Imitation Learning
Tutorial Talk
(October 16th, 2020)

MyungJae Shin
Seoul National University College of Medicine Hospital, Seoul, Republic of Korea
https://170928.github.io
mjshin.ml@gmail.com

Imitation Learning in a Nutshell

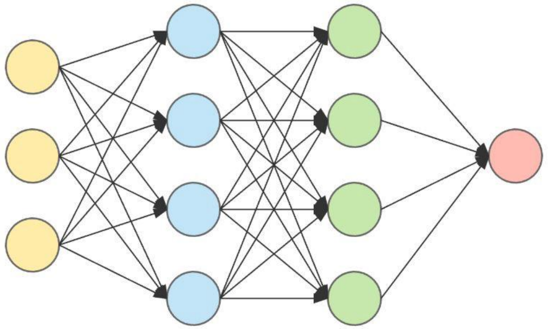
Given: demonstrations or demonstrator

Goal: train a policy to mimic demonstrations

Expert Demonstrations

State/Action Pairs

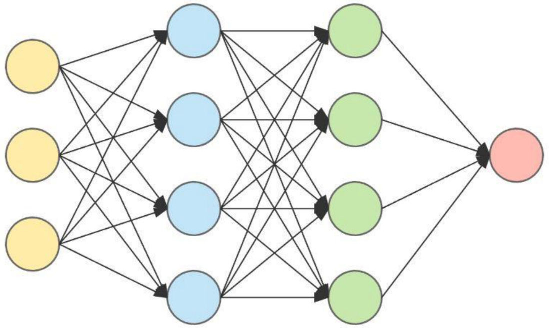
Learning



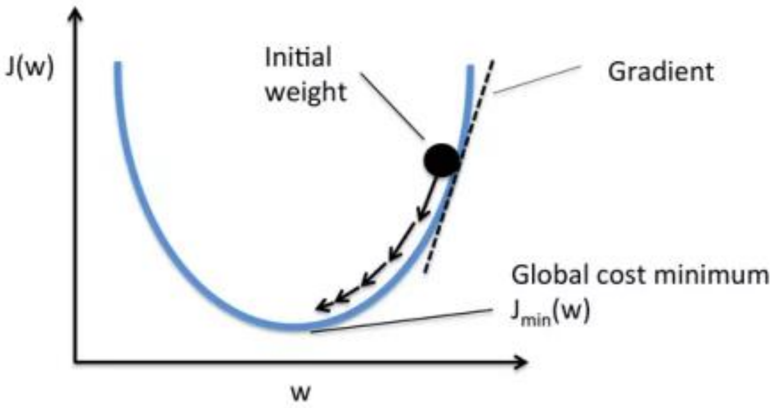
Ingredients of Imitation Learning

Demonstrations or Demonstrator

Environment / Simulator



Policy



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: Extenstions 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-model imitation learning
- Weaker Feedback

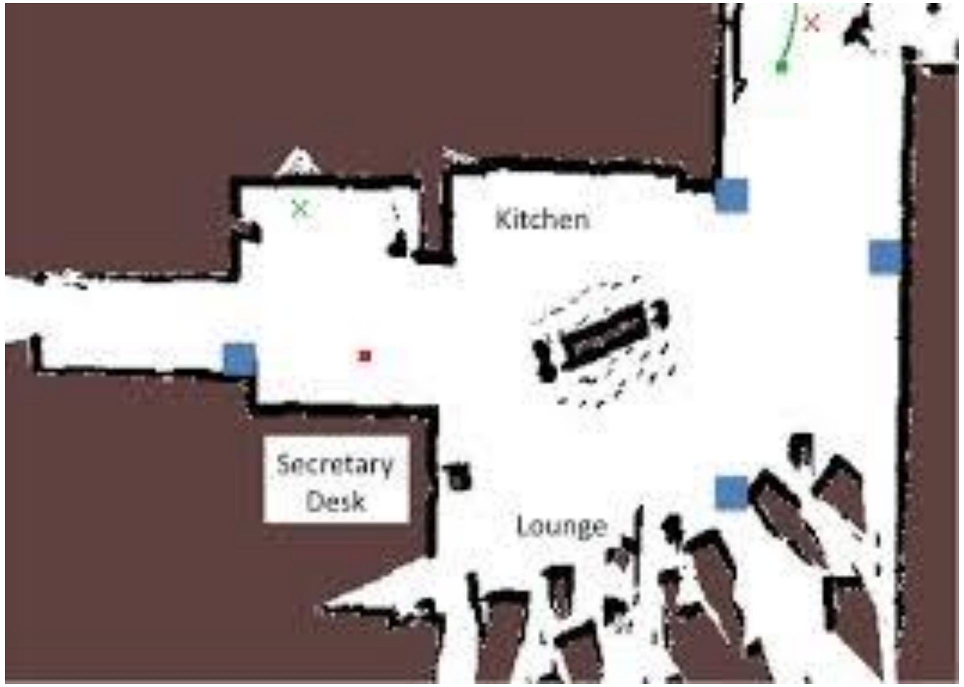
ALVINN

Dean Pomerleau et al., 1989-1999

Helicopter Acrobatics

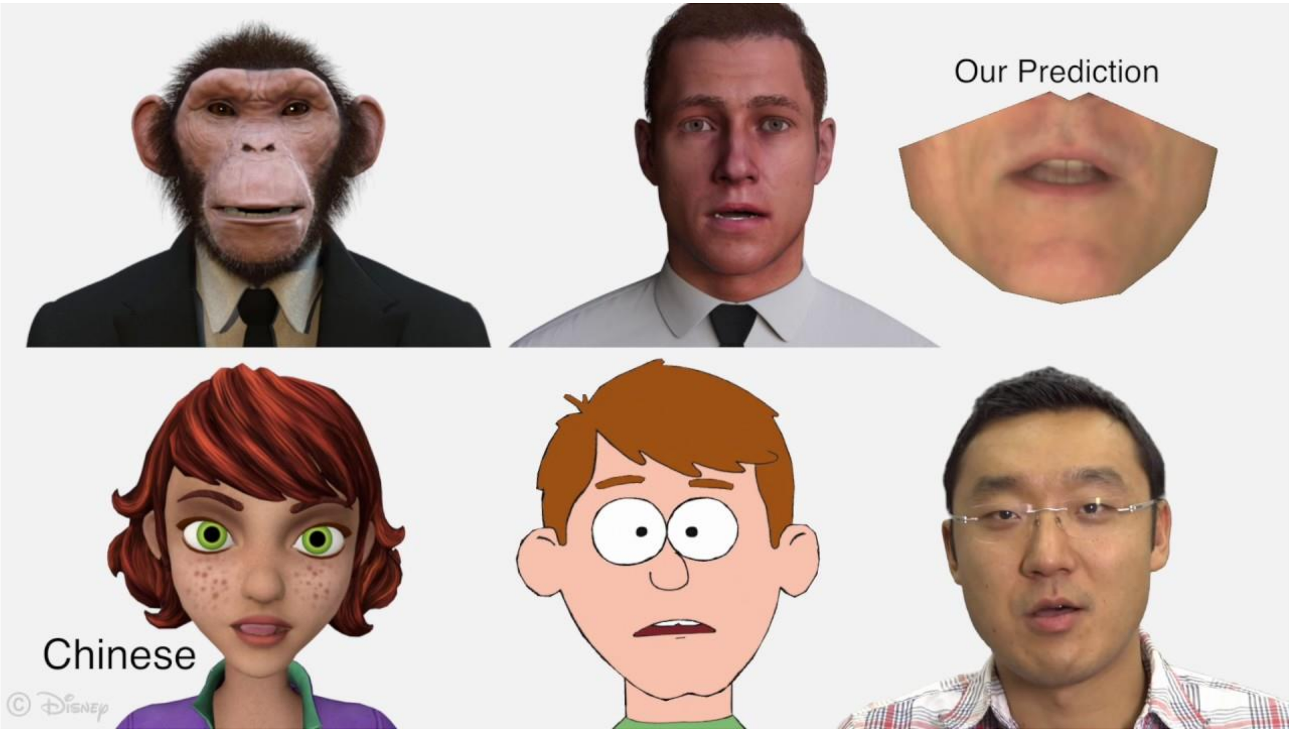
Learning for Control from Multiple Demonstrations
Adam Coates, Pieter Abbeel, Andrew Ng, ICML 2008
An Application of of Reinforcement Learning to Aerobatic Helicopter Flight
Pieter Abbeel, Adam Coabes, Morgan Quigley, Andrew Y. Ng, NIPS 2006

Inferring Human Intent



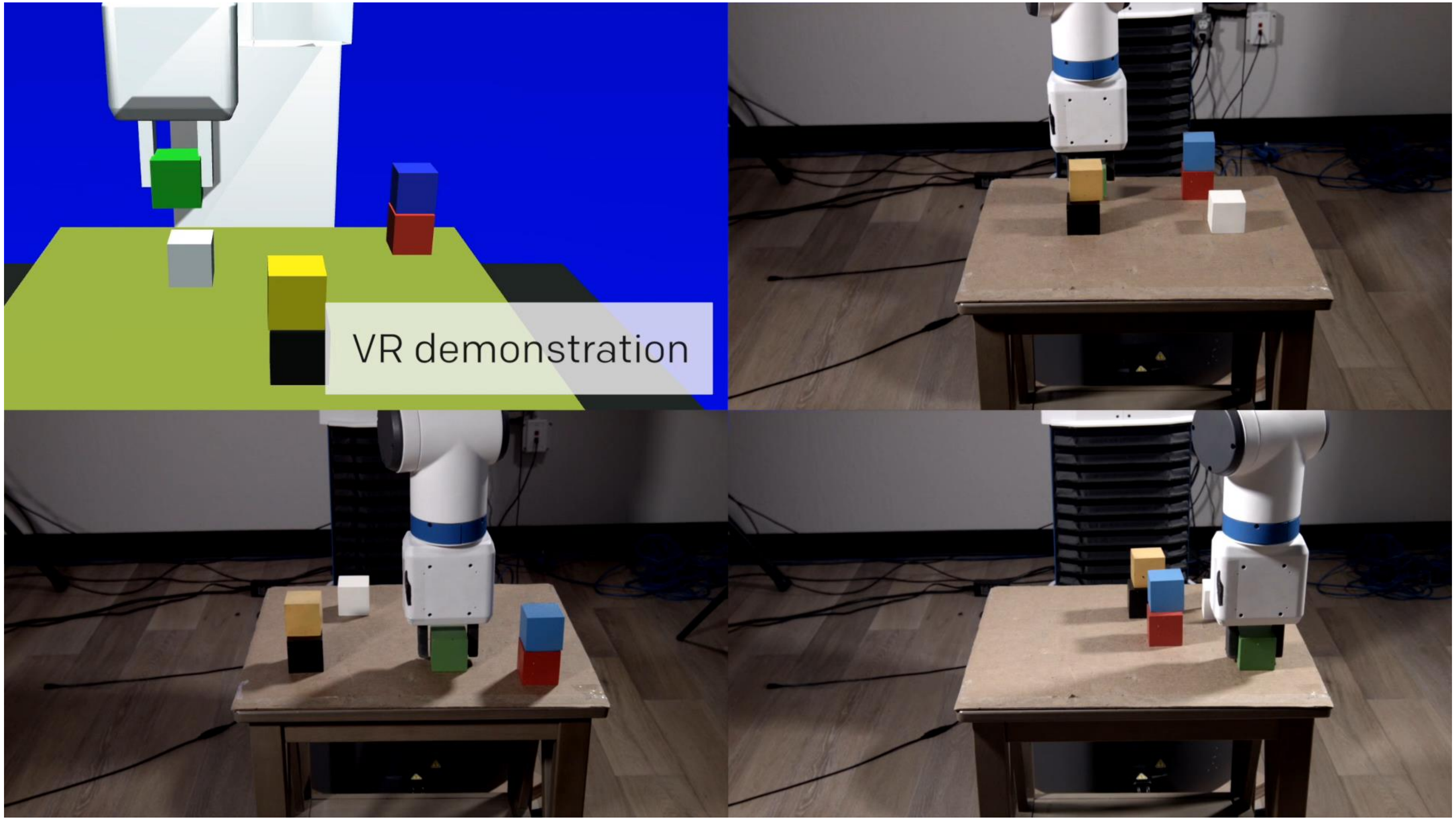
Planning-based Prediction for Pedestrians
Brian Ziebart et al., IROS 2009

Speech Animation



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

One Shot Imitation Learning



Notation & Setup

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

Action: a (sometimes y)

Policy: π_θ (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 $\boldsymbol{\tau} = (s_0, a_0, s_1, a_1, \cdots)$

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

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

$P(s|\pi)$: distribution of states 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: $D = \{\boldsymbol{\tau} \coloneqq (\mathbf{s}, \mathbf{a})\}$ from π^*

- $\mathbf{s}$ = sequence of s
- $\mathbf{a}$ = sequence of a

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

Images from Stephane Ross

Example #2: Basketball Trajectories

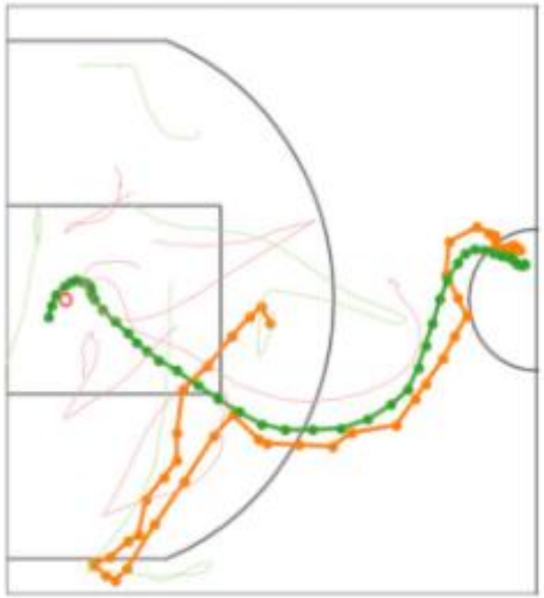
s = location of players & ball

a = next location of player

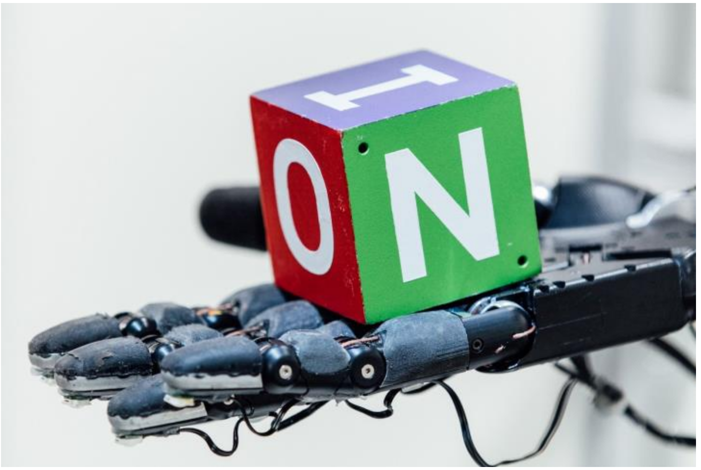
Training set: $D = \{\boldsymbol{\tau} \coloneqq (\mathbf{s}, \mathbf{a})\}$ from π^*

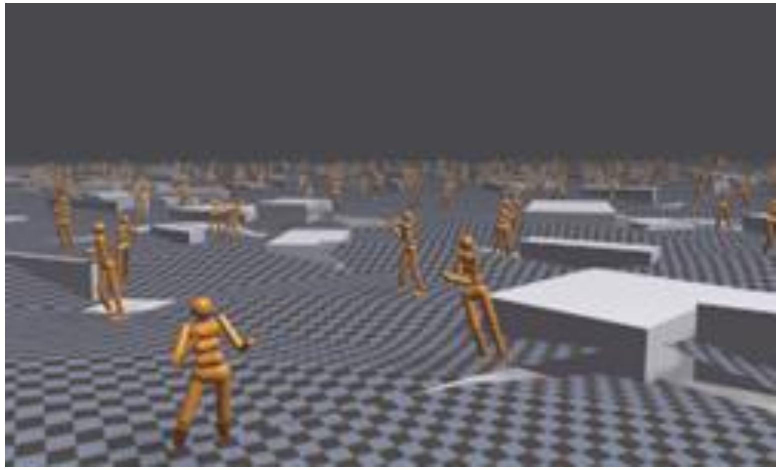
- $\mathbf{s}$ = sequence of s
- $\mathbf{a}$ = sequence of a

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



Data Sources





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

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

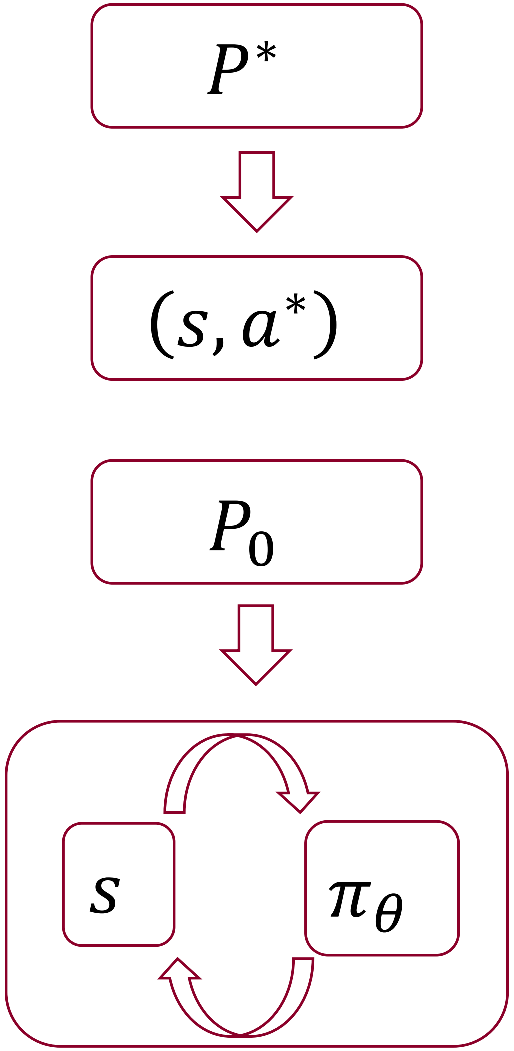
Interpretations:

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):

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

Distribution provided exogenously

Training Loss

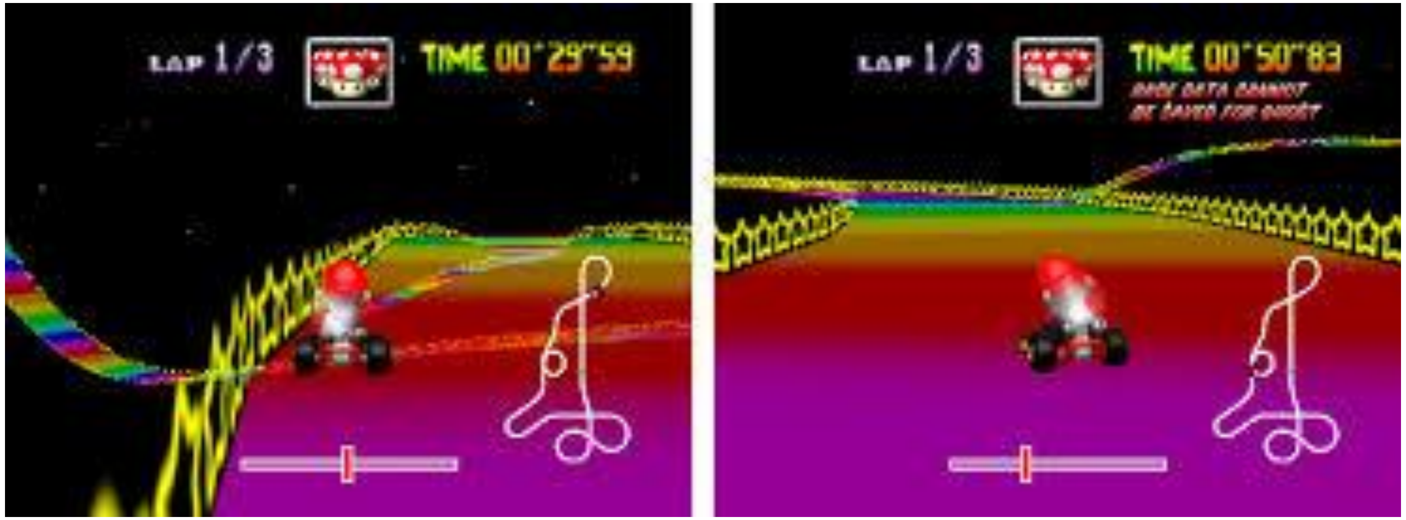
(General) Imitation Learning:

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

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!

IID Assumption
(Supervised Learning)

P^*

$\Downarrow$

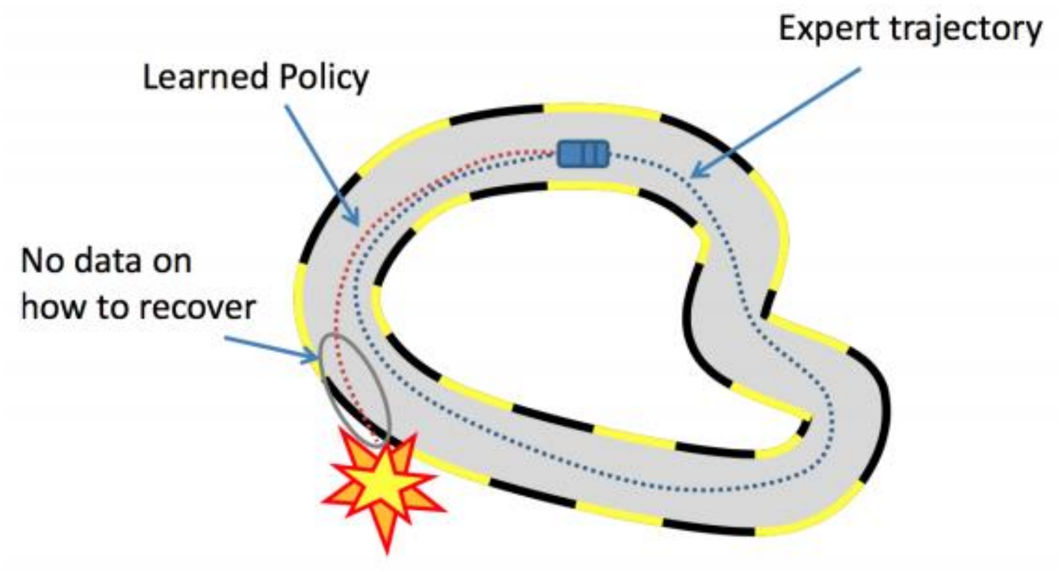
(s, a^*)

P_0

$\Downarrow$

$s \rightleftarrows \pi_\theta$

Limitations of Behavioral Cloning



Behavioral Cloning
Makes mistakes, enters new state
Cannot recover from new states

Lecture : Stanford cs 234

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

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

Works well when P^* close to P_θ

Inverse RL

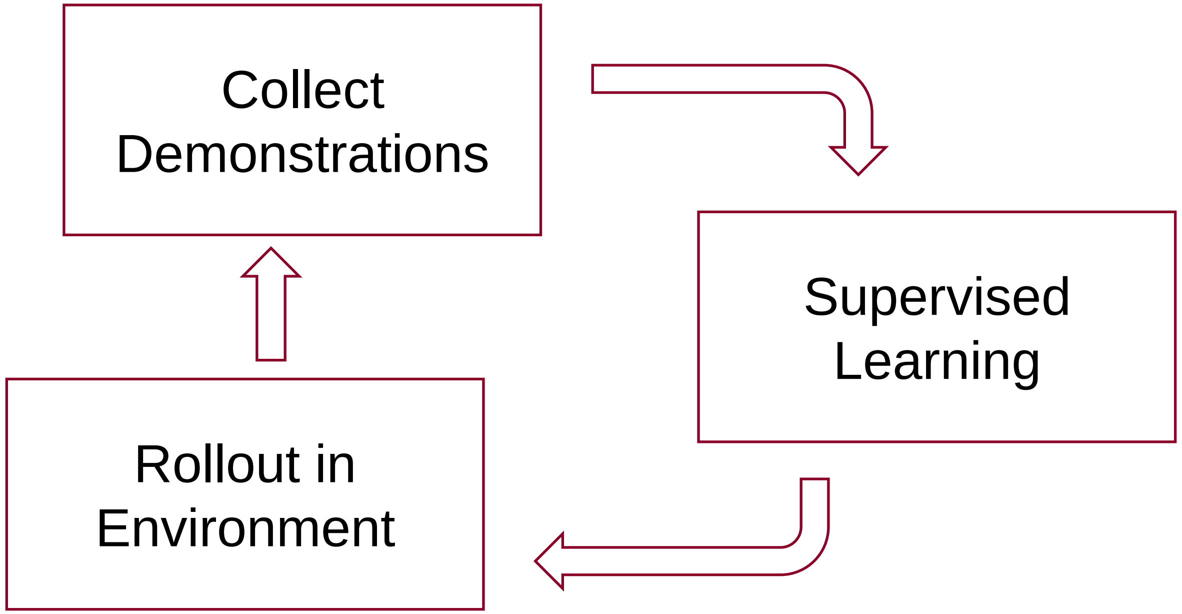
Learn r such that:

$$\pi^* = \underbrace{\mathrm{argmax}_{\theta}\, E_{s\sim P(s|\theta)} r(s, \pi_\theta(s))}_{\textbf{RL problem}}$$

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	Optional
Inverse Reinforcement Learning	No	Yes	Yes	No	Yes

Other Considerations

Choice of imitation loss (e.g., generative adversarial learning)

Suboptional demonstrations (e.g., outperform teacher)

Partial demonstrations (e.g., weak feedback)

Domain transfer (e.g., few-shot learning)

Structured domains (e.g., multi-agent systems, structured prediction)

Interactive Direct Policy Learning

Behavioral Cloning is simplest example

Beyond BC: using interacive demonstrator

Often analyzed via learning reductions

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

Learning Reductions

Behavioral Cloning:

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

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

Error on T-step trajectory is $O(T^2\varepsilon)$

*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]

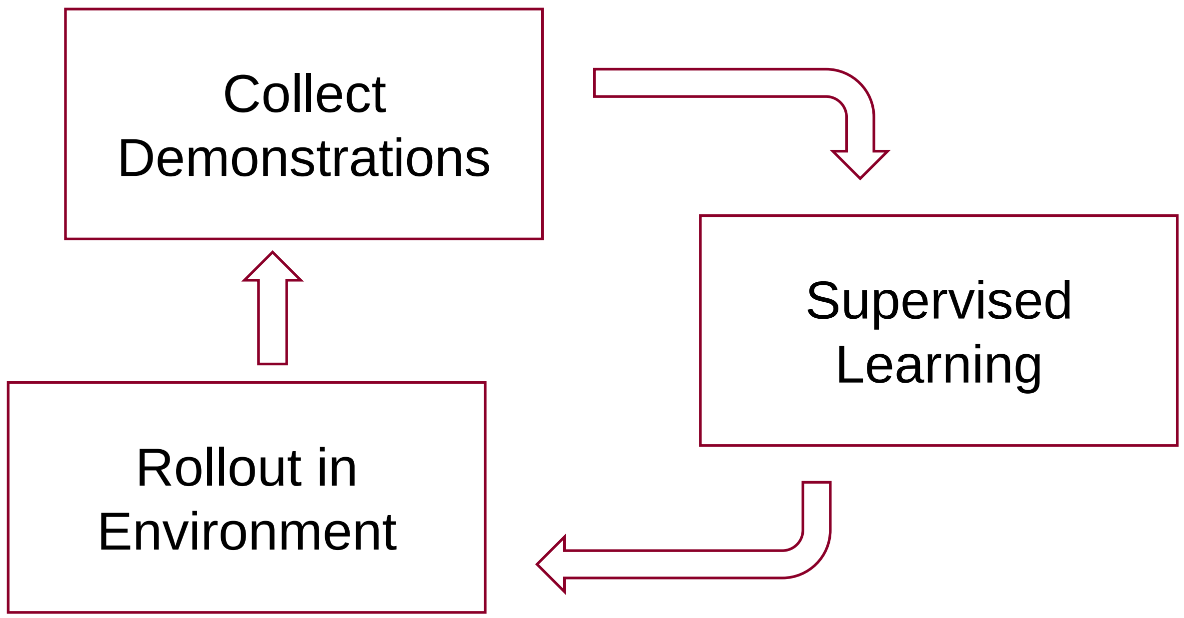
Direct Policy Learning vs Interactive Expert

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

Can query expert at any state

Steering
From expert

Construct loss function

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

Typically applied to rollout trajectories:

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

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

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

Images from Stephane Ross

Expert provides feedback on state visited by policy

Alternating Optimization (Naïve Attempt)

Steering
From expert

1. Fix P, estimate π
   - Solve

2. Fix π, estimate P
   - 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 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 $P_1 \cup \cdots \cup P_m$
- 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)

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

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

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

Best in policy class

"Online Regret"

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

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

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

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 $P_m \rightarrow \pi_m = \beta\pi'_m + (1-\beta)\pi_{m-1}$

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

π_0 **is expert**
(not available at test time)

π_m not much worse than π_{m-1}

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

π_M not much worse than π_0

π_0 negligible in π_M

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

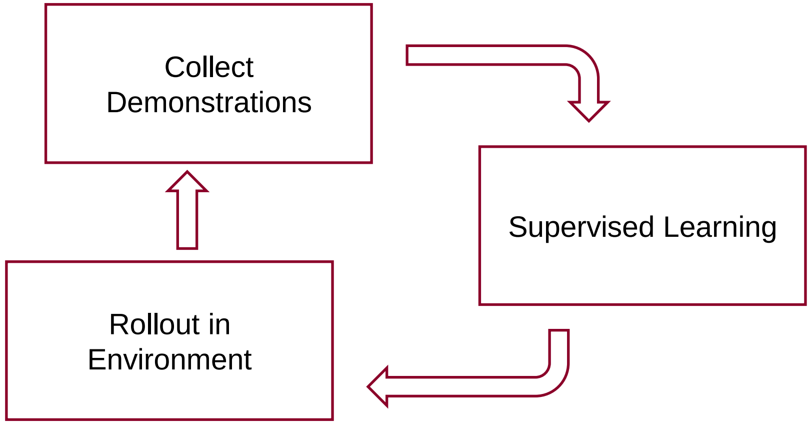
Two approaches: Data Aggregation & Policy Aggregation

- Ensures convergence
- Motivated by different theory

Not covered:

Depends on application

- What is expert feedback & loss function?



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 P_θ

Inverse RL

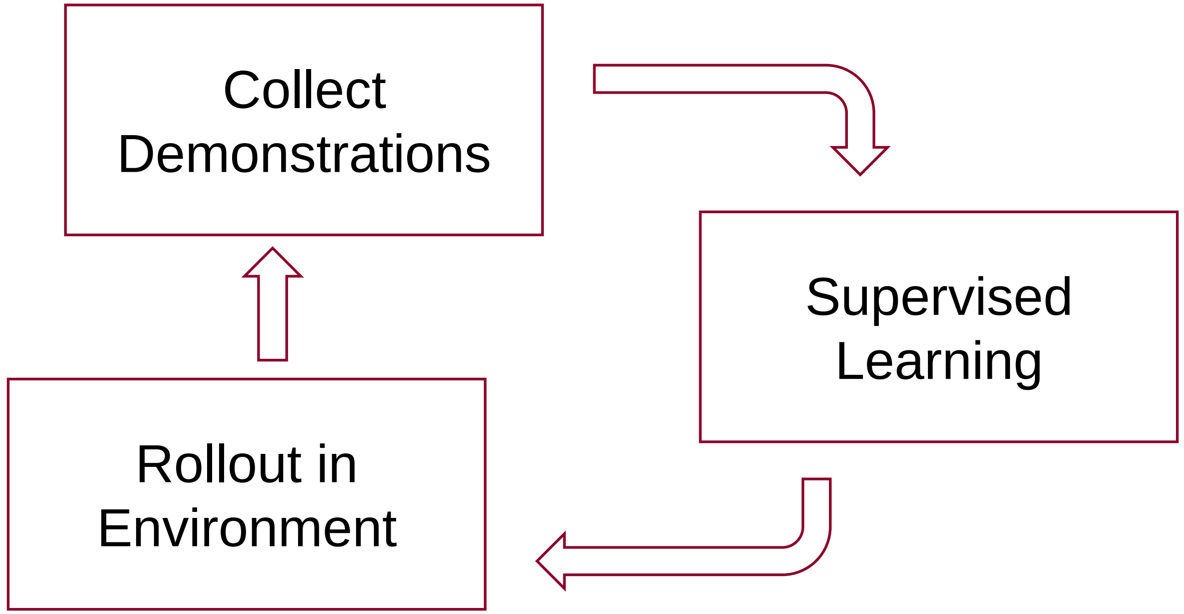
Learn r such that:

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

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)

Learning Policy w/o Expert: Reinforcement Learning

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

Starting state distribution

Reward function

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

Goal: maximize cumulative rewards

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

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

Fully specified MDP: value & policy iteration

Learning Policy w/o Expert: Reinforcement Learning

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

Optimize Value Function wrt Parameterized Policy π_θ

$$V(\pi) = \mathbb{E}_{s_0 \sim p_0}[\boldsymbol{cumulative\ rewards} | \pi_\theta]$$

$$Q^\pi(s,a) = \mathbb{E}_\pi[\boldsymbol{cumulative\ rewards} | \pi_\theta, s_0 = s, a_0 = a]$$

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

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

Learning Policy w/o Expert: Reinforcement Learning

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

<u>REINFORCE algorithm:</u>

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

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

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

Also check out…

- **Introduction to Reinforcement Learning with Function Approximation** – Richard Sutton – NIPS 2015 Tutorial
- **Deep Reinforcement Learning** – David Silver –ICML 2016 Tutorial
- **Deep Reinforcement Learning, Decision Making, and Control** – Sergey Levine & Chelsea Finn – ICML 2017 Tutorial
- **Deep Reinforcement Learning through Policy Optimization** – Pieter Abbeel & John Schulman – NIPS 2016 Tutorial

