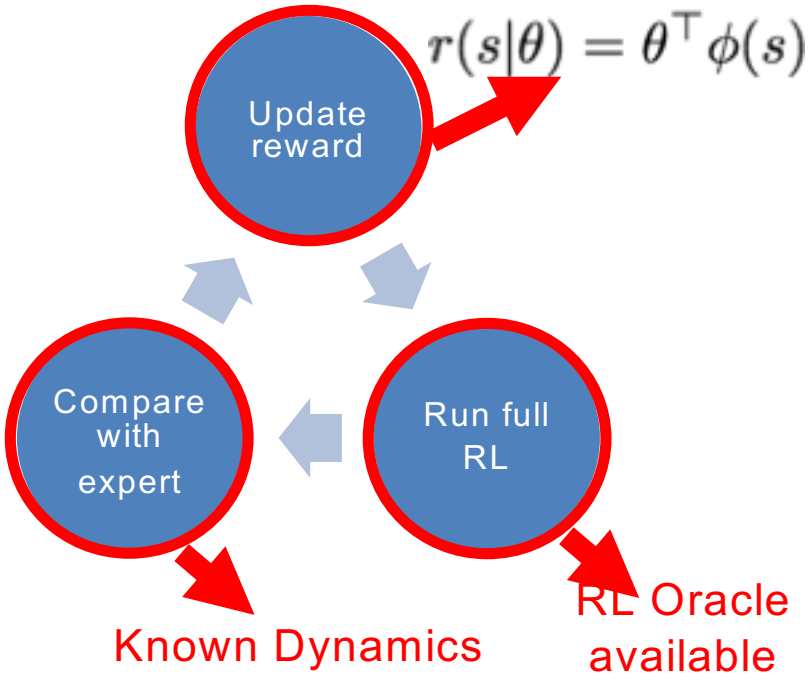
Different from "idealized" IRL

Imitation Learning via Inverse RL (model-given)

Abbeel & Ng, ICML '04
Syed & Schapire, NIPS '07

Assumptions:

$$(S, A, P, \cancel{R}, \gamma, p_0)$$



Syed & Schapire, NIPS '07 $\|\theta\|_1 = 1, \theta \succeq 0$

Abbeel & Ng, ICML '04 $\|\theta\|_2 \leq 1$

Linear Reward ⟹ Feature Expectations

Abbeel & Ng, ICML '04
Syed & Schapire, NIPS '07

Assume: $r(s) = \theta \cdot \phi(s)$

Value of a policy expressed in terms of feature expectation

$$V(\pi|s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \theta \cdot \phi(s_t)|\pi\right]$$

$$= \theta \cdot \underbrace{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)\right]}_{\mu(\pi)}$$

feature expectations

Feature Matching ⟹ Optimality

Abbeel & Ng, ICML ‘04
Syed & Schapire, NIPS ‘07

Feature Expectation:

$$\mu(\pi) = \mathbb{E}\left[\texttt{visited state features} \mid \pi \right]$$

If reward r is linear:

$$\mu(\pi) = \mu(\pi^*) \Rightarrow V(\pi) = V(\pi^*)$$

Don’t know exactly

Feature Matching in Inverse RL (model-given)

Abbeel & Ng, ICML '04

L2 norm

Find π s.t $\|\mu(\pi) - \mu(\pi^*)\|_2 \leq \epsilon$

Why? $|V(\pi) - V(\pi^*)| = |\theta^\top \mu(\pi) - \theta^\top \mu(\pi^*)|$

$\leq \|\theta\|_2 \|\mu(\pi) - \mu(\pi^*)\|_2 \leq 1 \cdot \epsilon$

Algorithm: solve max-margin problem in each loop

$$\max_{\eta,\theta} \quad \eta$$

$$\text{s.t.} \quad \theta^\top \mu(\pi^*) \geq \theta^\top \mu(\pi_j) + \eta, j = 0, \ldots, i-1$$

$$\|\theta\|_2 \leq 1$$

小成某個程度以後就會停止

Stop when $\eta \leq \epsilon$

Theory: at most $O(\frac{k}{(1-\gamma)^2\epsilon^2} \log \frac{k}{(1-\gamma)\epsilon})$ iterations

Feature Expectation Matching

Want to find π s.t discounted state visitation features match the expert $\mu(\pi) \approx \mu(\pi^*)$

Ill-posed problem → regularization

Another regularization method: maximum entropy

Policy Learning is Still Ambiguous

1. Many reward functions correspond to the same policy

2. Many stochastic mixtures of policies correspond to the same feature expectation

$$\left.\begin{array}{l} \mu(\pi_1) \approx \mu(\pi^*) \\ \\ \mu(\pi_2) \approx \mu(\pi^*) \end{array}\right\} \mu\left(\alpha\pi_1 + (1-\alpha)\mu_2\right) \approx \mu(\pi^*)$$

Maximum Entropy Principle

Policy π induces distribution over trajectories $P(\tau)$

Feature matching: $\sum_{\tau} P(\tau)\mu(\tau) = \mu(\pi^*)$ expert 期望值

$$\sum_{\tau} P(\tau) = 1$$

Maximum entropy principle: The probability distribution which best represents the current state of knowledge is the one with largest entropy (E.T. Jaynes 1957)

Information Theory and Statistical Mechanics - E.T. Jaynes - Phys. Rev. 1957

Maximum Entropy Principle

Choose trajectory distribution that satisfies feature matching constraints without over-committing

$H(\pi)$

As uniform as possible

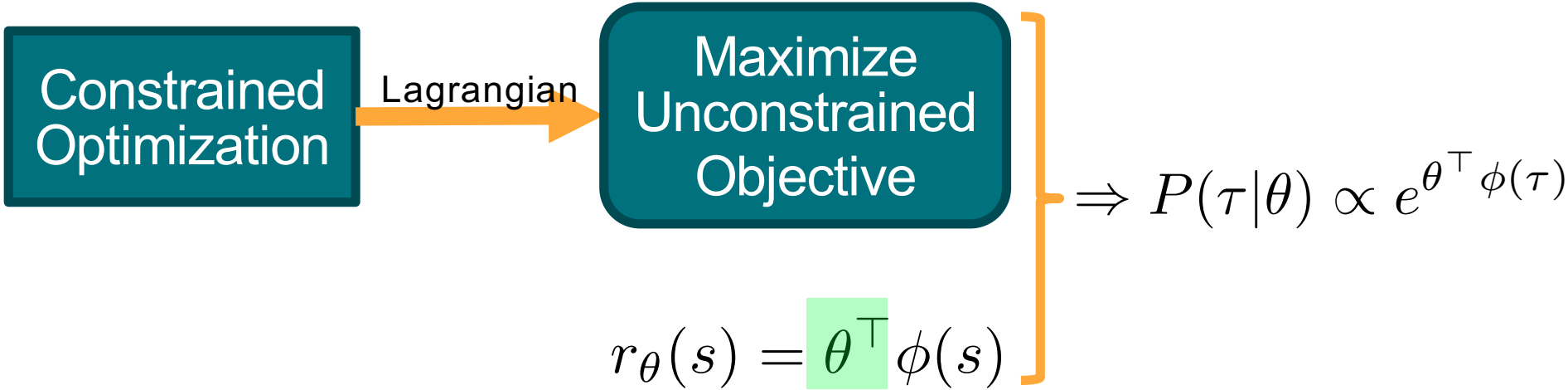
$$\max_P \quad -\sum_\tau P(\tau)\log P(\tau)$$

$$\text{s.t.} \quad \sum_\tau P(\tau)\mu(\tau) = \hat{\mu}^*$$

$$\sum_\tau P(\tau) = 1$$

$$\frac{1}{m}\sum_{\tau_i \in \mathcal{D}} \mu(\tau_i)$$

Maximum Entropy Principle



The distribution that maximizes entropy given linear constraints is in the exponential family

MaxEnt Inverse RL (model-given)

Ziebart et al., AAAI '08

MaxEnt formulation: $P(\tau|\theta) = \frac{1}{Z(\theta)} e^{\sum_{s_t \in \tau} \theta^\top \phi(s_t)}$

$Z(\theta) = \int e^{r(\tau|\theta)} d\tau$

希望trajectory之間距離越大越好(明顯看到某個trajectory很好)
分布越平均 越不好看出好壞

reward inference: max log likelihood

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{\tau_i \in \mathcal{D}} \log P(\tau_i|\theta)$$

gradient descent

MaxEnt Inverse RL (model-given)

Ziebart et al., AAAI '08

Gradient descent over log-likelihood

$$\nabla_\theta L(\theta) = \frac{1}{m} \sum_{\tau_i \in \mathcal{D}} \underbrace{\mu(\tau_i)}_{\text{expert state feature}} - \sum_s \underbrace{d_s^\theta}_{\text{state occupancy measure}} \phi(s)$$

$$= \frac{1}{T} \sum_{s' \in \tau_i} \phi(s')$$

Dynamic programming: state occupancy measure (visitation freq)

$$d_{t+1,s'} = \sum_a \sum_s d_{t,s} \pi_\theta(a|s) P(s'|s,a)$$

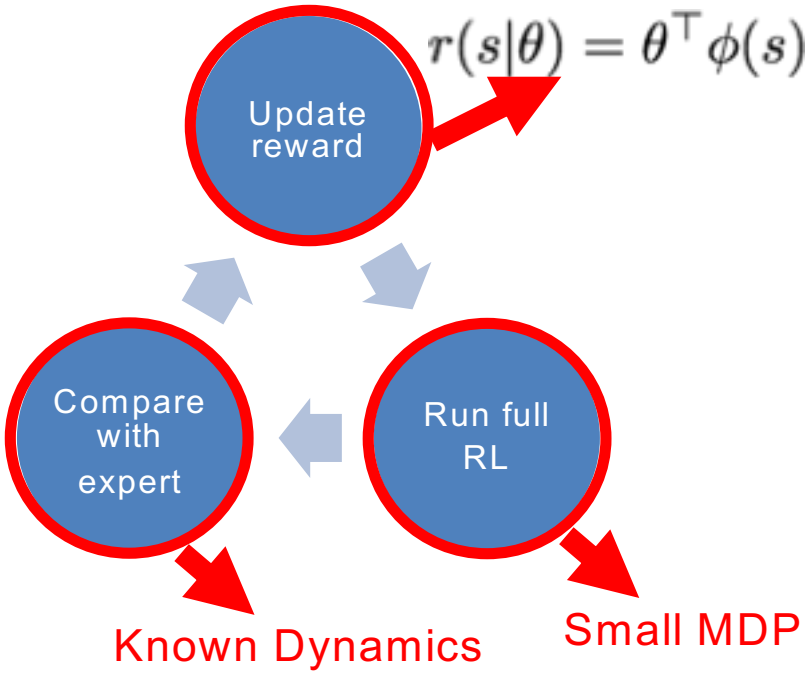
Solve forward RL w.r.t. θ

Given from the model

Inverse RL So Far...