Challenges with Reward Engineering

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

assumed given

Youtube link

Inverse Reinforcement Learning

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

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

Goal: Learn a reward function r^* so that

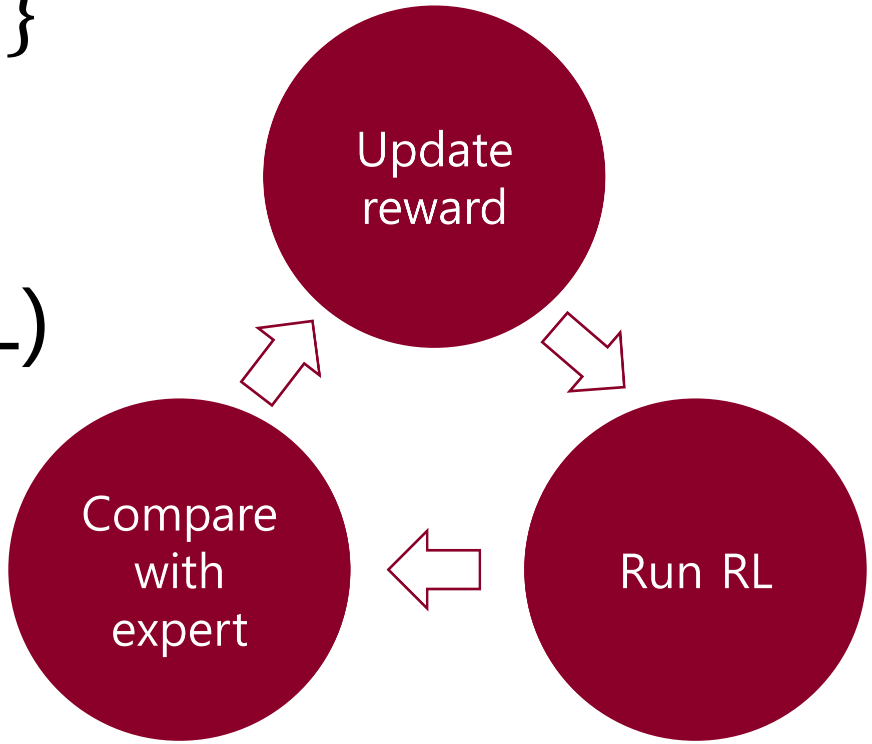
$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathbb{E}_\pi[r^*(s, a)]$$

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

Inverse Reinforcement Learning

- Inverse RL high-level recipe:

- Expert demonstrations: $\mathcal{D} = \{\tau_1, \cdots, \tau_m\}$
- Learn reward function: $r_\theta(s_t, a_t)$
- 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 $\boldsymbol{r}$

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

Different from "idealized" IRL

Game-Theoretic Inverse RL (model-given)

Syed & Schapire, NPS ‘07

Goal : find π performing better than $\boldsymbol{r}$

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

Different from “idealized” IRL

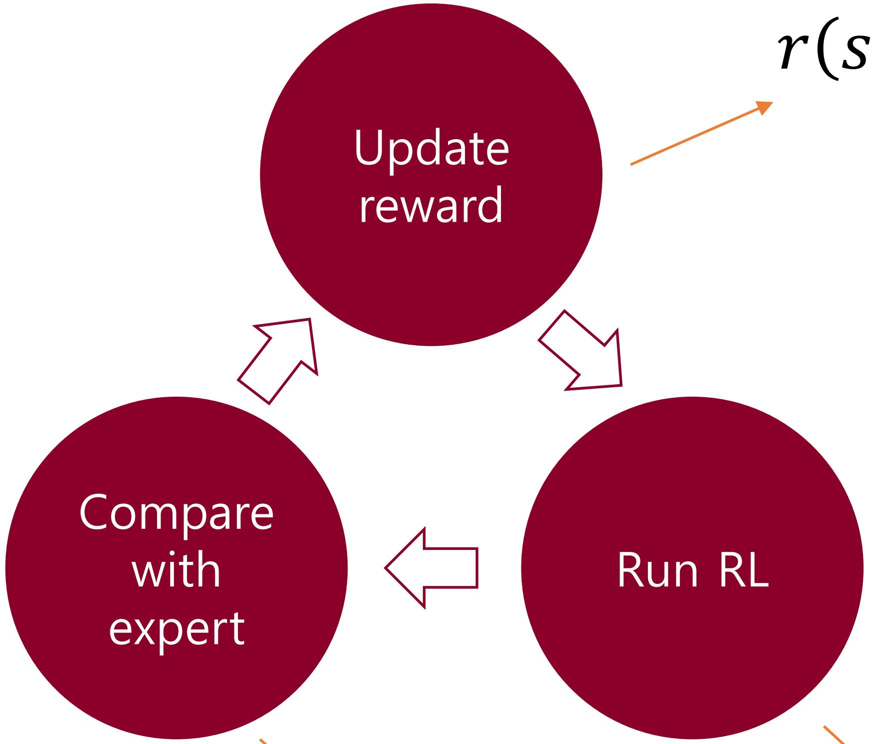
Imitation Learning via Inverse RL (model-given)

Abbeel & Ng, ICML ‘04

Syed & Schapire, NPS ‘07

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

$$r(s|\theta) = \theta^\top \phi(s)$$



Abbeel & Ng, ICML ’04 $\|\theta\|_2 \leq 0$

Syed & Schapire, NPS ’07 $\|\theta\|_1 = 1, \theta \succcurlyeq 0$

Known Dynamics

RL Oracle available

Linear Reward ⟶ Feature Expectations

Abbeel & Ng, ICML '04

Syed & Schapire, NPS '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 $\Rightarrow$ Optimality

Abbeel & Ng, ICML '04

Syed & Schapire, NPS '07

Feature Expectation:

$$\mu(\pi) = \mathbb{E}[\,\boldsymbol{visited\ state\ features}\ |\pi]$$

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

Find π s.t $\|\mu(\pi) - \mu(\pi^2)\|_2 \leq \epsilon$

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

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

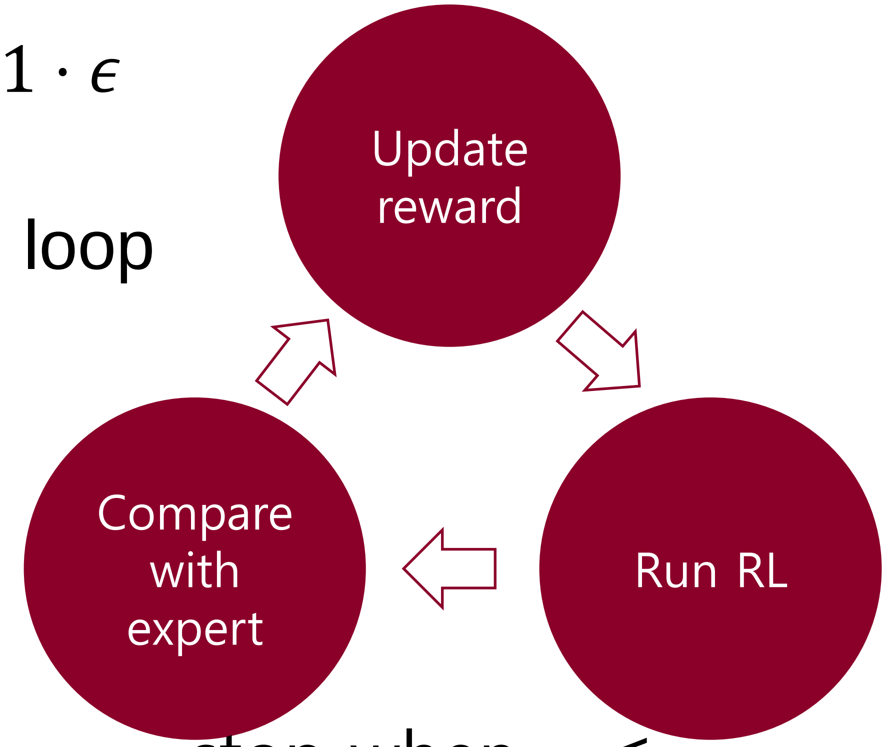
Algorithm: solve max-margin problem in each loop

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

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

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

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



stop when $\eta \leq \epsilon$

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

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

Maximum Entropy Principle

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

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

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

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 – Pys. Rev. 1957

Maximum Entropy Principle

Constrained Optimization $\xrightarrow{\text{Lagrangian}}$ Maximize Unconstrained Objective

$$r_\theta(s) = \theta^\intercal \phi(s)$$

$$\Rightarrow P(\tau|\theta) \propto e^{\tau^\intercal \phi(\tau)}$$

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

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

reward inference: max log likelihood

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

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)

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

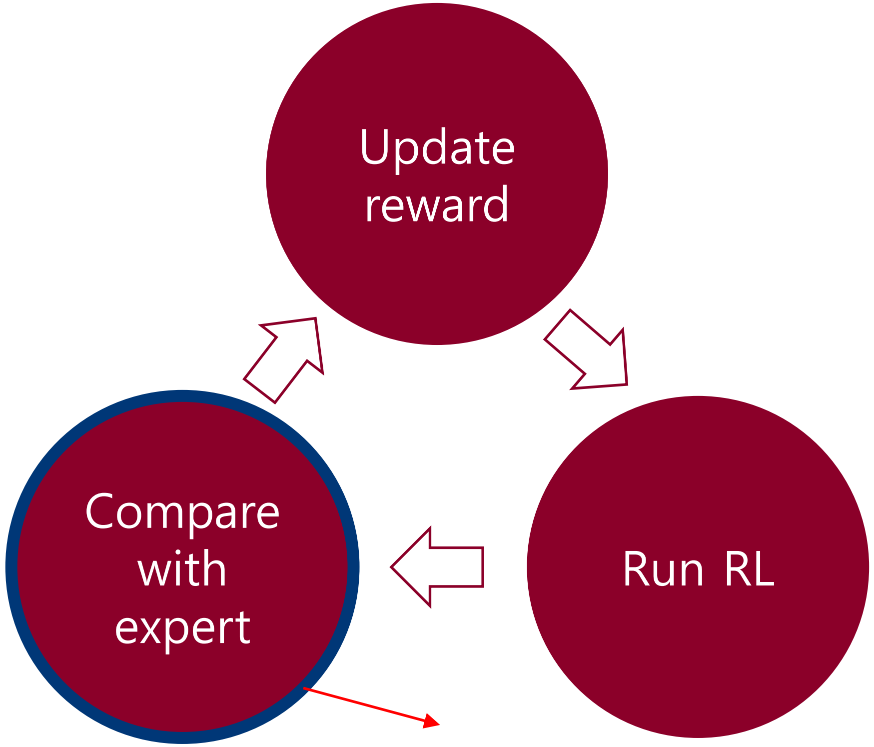
Solve forward RL $w.r.t.\ \theta$

Given from the model

Next…

<u>Model-given:</u>

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

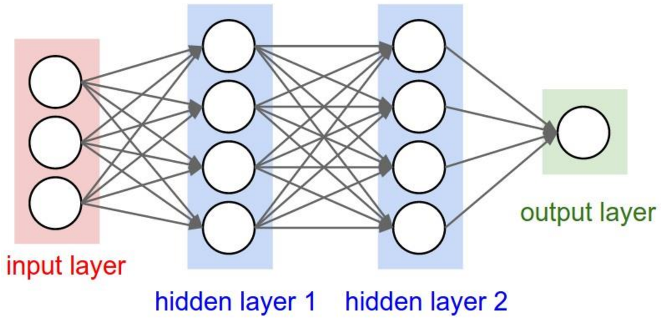


Known Dynamics

<u>Model-given:</u>

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

r_θ



Interact with environment / simulator (Model-free)

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

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

Max likelihood objective

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_{\tau_i \in \mathcal{D}} \log P(\tau_i|\theta)$$

Need to sample to evaluate gradients

$$= \arg\max_\theta \frac{1}{|\mathcal{D}|} \sum_{\tau_i \in \mathcal{D}} r_\theta(\tau_i) - \log Z(\theta)$$

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 $\mathcal{D}_{\text{samp}}$

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

Reward Optimization

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

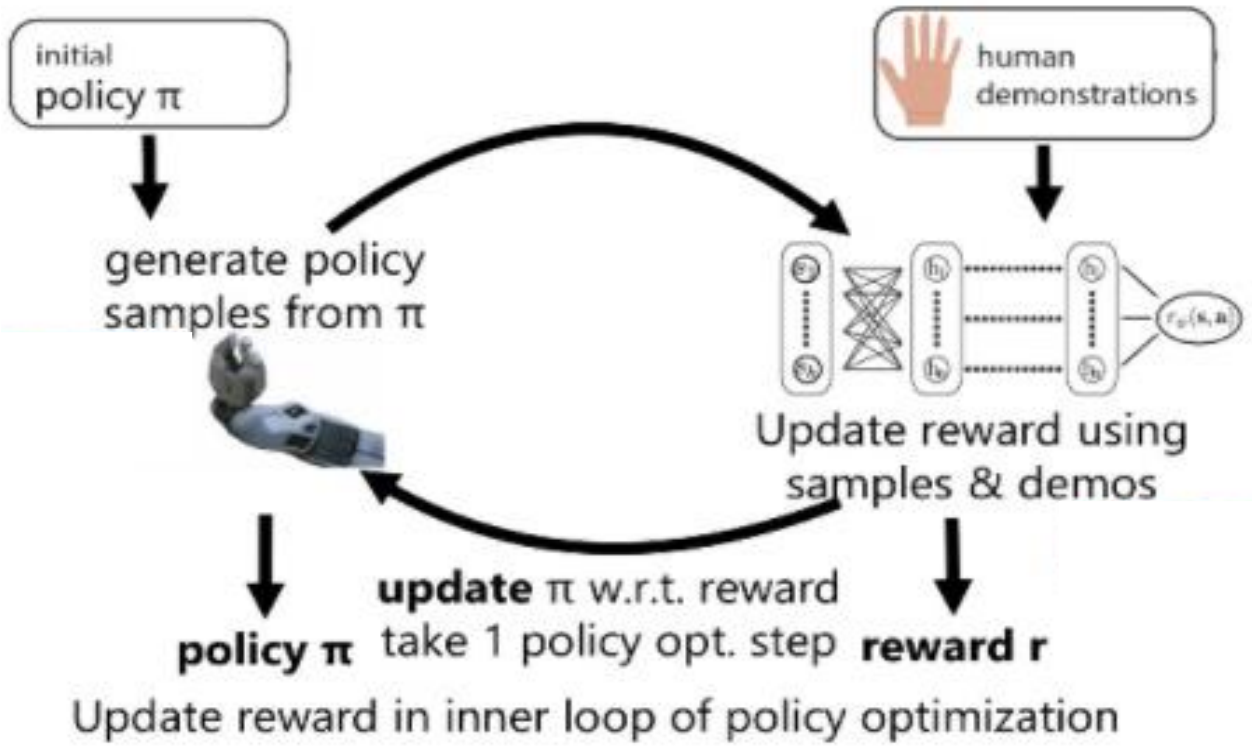
Policy Optimization

MaxEnt Deep IRL + Model-Free Optimization

Finn et al., ICML '16

- Generate samples by preventing the policy from changing too rapidly

- Update policy: take only 1 policy optimization step

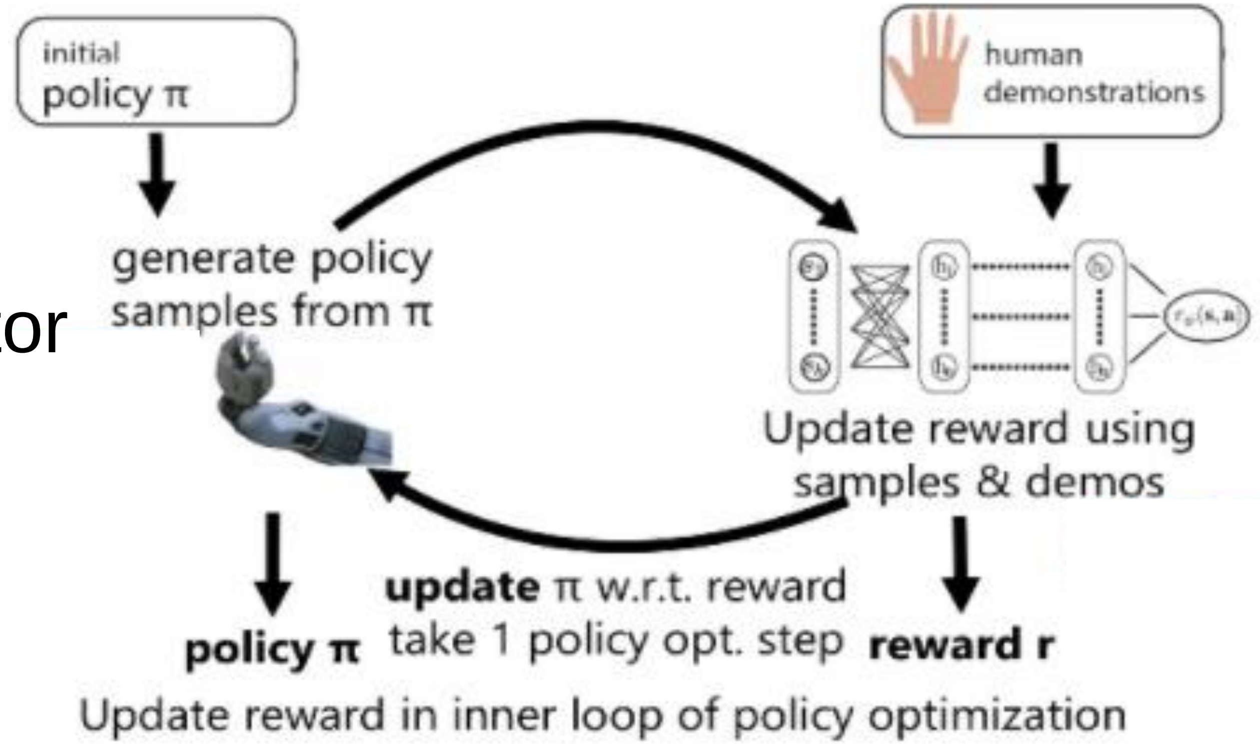


Figures from Chelsea Finn's presentation

Connection to Generative Adversarial Learning*

Finn et al., ICML '16

- Policy: Generator
- Reward Function: Discriminator



*Generative Adversarial Imitation Learning – Ho & Ermon - NIPS '16

IL via Occupancy Measure Matching

Apprenticeship Learning Using Linear Programming – Syed, Bowling & Schapire, ICML '08

Occupancy measure

$$d^{\pi}_{sa} = \text{visitation frequency of } (s,a) | \text{ following } \pi$$

$$d^{\pi}_{sa} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}(s_t = s, a_t = a) \,|\pi\right]$$

Fact:

$$V(\pi) = \mathbb{E}_{\pi}[r(s,a)] = \sum_{s,a} r(s,a) d^{\pi}_{sa}$$

IL Using Linear Programming (model-given)

Syed, Bowling & Schapire, ICML '08

Imitation learning $\Rightarrow$ occupancy measure matching

$$d_{sa}^{\pi} = d_{sa}^{\pi} \Rightarrow \mathbb{E}_{\pi}[r(s,a)] = \mathbb{E}_{\pi^*}[r(s,a)]$$

Direct imitation learning by solving dual LP of original MDP

No reward learning

Dual of IRL

Occupancy Measure Matching + MaxEnt

$$d_{sa}^{\pi} = d_{sa}^{\pi} \Rightarrow \mathbb{E}_{\pi}[r(s,a)] = \mathbb{E}_{\pi^*}[r(s,a)]$$

for any reward function $\boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a})$

Ill-posed occupancy matching $\longrightarrow$ MaxEnt principle again:

$$\max_{\pi} \quad H(\pi)$$
$$\text{s.t.} \quad \text{distance}(d^{\pi}, d^{*}) \leq \epsilon$$

Again, turning into unconstrained optimization:

$$\min_{\pi} \text{distance}(\boldsymbol{d}^{\boldsymbol{\pi}}, \boldsymbol{d}^{*}) - \boldsymbol{\lambda H}(\boldsymbol{\pi})$$

Occupancy Matching + Model-free Optimization

$$\min_{\pi} \text{distance}(\boldsymbol{d^{\pi}}, \boldsymbol{d^{*}}) - \boldsymbol{\lambda H}(\boldsymbol{\pi})$$

<u>Generative Adversarial Imitation Learning – Ho & Ermon, NIPS '16:</u>

$$\text{distance}(d^{\pi}, d^{*}) = \max_{\mathcal{D} \in (0,1)^{S \times A}} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi^{*}}\left[\log\left(1 - D(s,a)\right)\right]$$

$$\approx D_{KL}\left(d^{\pi} || \frac{(d^{\pi} + d^{*})}{2}\right) + D_{KL}\left(d^{*} || \frac{(d^{\pi} + d^{*})}{2}\right)$$

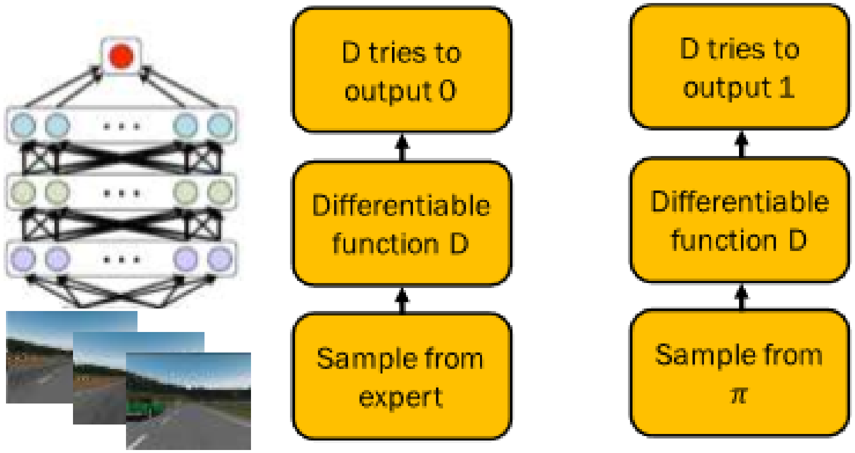
Jensen-Shannon divergence

Generative Adversarial Imitation Learning

Ho & Ermon, NIPS '16

Find saddle point (π, D)

$$\min_{\pi} \max_{\mathcal{D}\in(0,1)^{S\times A}} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi^*}\left[\log\left(1 - D(s,a)\right)\right] - \lambda H(\pi)$$



Generator Update

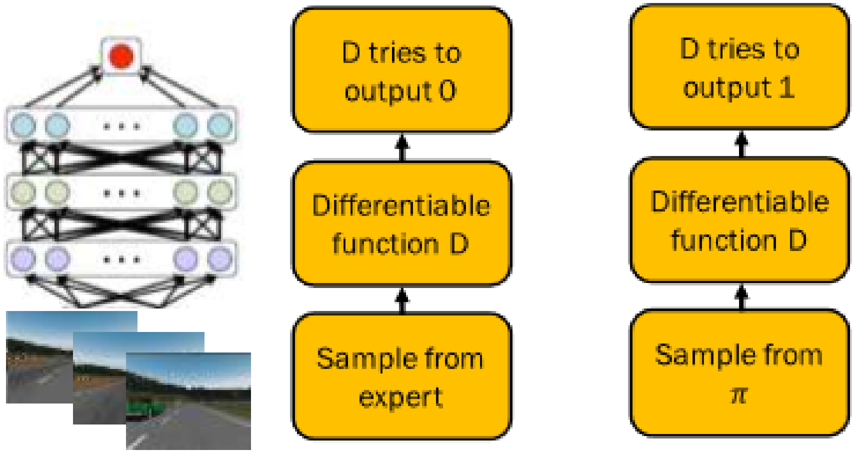
$$\widehat{\mathbb{E}}_{r_{\pi}}[\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s,a)] - \lambda \nabla_{\theta} H(\pi_{\theta})$$

Generative Adversarial Imitation Learning

Ho & Ermon, NIPS ‘16

Find saddle point (π, D)

$$\min_{\pi} \max_{\mathcal{D}\in(0,1)^{S\times A}} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi^*}\left[\log\left(1 - D(s,a)\right)\right] - \lambda H(\pi)$$



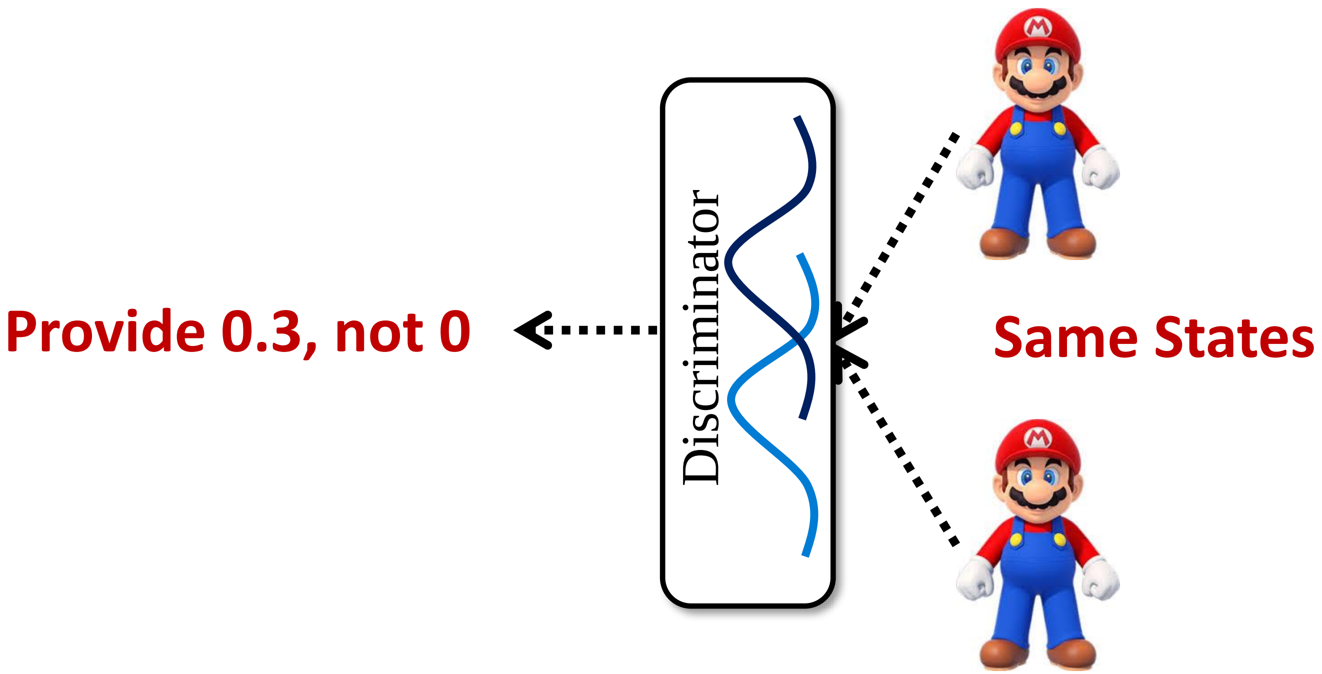
Generator Update

$$\hat{\mathbb{E}}_{r_\pi}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta)$$

Randomized Adversarial Imitation Learning

Shin IJCAI'19

The instability of the discriminator hampers the learning stability



Randomized Adversarial Imitation Learning

Shin IJCAI'19

Usually in AI:

$$f'(x) = \frac{df}{dx}$$

- To update the weights of policy the gradient descent method is used.

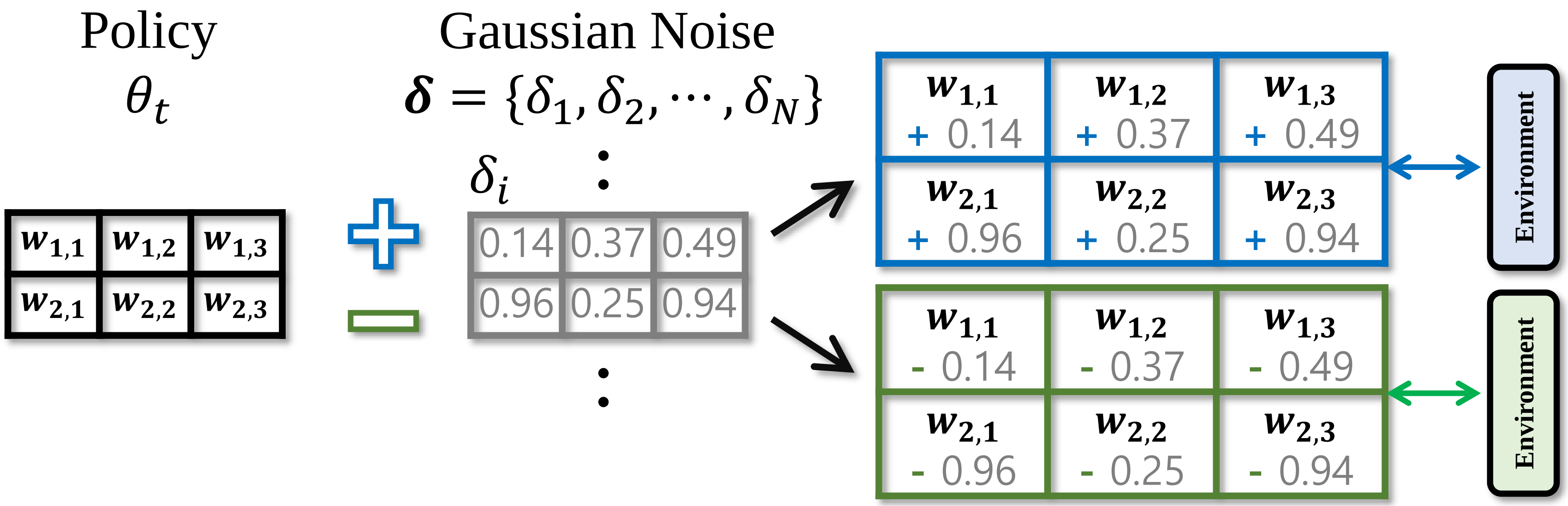
Proposed method

$$f'(a) = \frac{f(a+h) - f(a)}{h}$$

- To update the weights of policy **the method of finite differences.**

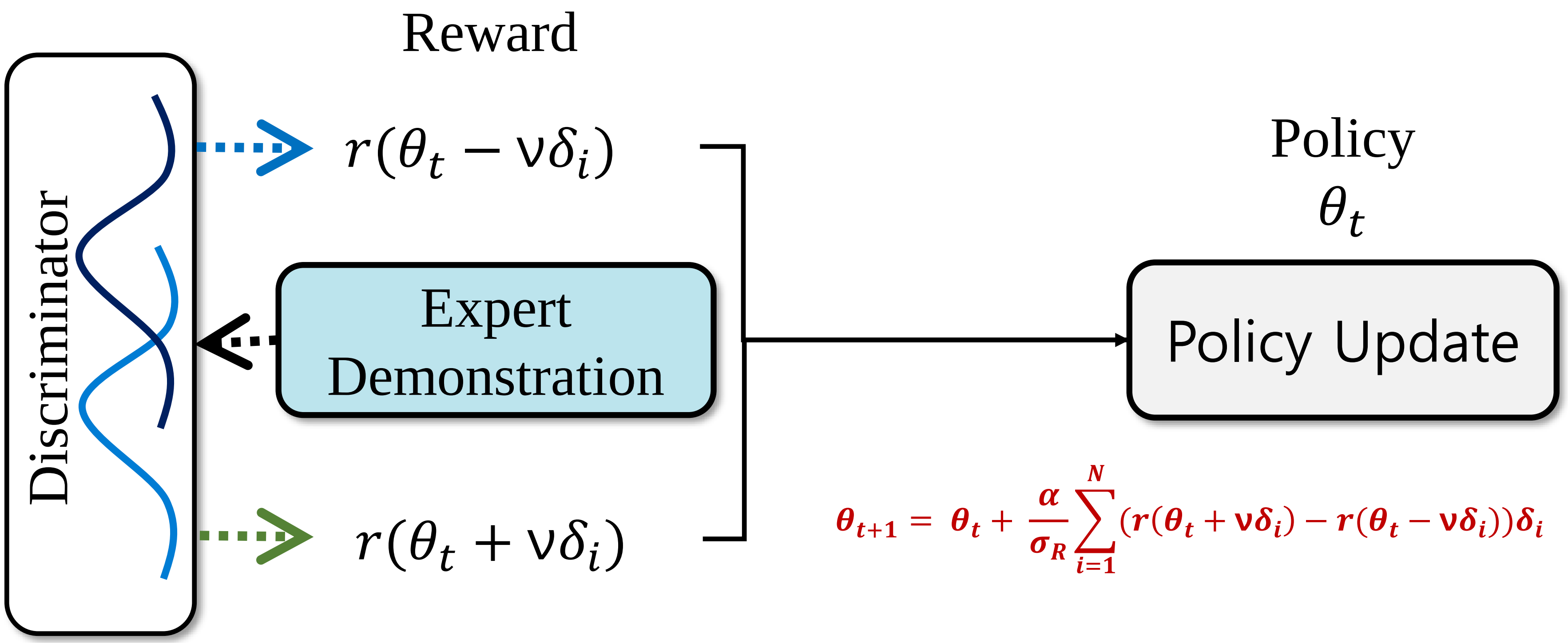
Randomized Adversarial Imitation Learning

Shin IJCAI‘19



Randomized Adversarial Imitation Learning

Shin IJCAI'19



$$\theta_{t+1} = \theta_t + \frac{\alpha}{\sigma_R}\sum_{i=1}^{N}(r(\theta_t + \nu\delta_i) - r(\theta_t - \nu\delta_i))\delta_i$$

List of Core Papers Mentioned

Search-based structured prediction – Daume, Langford & Marcu, Machine Learning 2009

A reduction of imitation learning and structured prediction to no-regret online learning – Ross, Gordon & Bagnell, AISTATS 2011

Efficient reductions for imitation learning - Ross, Gordon & Bagnell, AISTATS 2010

A reduction from apprenticeship learning to classification – Syed & Schapire, NIPS 2010

Apprenticeship learning via inverse reinforcement learning – Abbeel & Ng, ICML 2004

A game-Theoratic approach to apprenticeship learning – Syed & Schapire, NIPS 2007

Apprenticeship learning using linear programming – Syed, Bowling & Schapire, ICML 2008

Maximum entropy inverse reinforcement learning – Ziebart, Masss, Bagnell & Dey, AAAI 2008

Generative adversarial imitation learning – Ho & Ermon, NIPS 2016

Guided cost learning: deep inverse optimal control via policy optimization – Finn, Levine & Abbeel, ICML 2016

Speech Animation

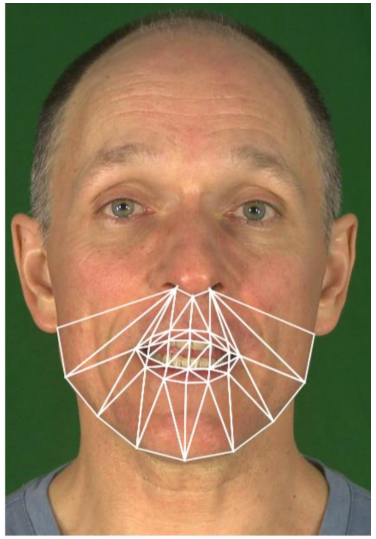
A Deep Learning Approach for Generalized Speech Animation

Input sequence $X = \langle x_1, \cdots, x_T \rangle$

Output sequence $Y = \langle y_1, \cdots, y_T \rangle \quad y \in R^D$

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

[Taylor et al., SIGGRAPH 2017]