Model-given:

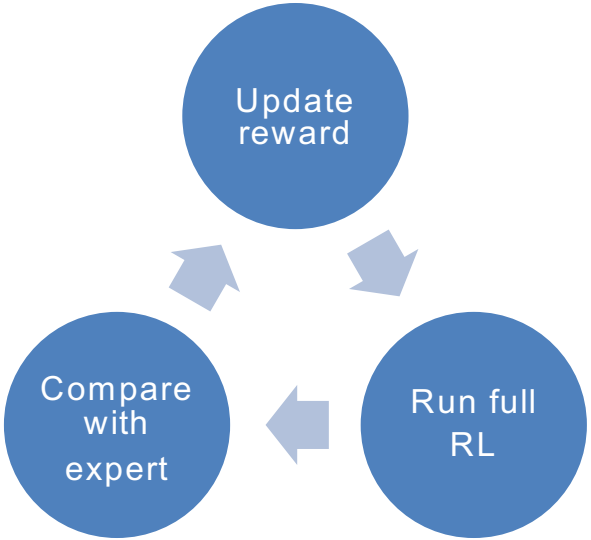
$$(S, A, P, \gamma, p_0)$$



Next ...

Model-given:

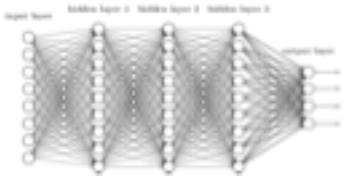
(S, A, P, γ, p_0)



Model-free:

(S, A, γ, p_0)

r_θ



Interact with environment / simulator (Model-free)

Large / continuous space

MaxEnt Deep IRL + Model-Free Optimization

Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization - Finn et al., ICML '16

reward $r_\theta(s_t, a_t)$ parameterized by neural net

Recall MaxEnt formulation $P(\tau|\theta) = \frac{1}{Z(\theta)} e^{r_\theta(\tau)}$

$$Z(\theta) = \int e^{r(\tau|\theta)} d\tau$$

Max likelihood objective

$$\theta^* = \operatorname*{argmax}_\theta L(\theta) = \operatorname*{argmax}_\theta \sum_{\tau_i \in \mathcal{D}} \log P(\tau_i|\theta)$$

$$= \operatorname*{argmax}_\theta \frac{1}{|\mathcal{D}|} \sum_{\tau_i \in \mathcal{D}} r_\theta(\tau_i) - \log Z(\theta)$$

Need to sample to evaluate gradients

MaxEnt Deep IRL + Model-Free Optimization

Finn et al., ICML '16

Approximate $Z(\theta) = \int e^{r(\tau|\theta)} d\tau$

Use "proposal" distribution $q(\tau)$ to sample trajectories $\mathcal{D}_{\text{samp}}$

$$Z(\theta) \approx \underset{\tau \in \mathcal{D}_{\text{samp}}}{\text{average}} \left(\frac{e^{r_\theta(\tau)}}{q(\tau)} \right)$$

Reward Optimization

Optimal dist $q(\tau) \propto e^{r_\theta(\tau)}$, which depends on optimal policy w.r.t. unknown r_θ

Policy Optimization

MaxEnt Deep IRL + Model-Free Optimization

Finn et al., ICML '16

Estimated log likelihood:

$$L(\theta) \approx \frac{1}{m} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} r_\theta(\tau_i) + \log \frac{1}{n} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} \frac{e^{r_\theta(\tau_j)}}{q(\tau_j)}$$

sampling distribution $q(\tau) \propto e^{r_\theta(\tau)}$
(also depends on θ)

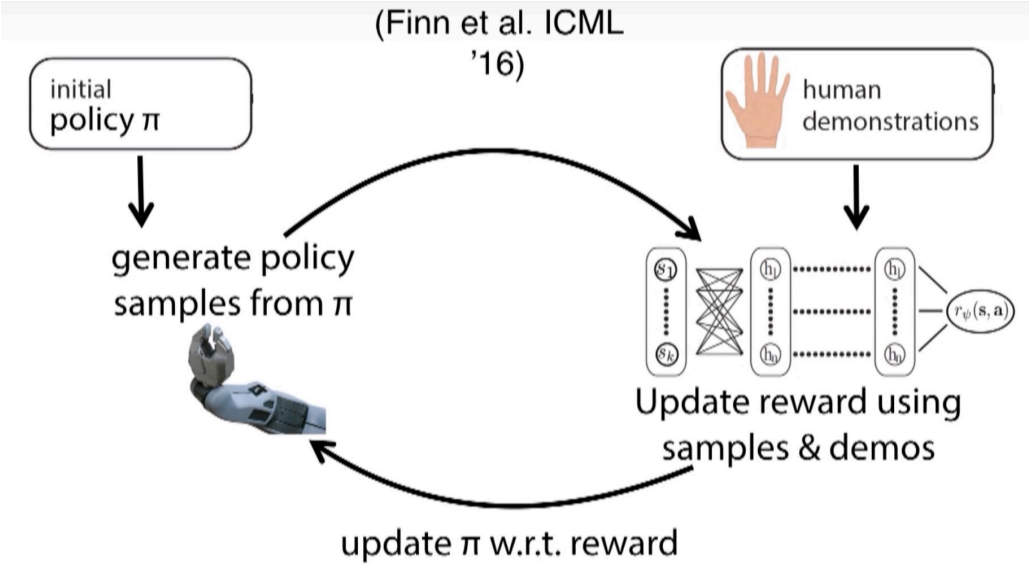
with gradient:

$$\nabla_\theta L(\theta) \approx \frac{1}{m} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \nabla_\theta r_\theta(\tau_i) + \frac{1}{Z} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} w_j \nabla_\theta r_\theta(\tau_j)$$

$$\text{where } w_j = \frac{e^{r_\theta(\tau_j)}}{q(\tau_j)} \text{ and } Z = \sum_j w_j$$