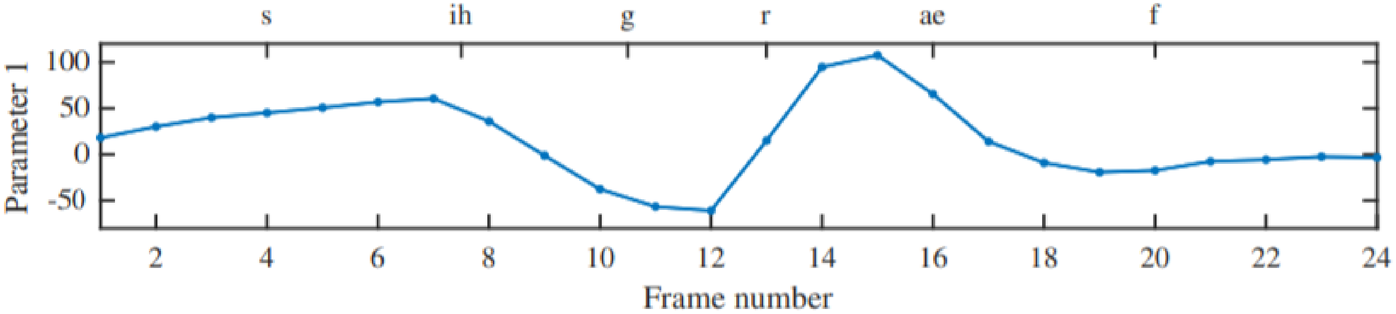


X

Frame	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Label	-	s	s	s	s	ih	ih	ih	g	g	g	r	r	ae	ae	ae	ae	f	f	f	f	-

Phoneme sequence

Y



Train using Behavioral Cloning

Sequence of face configurations

Structured Prediction
& Learning to Search

Structured Prediction

Learn mapping from structured space X to structured space Y

Typically requires solving optimization problem at prediction

- e.g., Viterbi algorithm for MAP inference in HMMs

Example: Sequence Prediction

Part-of-Speech Tagging

- Given a sequence of words x
- Predict sequence of tags y (dynamic programming)

$\boldsymbol{x}$ The rain wet the cat

$\boldsymbol{y}$ Det → N → V → Det → N

$\boldsymbol{y}$ Det → V → V → Adj → V

$\boldsymbol{y}$ Adv → Det → N → V → V

Goal: Minimize Hamming Loss

Goal: Learn Decision Function

(In Inference/Optimization Procedure)

- Sequentially predict outputs:

Interactive Feedback via Structured Labels

Supervised Learning on (s, a)

$\boldsymbol{x}$ The rain wet the cat

$\boldsymbol{y}$ Det → N → V → Det → N

$s_1 = (x, \emptyset)$

$s_2 = (x, \{Det\})$

$s_3 = (x, \{\mathrm{Det}, \mathrm{N}\})$

$\vdots$

$\pi(st) \rightarrow at \ \equiv yt$

State at t-th iteration Prediction

(Basic) Ingredients for Structured Prediction

- Deal with Distributional Drift (otherwise just use behavioral cloning)
  - Related approaches include scheduled sampling:
  - **Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks** [Bengio et al., NIPS 2015]
- Imitation Loss compatible w/ Structured Prediction Loss
  - Hamming Loss → 0/1 per-step loss (Hamming Loss is additive)
  - More generally: reduce to cost-sensitive classification loss
- Efficiency considerations
  - **Efficiently Programmable Learning to Search** [Daume et al., 2014]
  - **Deeply aggrevated: Differentiable imitation learning for sequential prediction** [Sun et al., 2017]

Learning Policies for Contextual Submodular Optimization

(Example of Cost-Sensitive Reduction)

Stephane Ross et al., ICML 2013

Context

Training set: (x, F_x) Monotone Submodular Function

Goal: learn π that sequentially maps x to action $set\ A$ to maximize F_x

Already selected actions

Learning Reduction: at state $s = (x, A)$ create cost for each action

- Define: $f_{max} = \max_a F_x(A + a) - F_x(A)$
- $c_s(a) = f_{max} - \left(F_x(A + a) - F_x(A)\right)$ Greedy Submodular Regret
- Supervised training example: (s, c_s)

Can prove convergence to $(1 - 1/e)$-optimal policy!

Sub-Optimal Expert Imitation Learning → Reinforcement Learning

- Query Access to Environment (Simple exploration)
  - **Learning to Search via Retrospective Imitation** [Song et al., 2018]
  - **Learning to Search Better than Your Teacher** [Chang et al., ICML 2015]
  - **Reinforcement and Imitation Learning via Interactive no-regret Learning** [Ross & Bagnell, 2014]
- More Sophisticated Exploration (also Query Access to Environment)
  - **Residual Loss Prediction: Reinforcement Learning with No Incremental Feedback** [Daume et al., ICLR 2018]
- Actor-Critic Style Approaches (also Query Access to Environment)
  - **Truncated Horizon Policy Search: Combining Reinforcement Learning and Imitation Learning** [Sun et al., ICLR 2018]
  - **Sequence Level Traveling with Recurrent Neural Networks** [Ranzato et al., ICLR 2016]

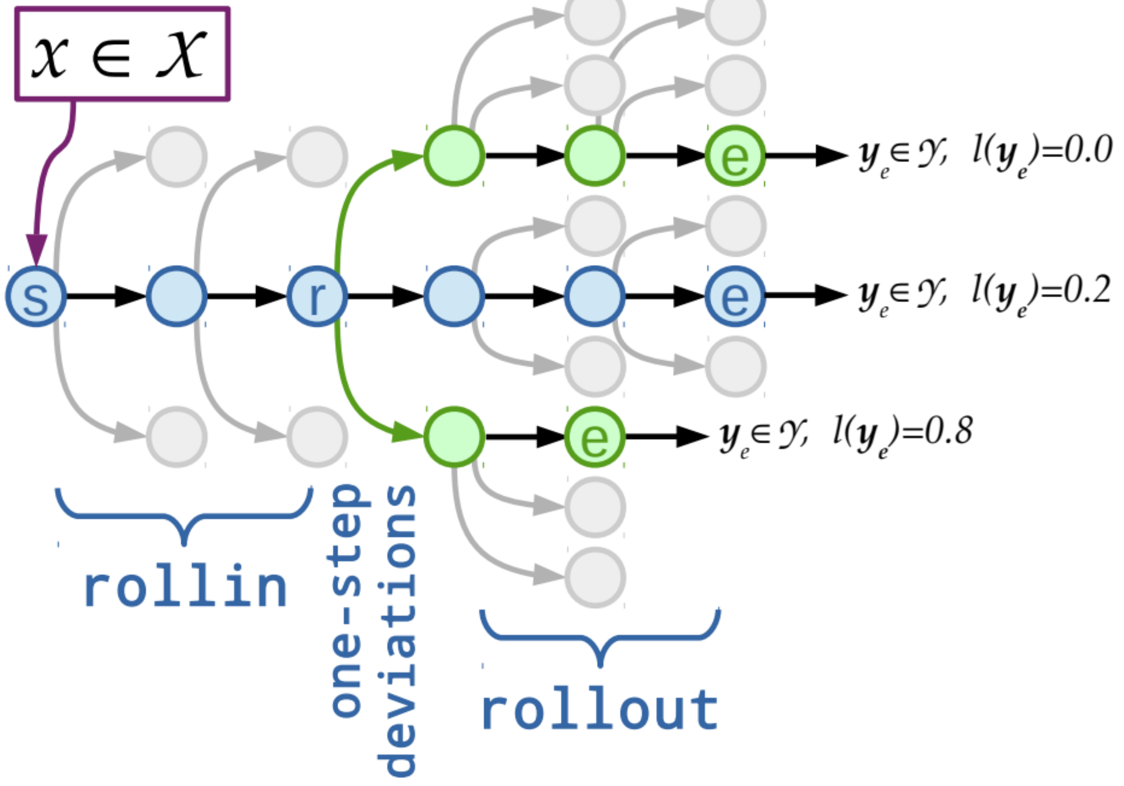
Learning to Search Better than Your Teacher

[Chang et al., ICML 2015]

Roll-in: execute policy w/o learning
Roll-out: execute policy for learning

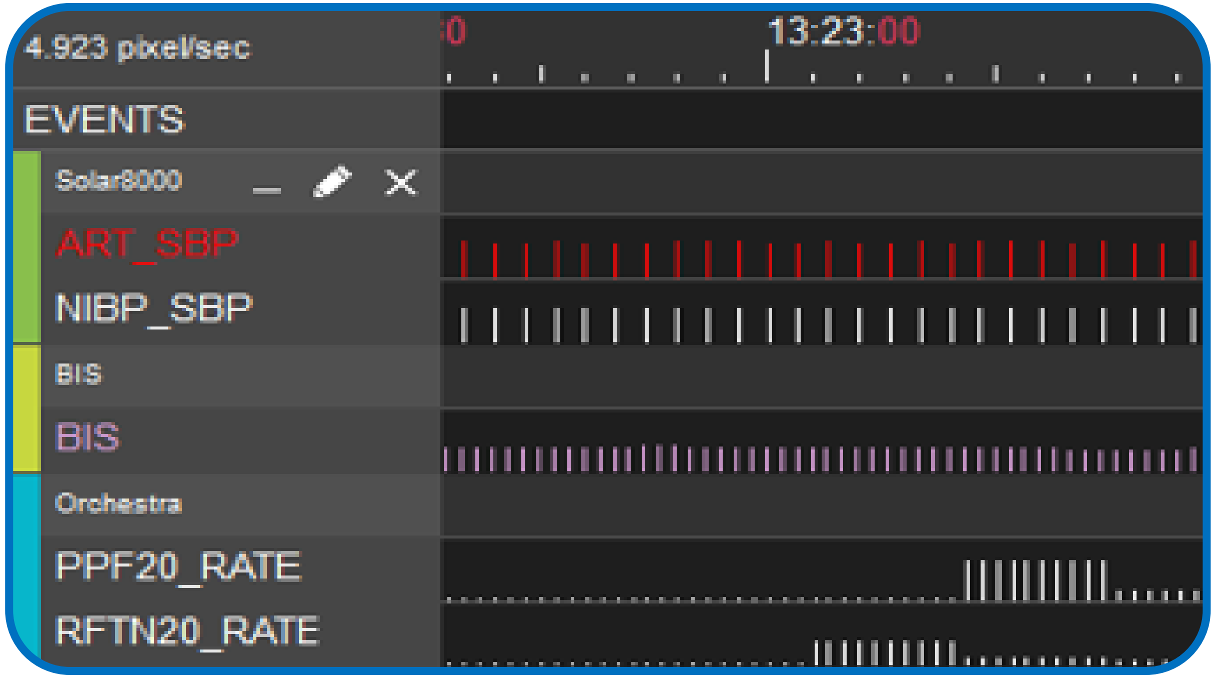
roll-out → / ↓ roll-in	**Reference**	**Mixture**	**Learned**
Reference	Inconsistent		
Learned	Not locally opt.	Good	RL

SEARN [Daume et al., 2009]



Anesthesiology & Imitation Learning

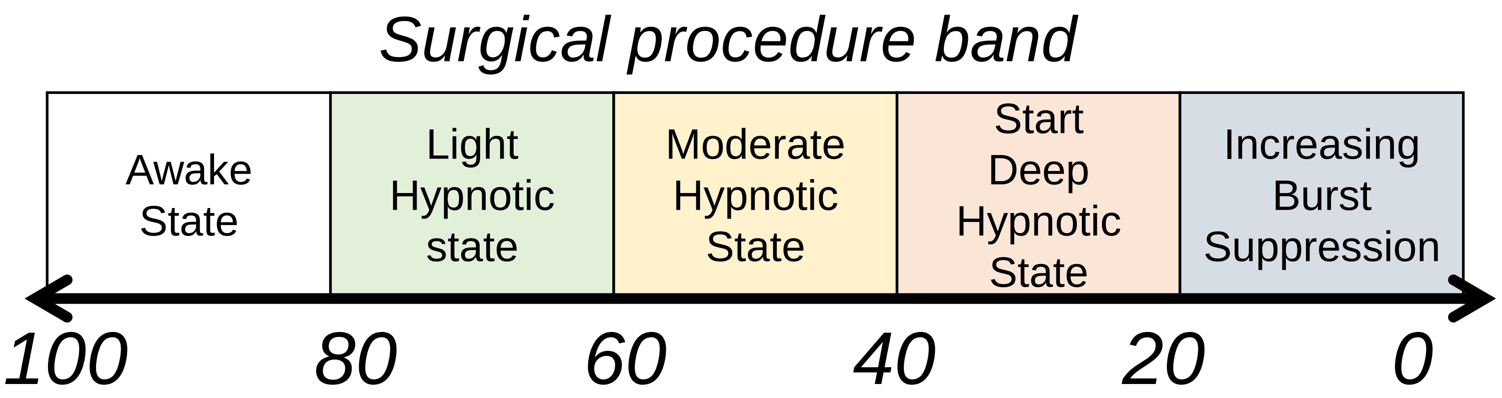
Propofol Infusion Control : Reinforcement Learning Approach



Vital Recorder

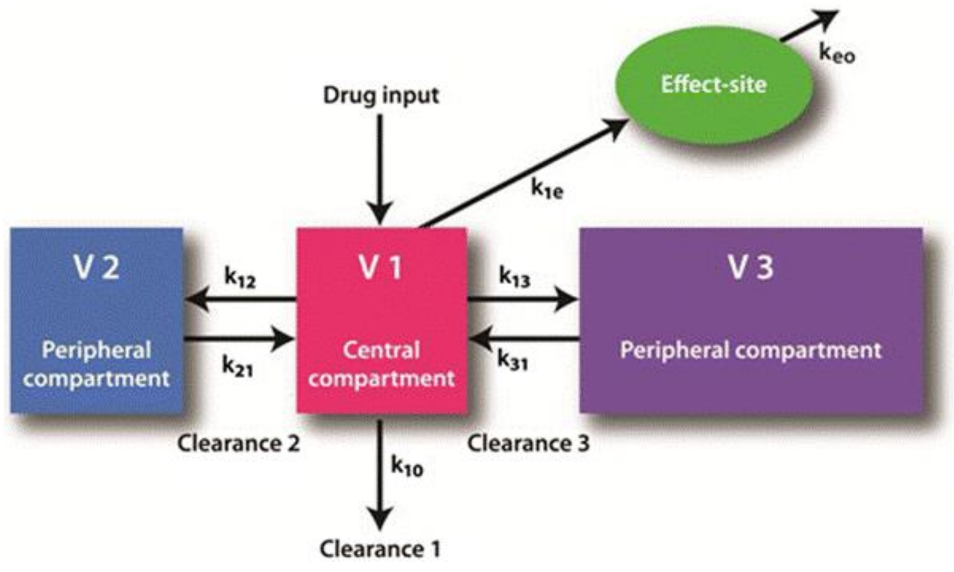
- Time-synchronised data captured from a variety of devices to facilitate an integrated analysis of vital signs data.

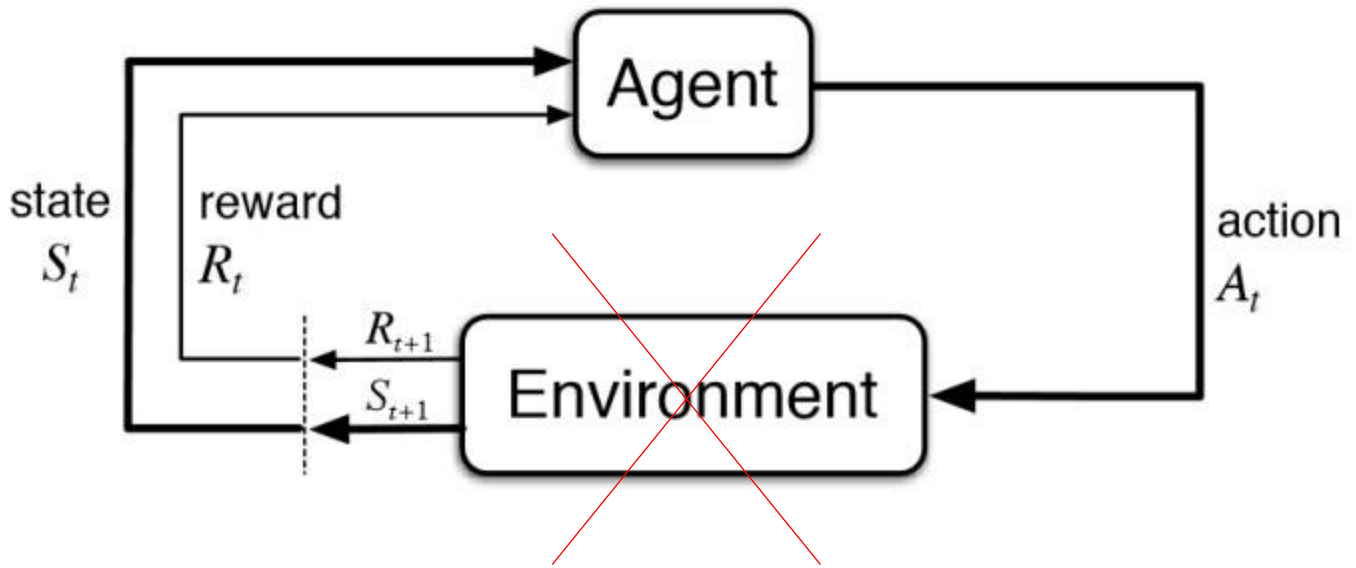
Propofol Infusion Control : Reinforcement Learning Approach



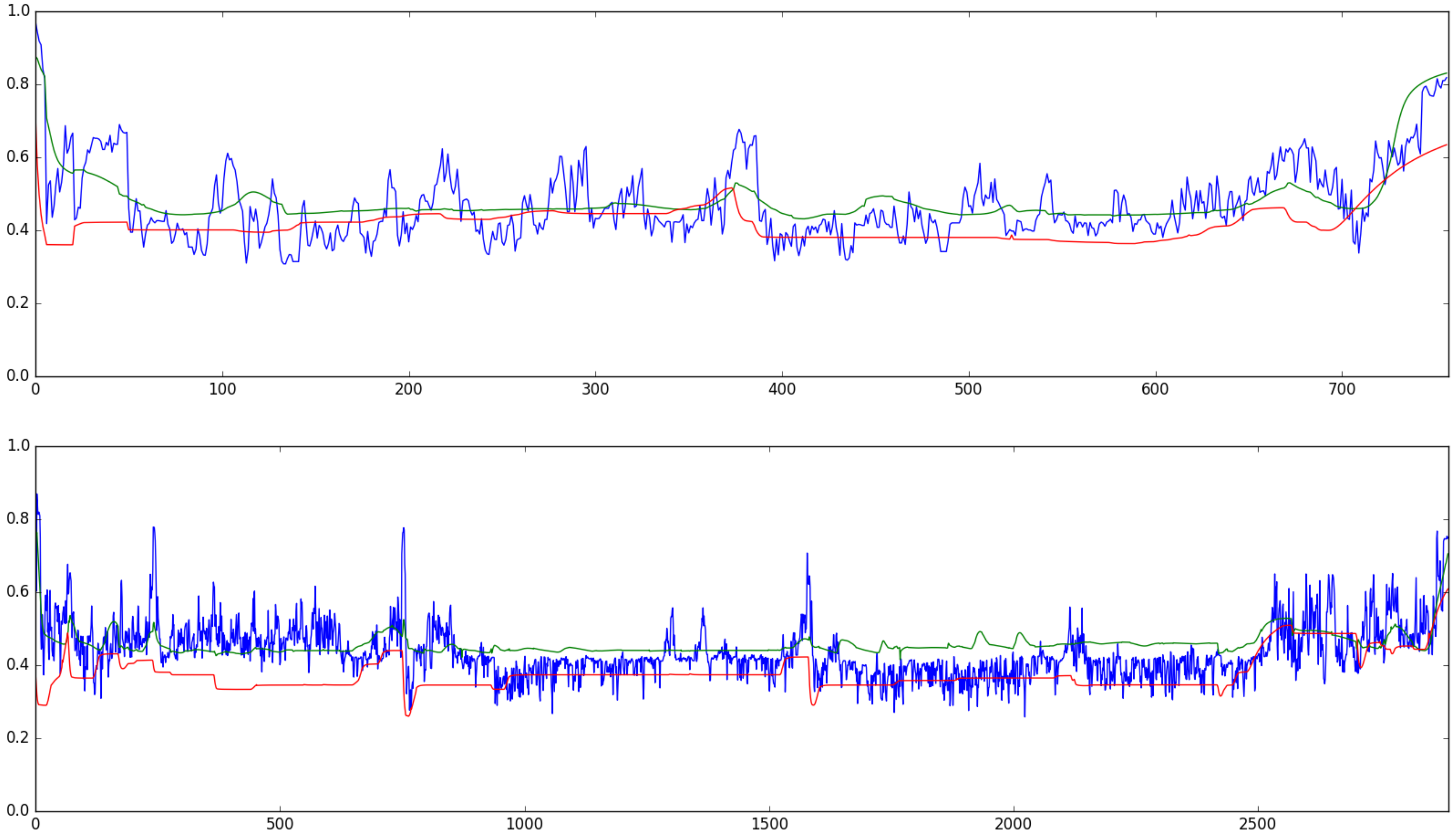
- Measure the brain activity and a desired depth of hypnosis
- The value of 100-90 corresponds to a fully awake state.
- The level of 90-60 and 60-40 indicate light and moderate hypnosis level, respectively.
- The level of deep hypnotic state (40-20) and increasing burst suppression (20-0) is significantly dangerous.

Propofol Infusion Control : Reinforcement Learning Approach

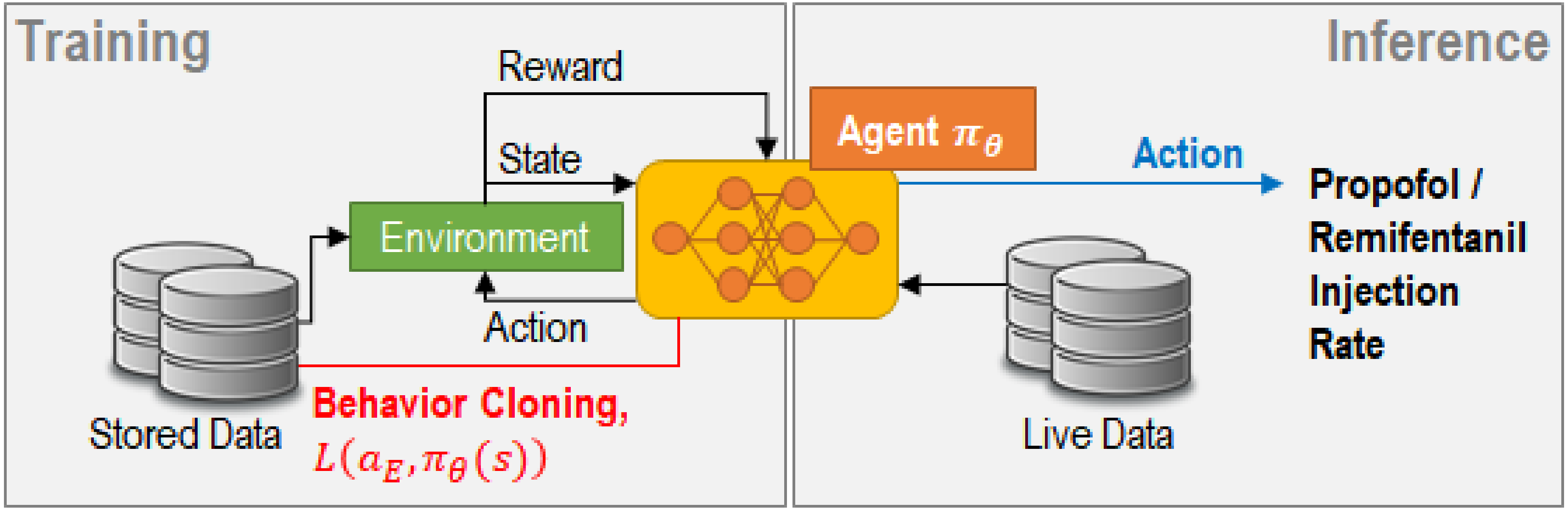




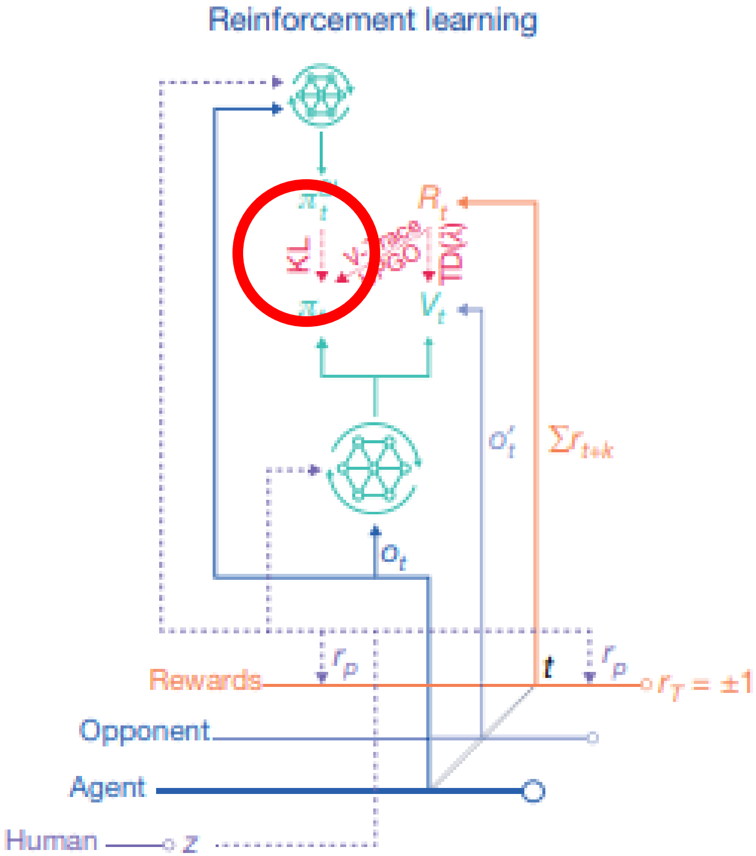
Propofol Infusion Control : Reinforcement Learning Approach



Propofol Infusion Control : Reinforcement Learning Approach

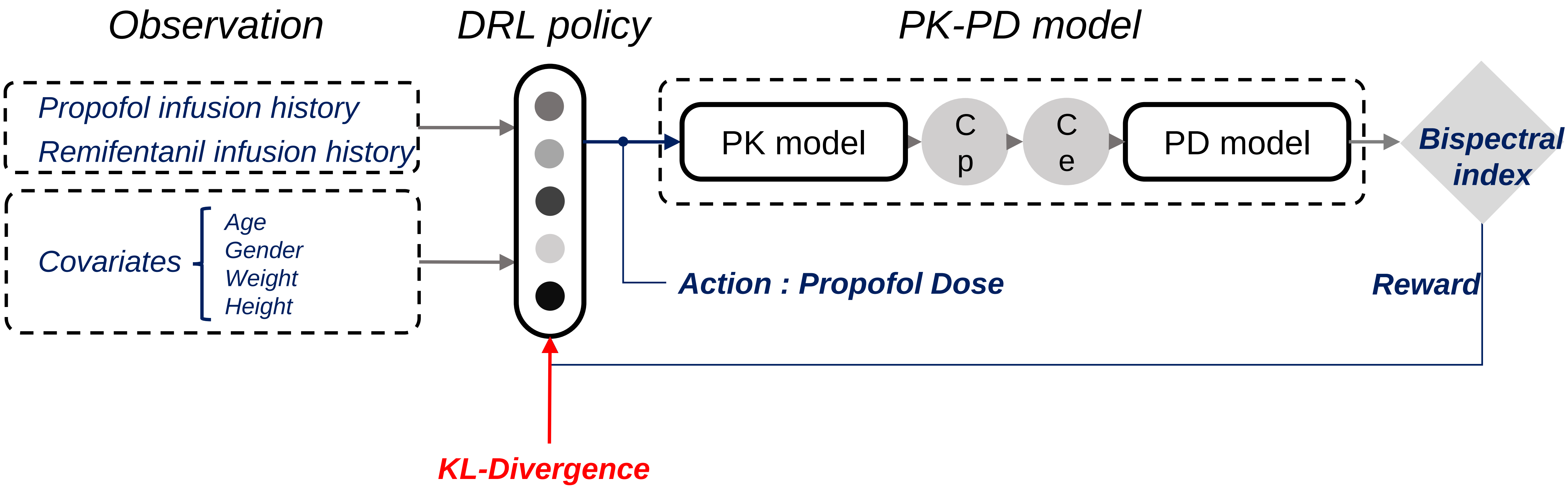


Propofol Infusion Control : Reinforcement Learning Approach



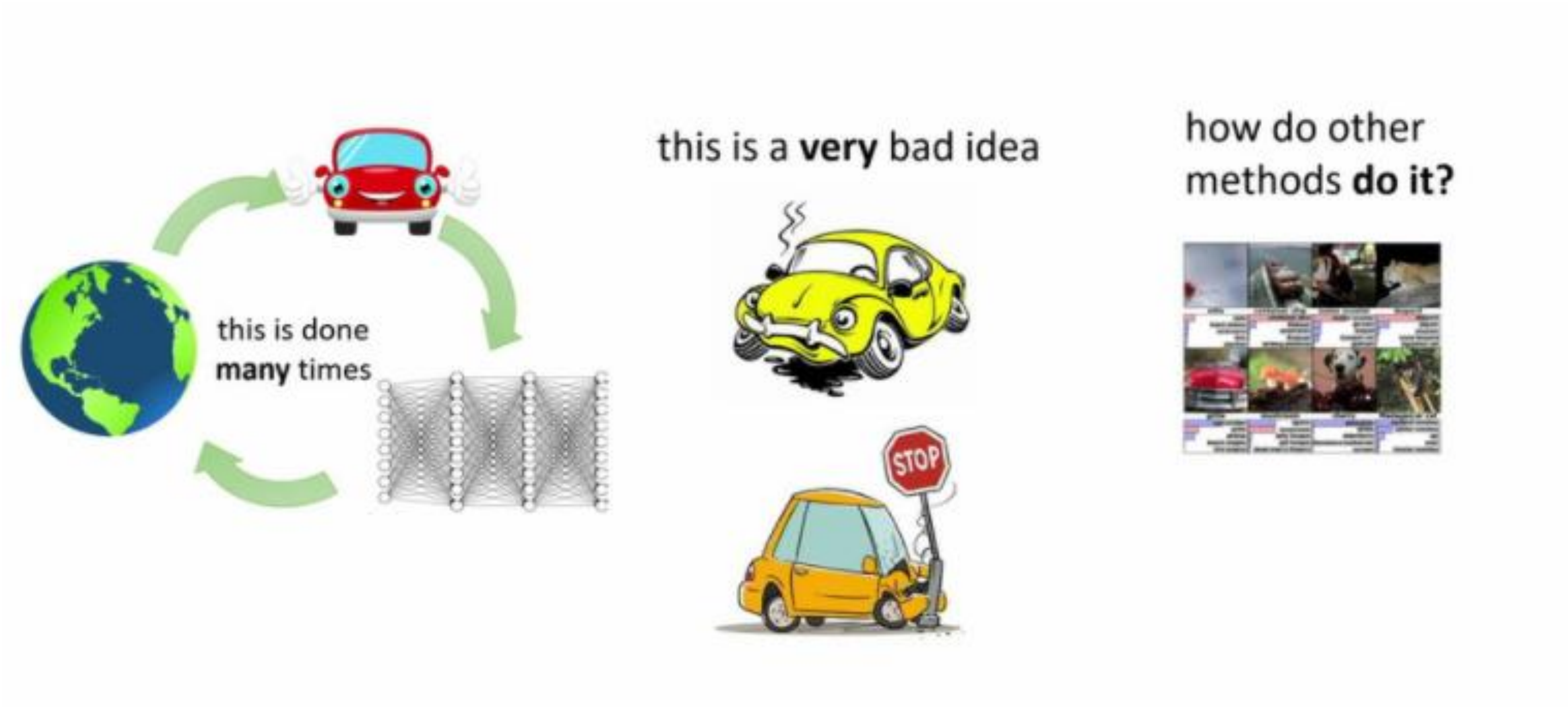
- Agent parameters were initially trained by supervised learning.
- The agent parameters were subsequently trained by a reinforcement learning algorithm.
- By using KL-Divergence, decision making of agent continuously follows that of expert.

Propofol Infusion Control : Reinforcement Learning Approach

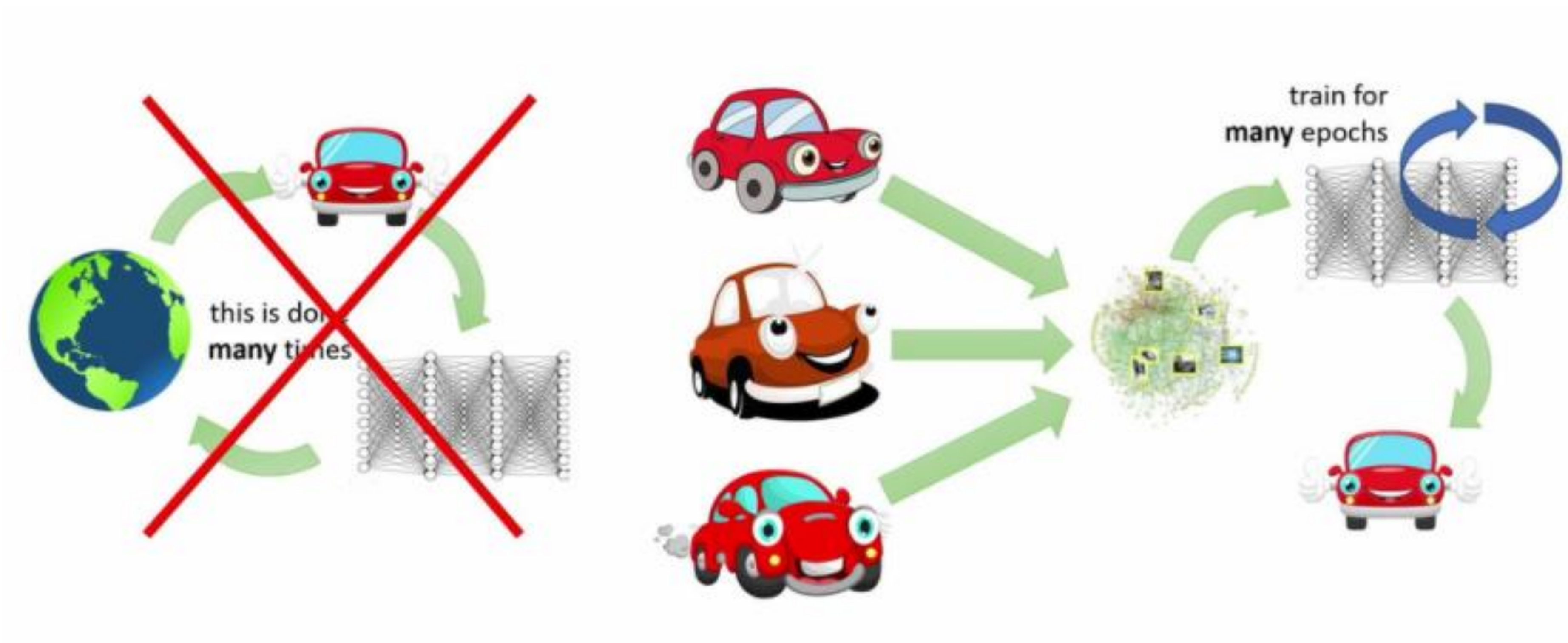


Autonomous Driving

Autonomous Driving with Imitation Learning



Autonomous Driving with Imitation Learning



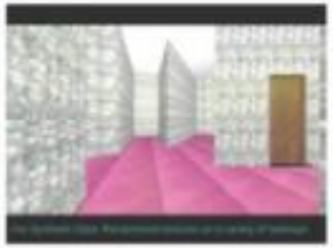
Autonomous Driving with Imitation Learning

the pipeline

observations → state estimation (e.g. vision) → modeling & prediction → planning → low-level control → controls