MaxEnt Deep IRL + Model-Free Optimization

- Generate samples by preventing the policy from changing too rapidly
- Update policy: take only 1 policy optimization step



Figures from Chelsea Finn's presentation

Tutorial Overview

Part 1: Introduction and Core Algorithms

- Teaser results
- Overview of ML landscape
- Types of Imitation learning
- Core algorithms of passive, interactive learning and cost learning

Part 2: Extensions and Applications

- **Speech Animation**
- **Structured prediction and search**
- **Improving over expert**
- **Filtering and sequence modeling**
- **Multi-objective imitation learning**
- **Visual / few-shot imitation learning**
- **Domain Adaptation imitation learning**
- **Multi-agent imitation learning**
- **Hierarchical imitation learning**
- **Multi-modal imitation learning**
- **Weaker Feedback**

Speech Animation

A Deep Learning Approach for Generalized Speech Animation

[Taylor et al., SIGGRAPH 2017]

Input sequence $X = < x1, \dots, xT >$

Output sequence $Y = < y1, \dots, yT >$ $y \in R^D$

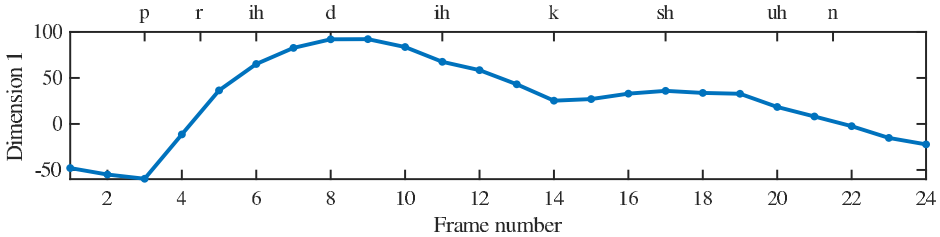
Goal: learn predictor $\pi : X \rightarrow Y$

X

Frame	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Token	-	p	p	r	ih	ih	d	d	ih	ih	ih	ih	k	k	sh	sh	sh	sh	uh	uh	n	-

Phoneme sequence

Y

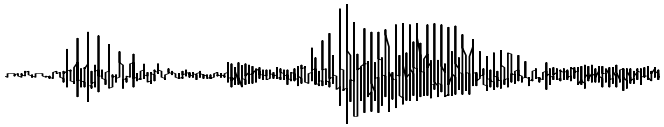


Sequence of face configurations

Train using Behavioral Cloning

A Deep Learning Approach for Generalized Speech Animation
Sarah Taylor, Taehwan Kim, Yisong Yue et al., SIGGRAPH 2017

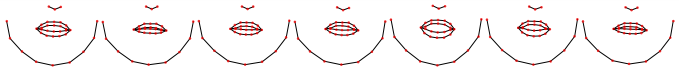
https://www.youtube.com/watch?v=9zL7qeiW9fE



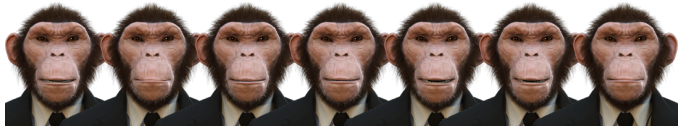
Input Audio

s s s s s ih ih ih g g r r ae ae ae ae f f f

Speech Recognition



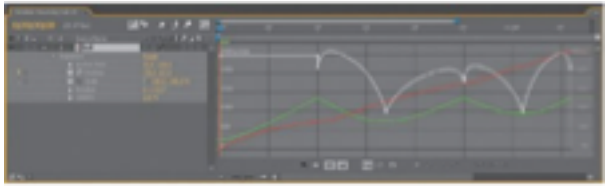
Speech Animation



(chimp rig courtesy of Hao Li)

Retargeting

E.g., [Sumner & Popovic 2004]



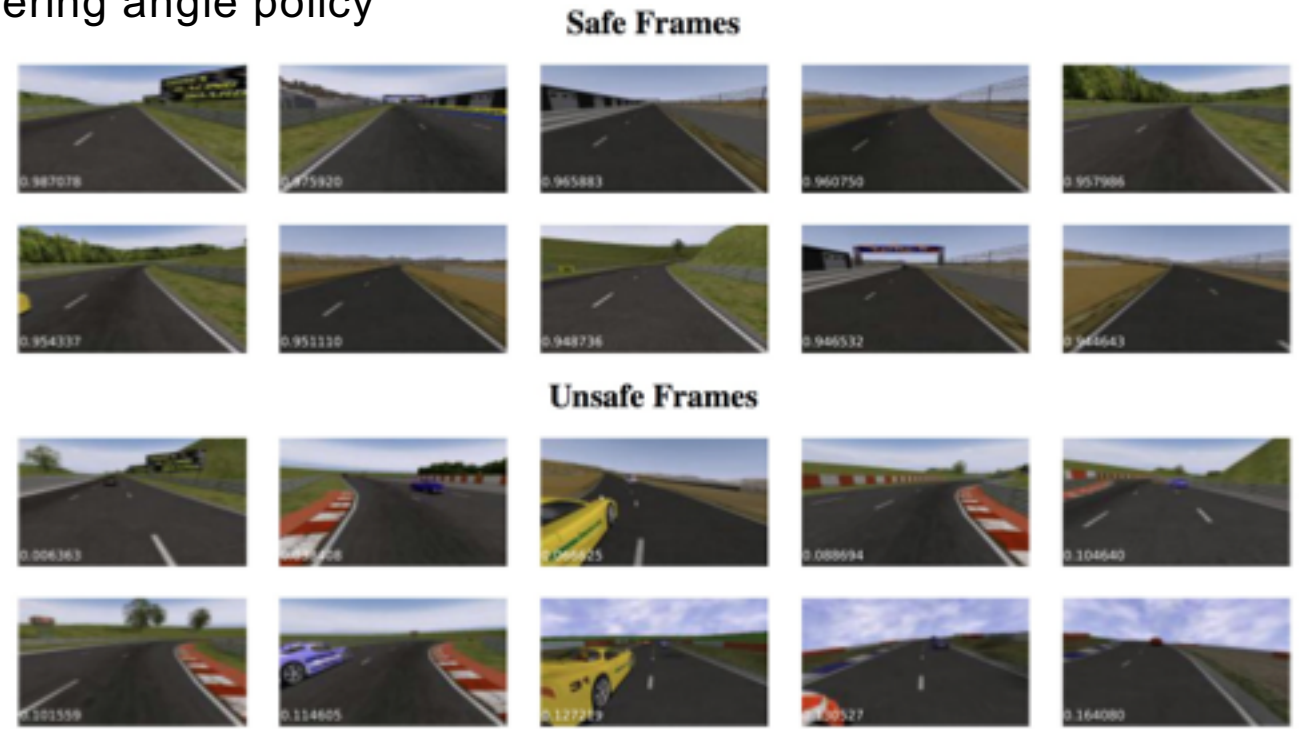
Editing

Imitation Learning with Multiple Objectives

Safe Imitation Learning for Autonomous Driving

Zhang & Cho, AAAI ‘17

π = steering angle policy



TORCS Simulation

Safe Imitation Learning for Autonomous Driving

Zhang & Cho, AAAI '17

c_{safe} = whether π likely to deviates from π^*

$$c^*_{\text{safe}}(\pi, s) = \begin{cases} 0, & \text{if } \|\pi(s) - \pi^*(s)\| > \epsilon \\ 1, & \text{otherwise} \end{cases}$$