- very manual
- limited capacity to improve ← hand-designed components will eventually become the bottleneck once the robot collects enough data
- we just don't know the right abstractions

the sim-to-real

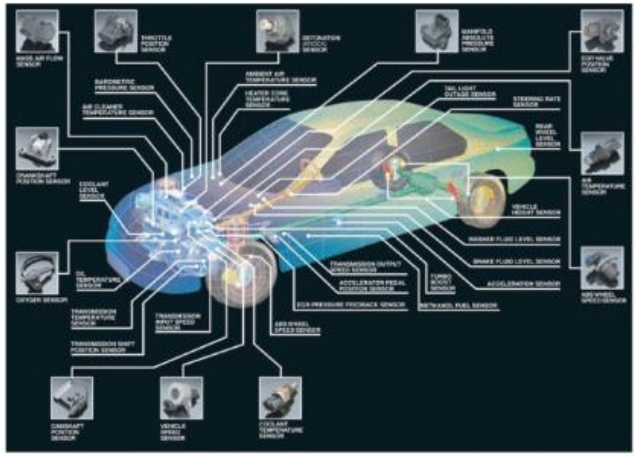


Sadeghi & Levine '16

"domain randomization"

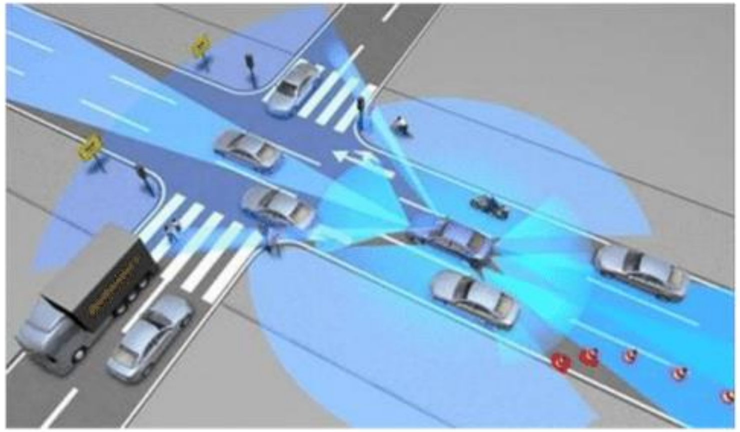
Autonomous Driving with Imitation Learning

Vehicle Sensors



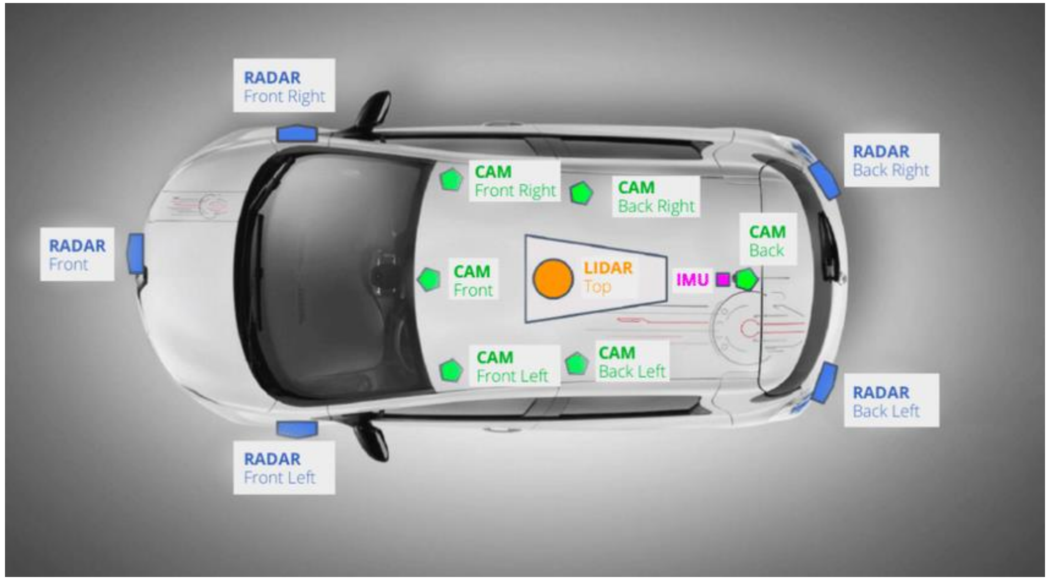
- Various sensors for vehicle
  (e.g. LIDAR, RADAR, ⋯)
- High performance
- Sensor fusion techniques

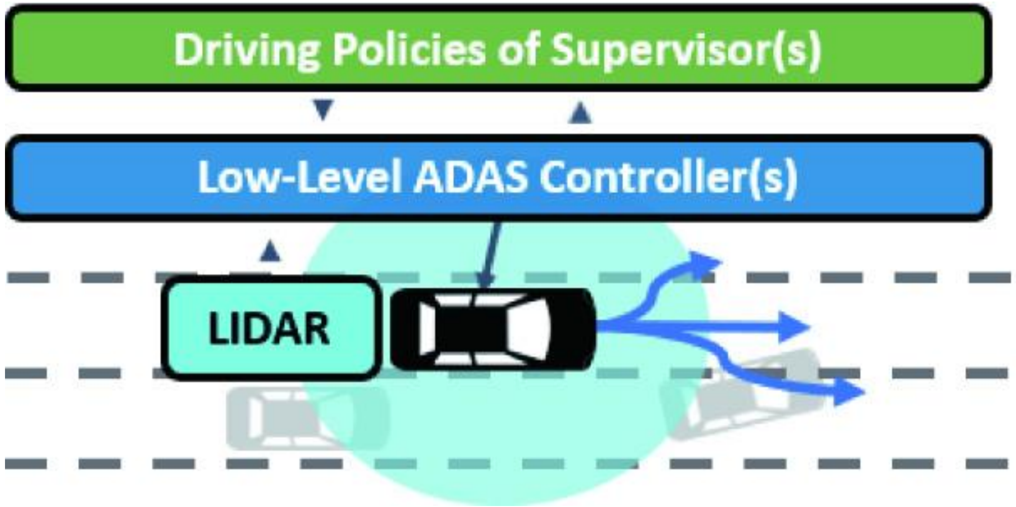
ADAS Algorithms



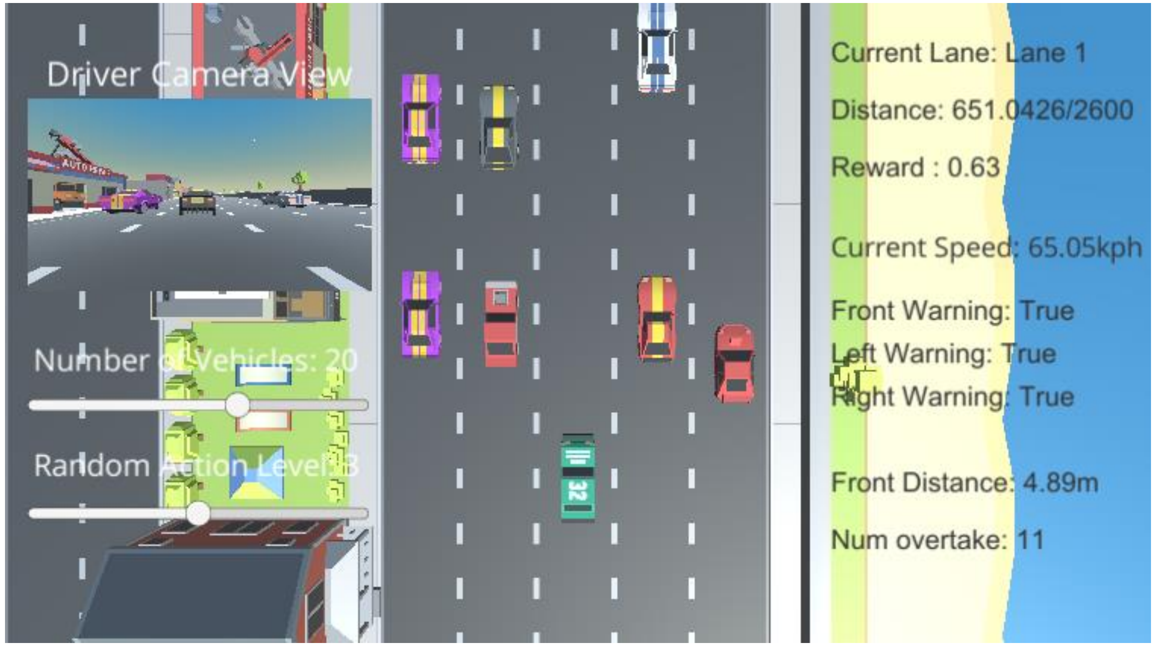
- Various ADAS
  (e.g. AEB, LKAS, BSD, ESC, ⋯)
- Already commercialized
- Essential function for safety

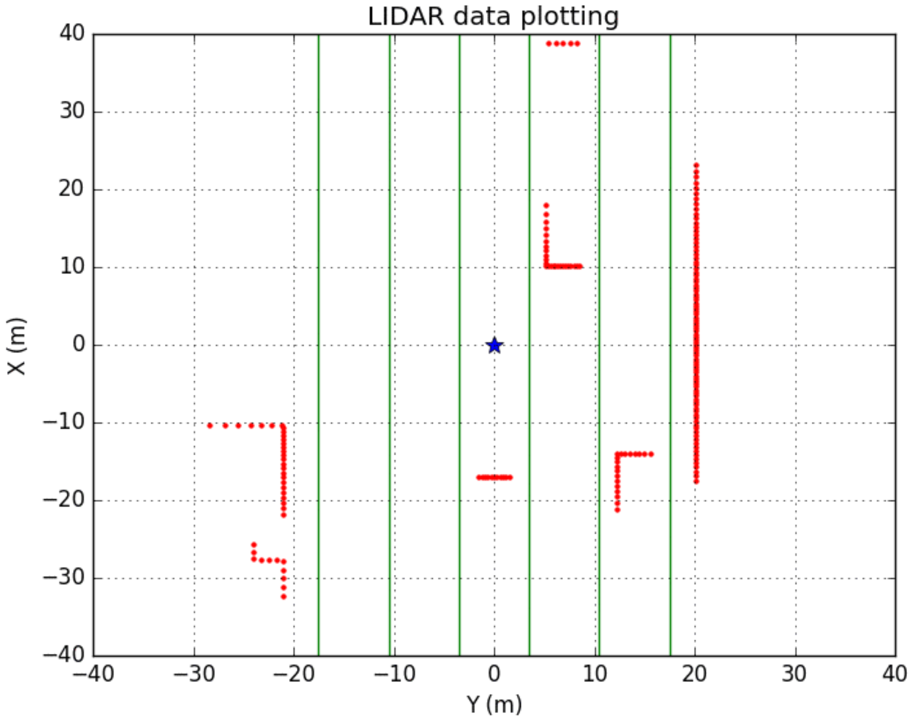
Autonomous Driving with Imitation Learning