If $c_{\text{safe}}(\pi, s) = 1$, drive with π, otherwise fall back to π^*

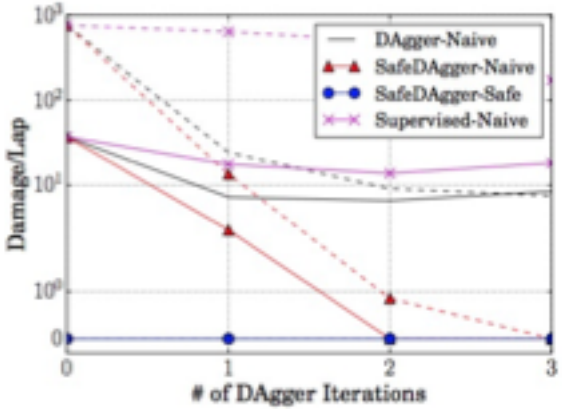
Safe Imitation Learning for Autonomous Driving

Zhang & Cho, AAAI '17

SafeDAgger:

Safety classifier in the loop

Only query "unsafe" states



Results:

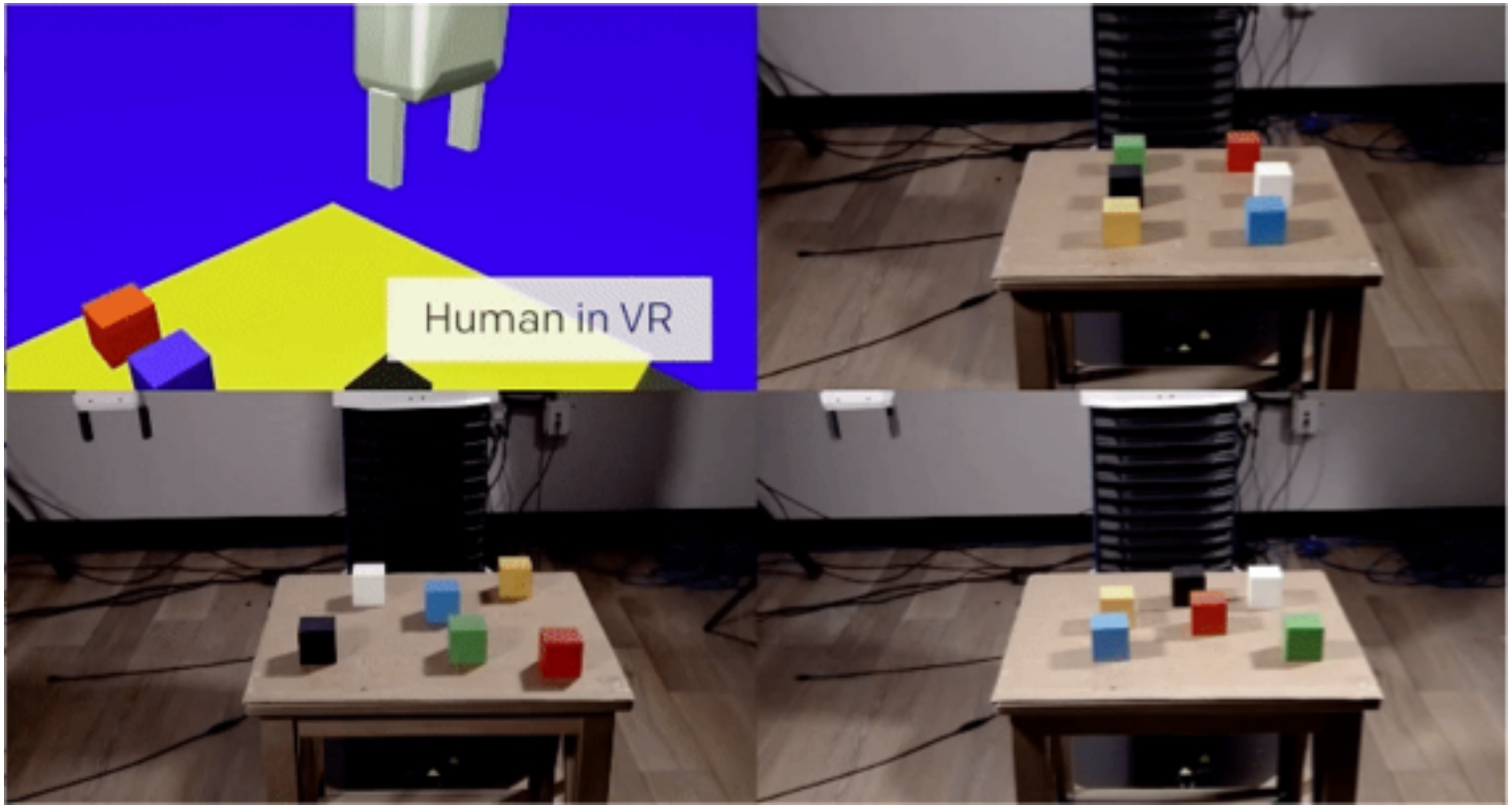
- Reduced number of damages
- Faster learning - can be viewed as active DAgger

Multi-task, Meta and Transfer Imitation Learning

One Shot Imitation Learning

https://www.youtube.com/watch?v=oMZwkliZzCM

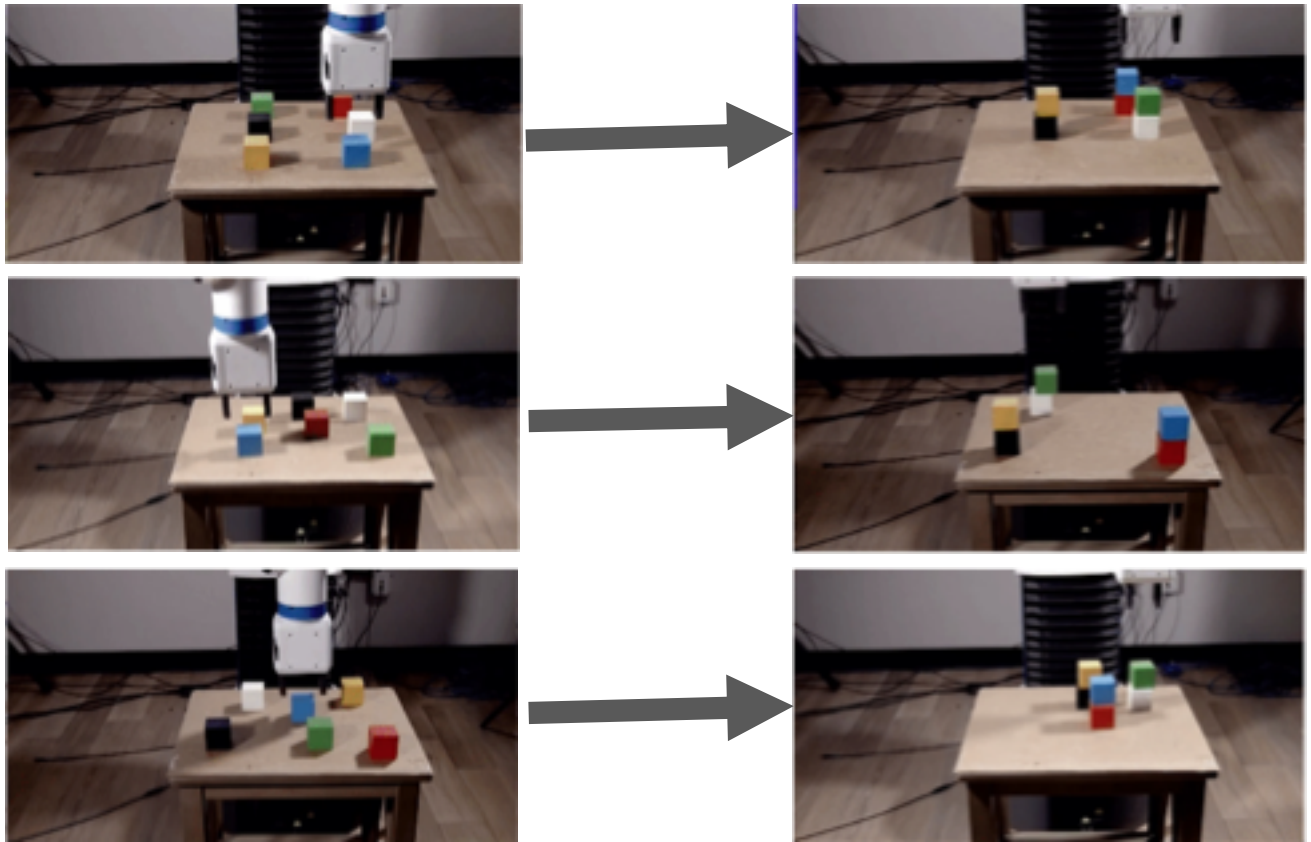
Duan et al., NIPS ‘17



One Shot Imitation Learning

Duan et al., NIPS ‘17
Tobin et al., IROS ‘17

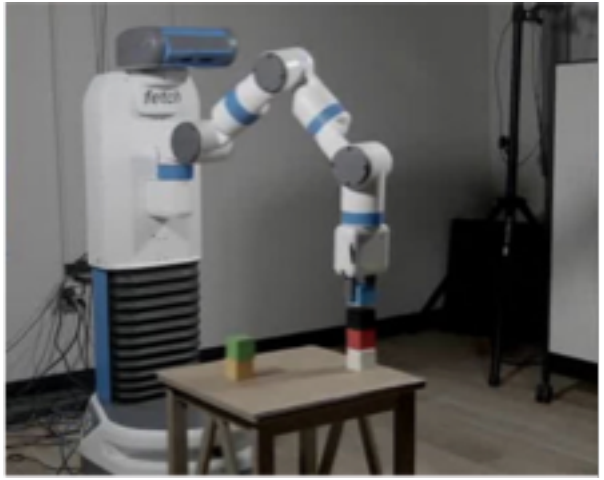
3 instances of an example task:



One Shot Imitation Learning

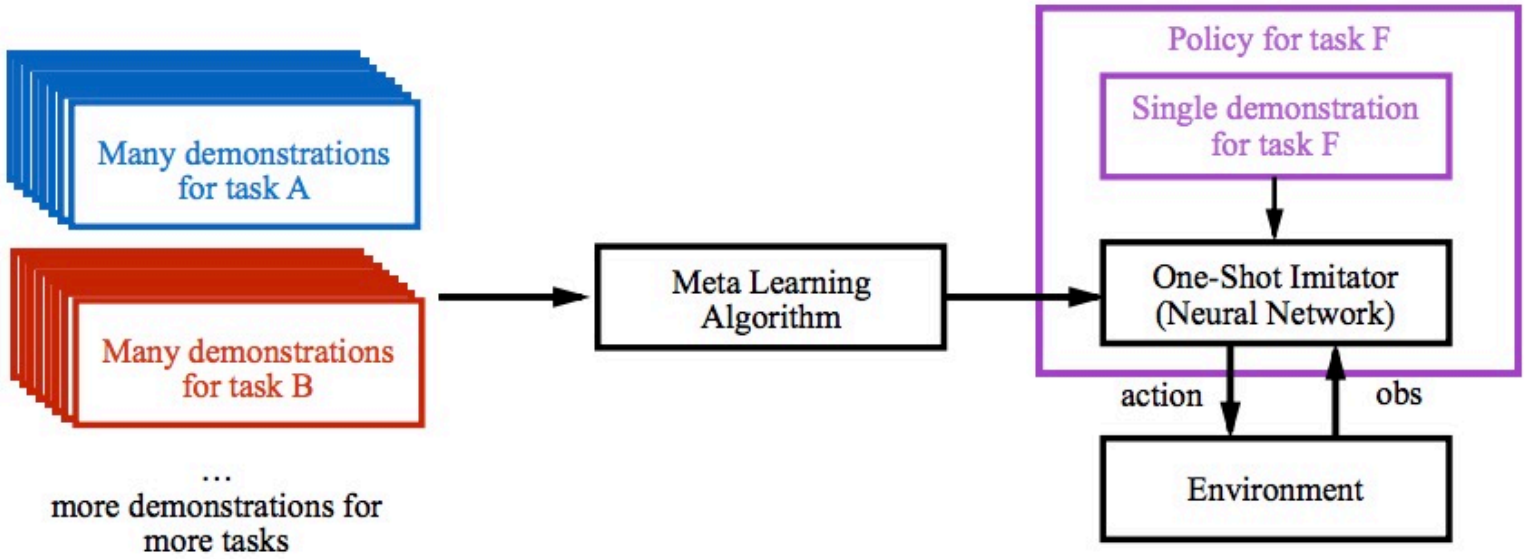
Duan et al., NIPS ‘17
Tobin et al., IROS ‘17

How about other related tasks?



One Shot Imitation Learning

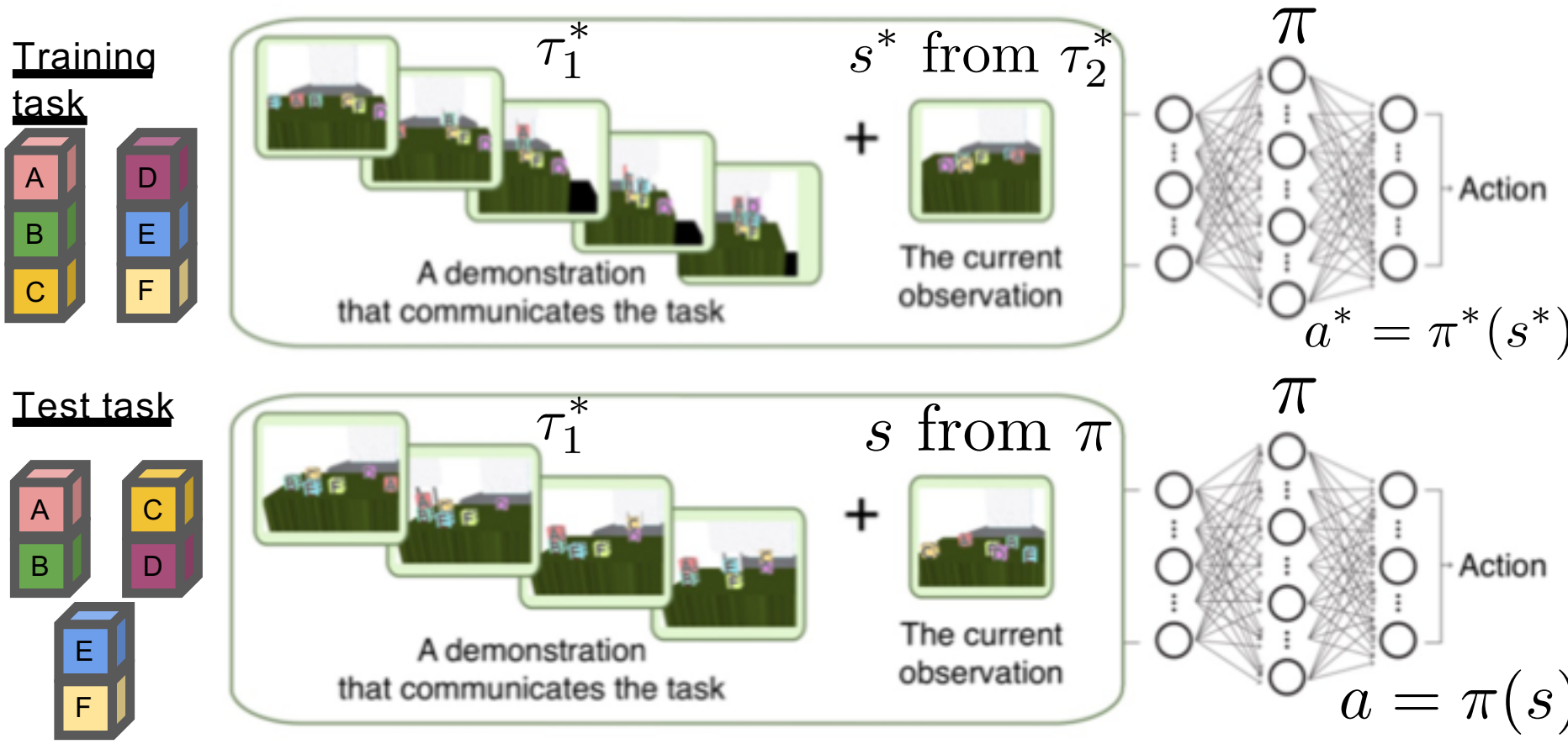
Duan et al., NIPS '17



(b) One-Shot Imitation Learning

One Shot Imitation Learning

Duan et al., NIPS '17

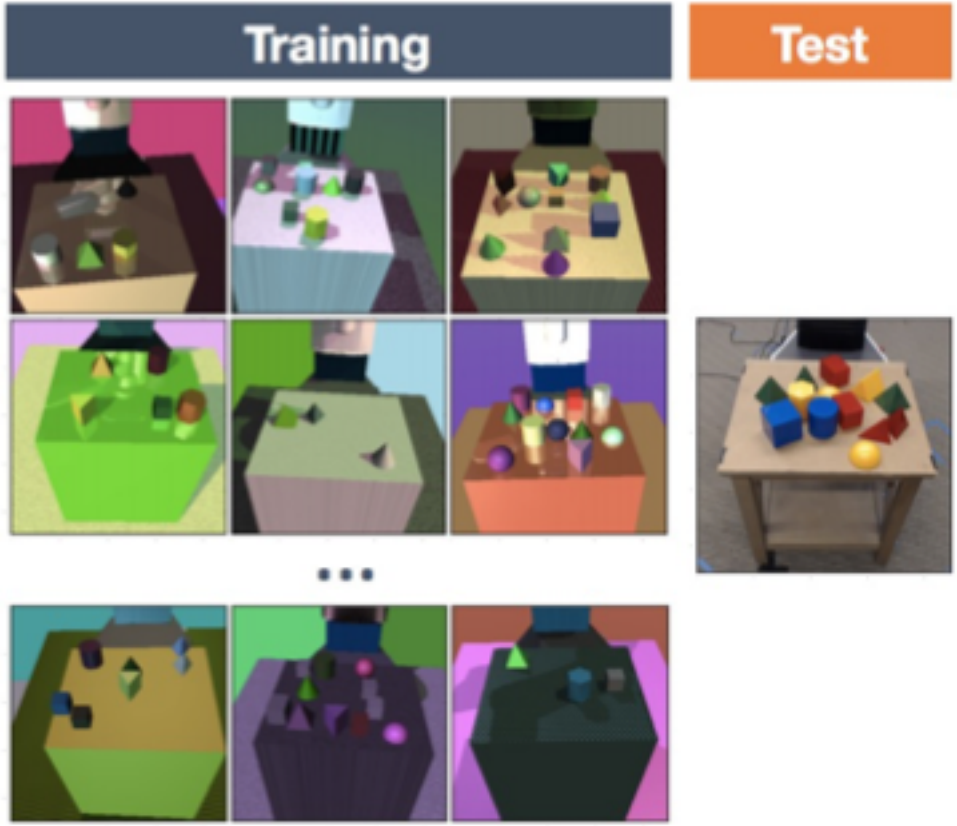


One Shot Visual Imitation Learning - Transfer

Tobin et al., IROS ‘17

Trained in simulation only using domain randomization

Test in the real world



One Shot Visual Imitation Learning - Transfer

Tobin et al., IROS ‘17

Training entirely in simulation, expert demonstrations through VR