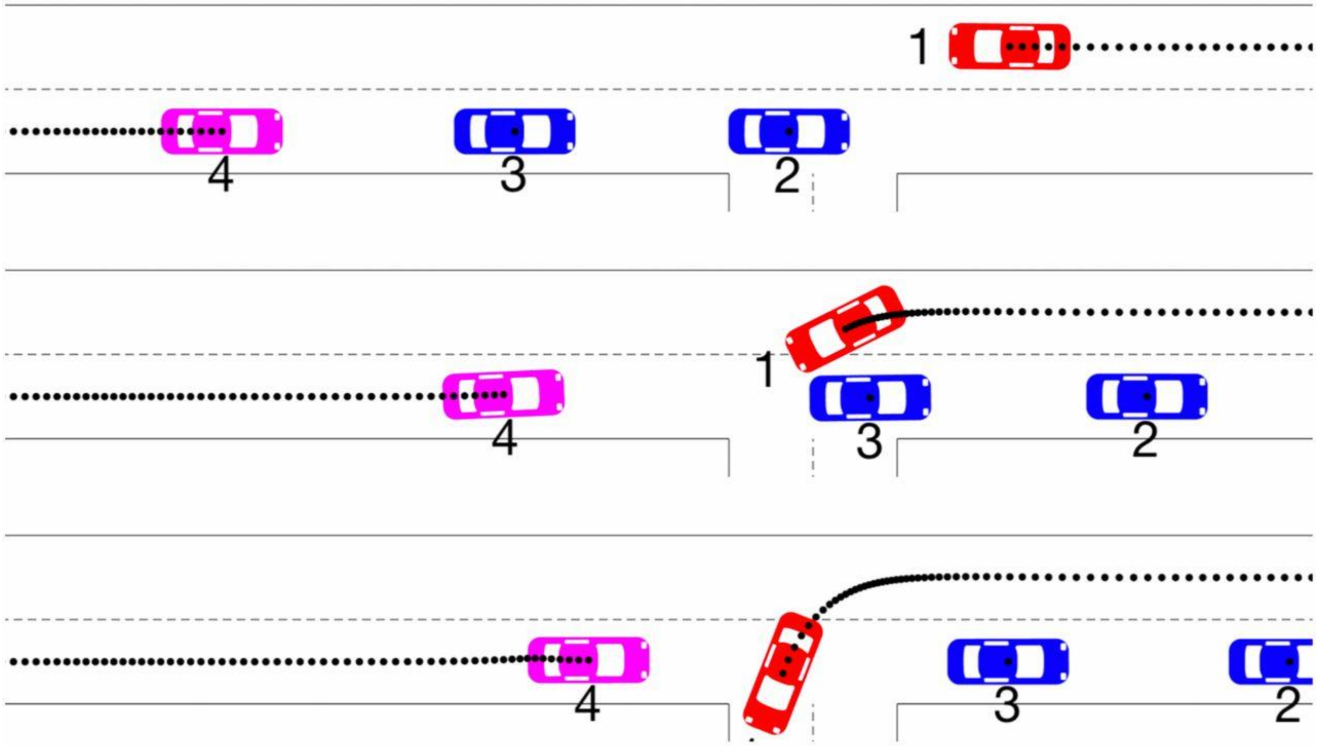


Autonomous Driving with Imitation Learning

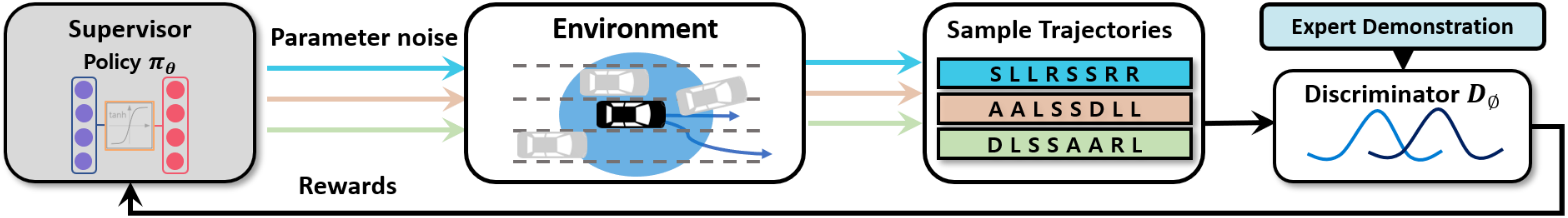




Autonomous Driving with Imitation Learning

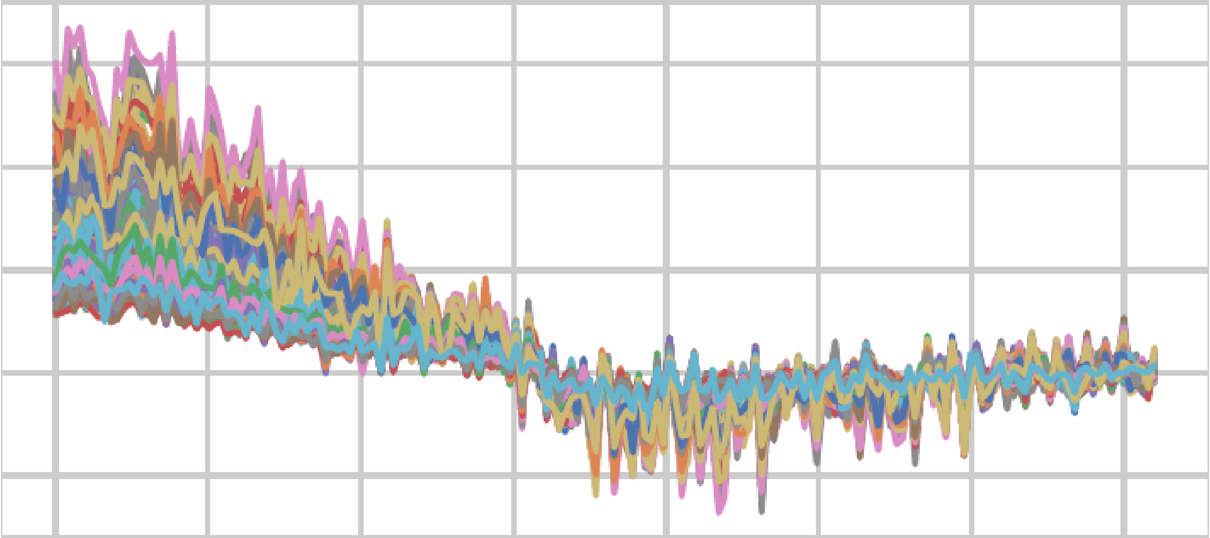


Autonomous Driving with Imitation Learning

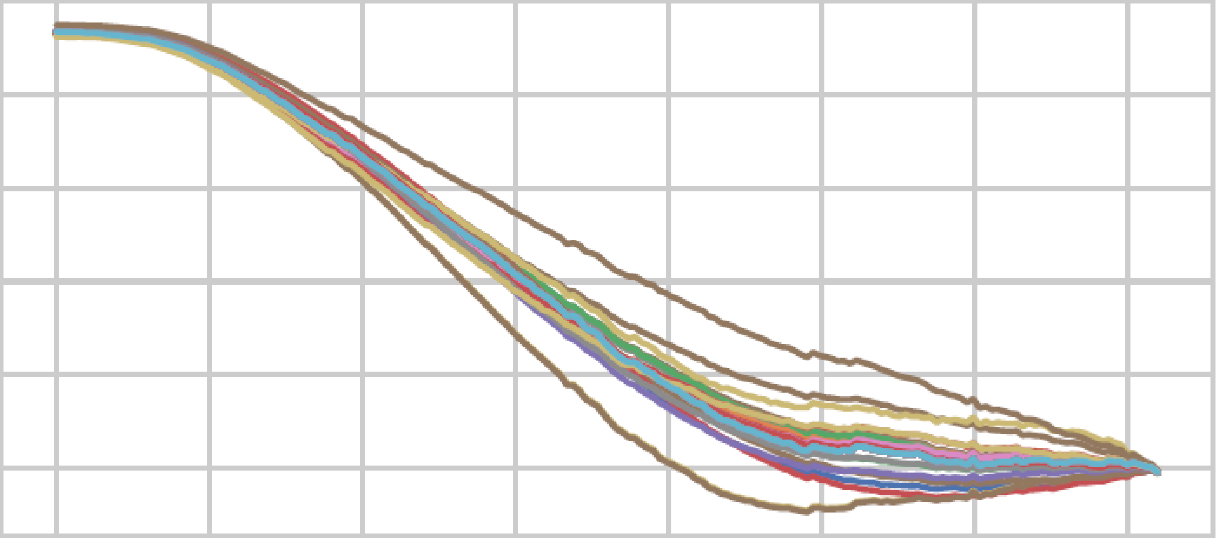


$$minimize \;\; \mathbb{E}_{\pi}[\log(D(s,a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a)]$$

Autonomous Driving with Imitation Learning

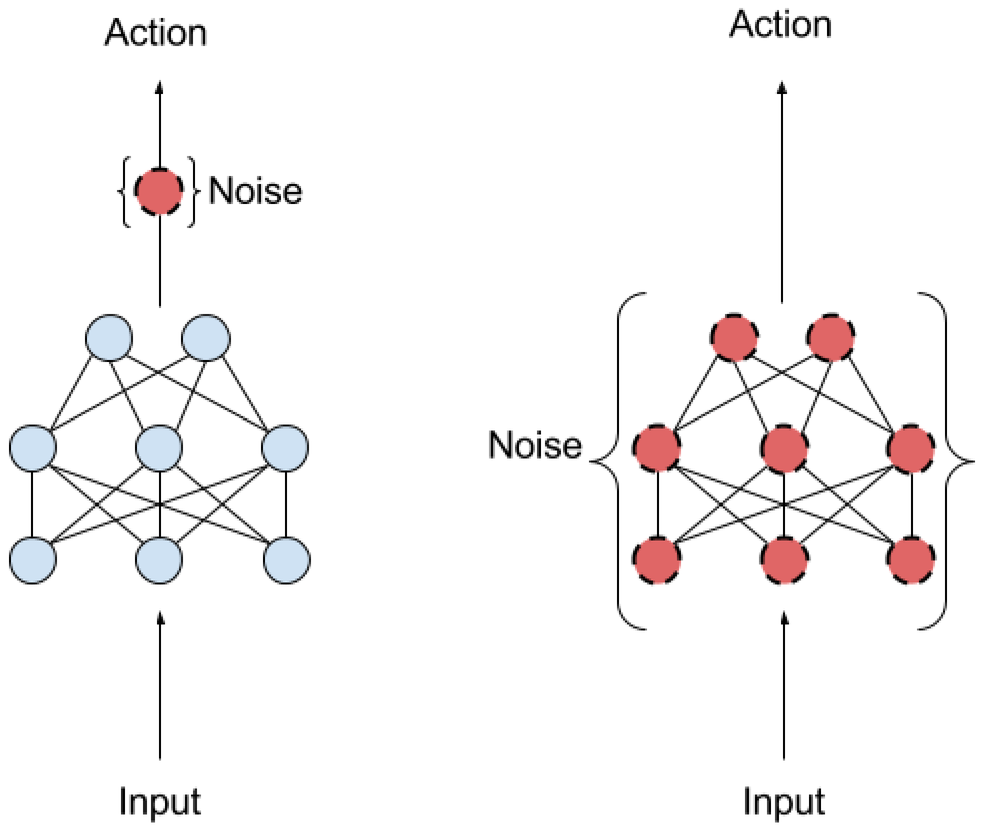


Model-Free RL



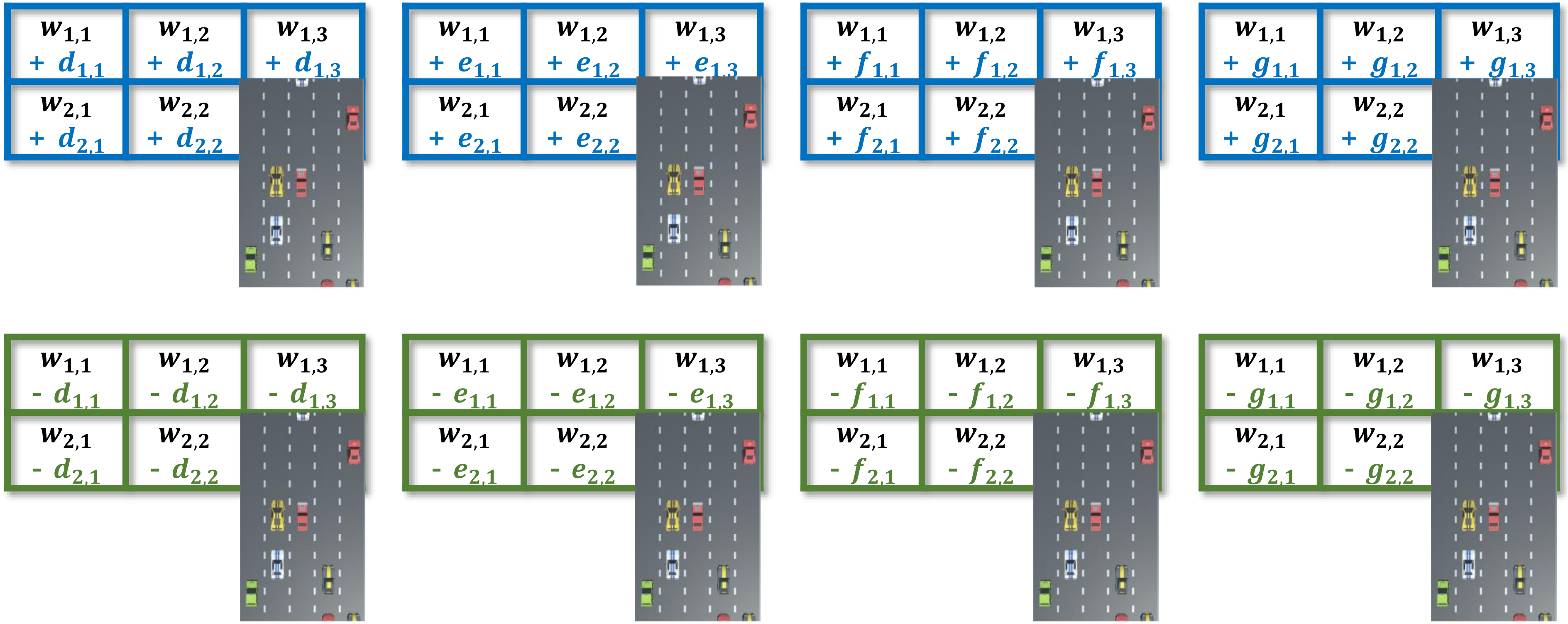
Proposed IL

Autonomous Driving with Imitation Learning

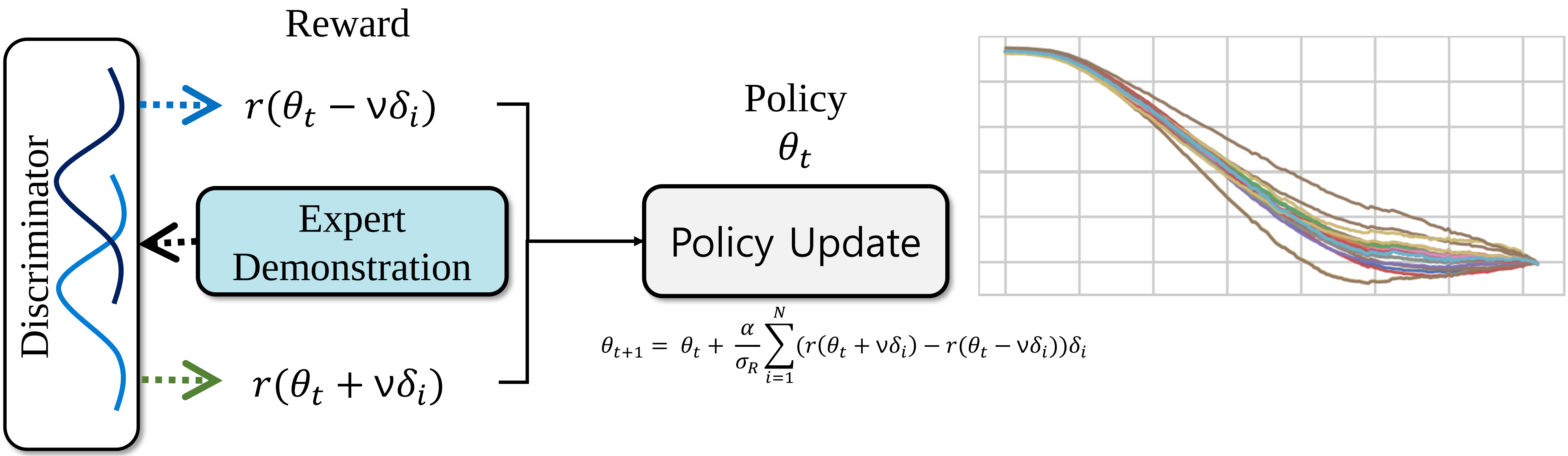


Better Exploration with Parameter Noise - OpenAI

Autonomous Driving with Imitation Learning

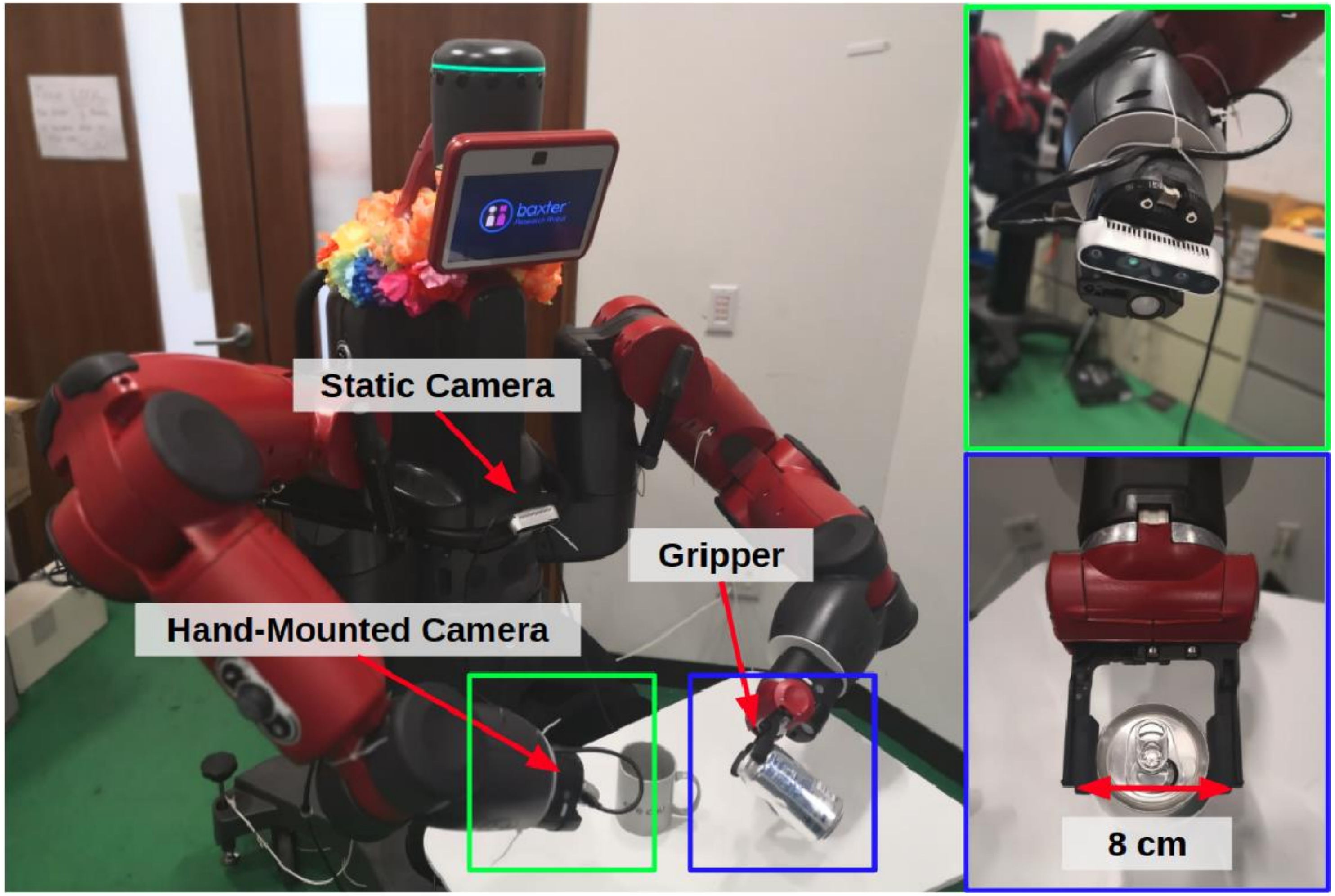


Autonomous Driving with Imitation Learning

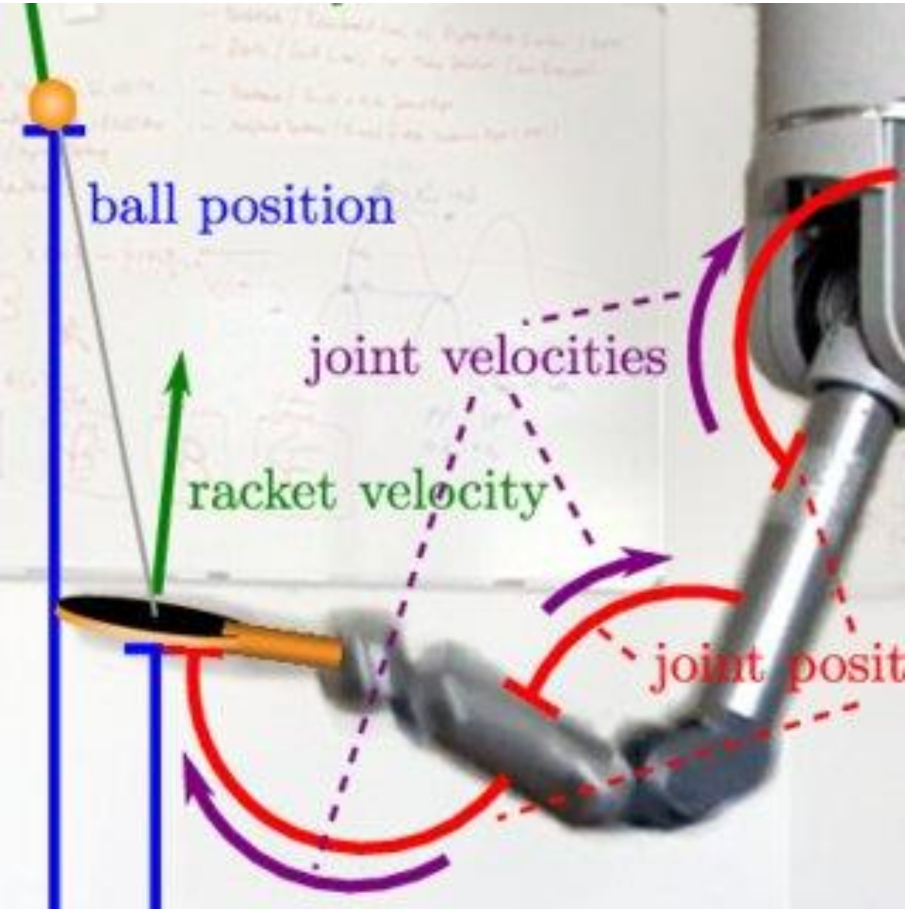


Robot Manipulation

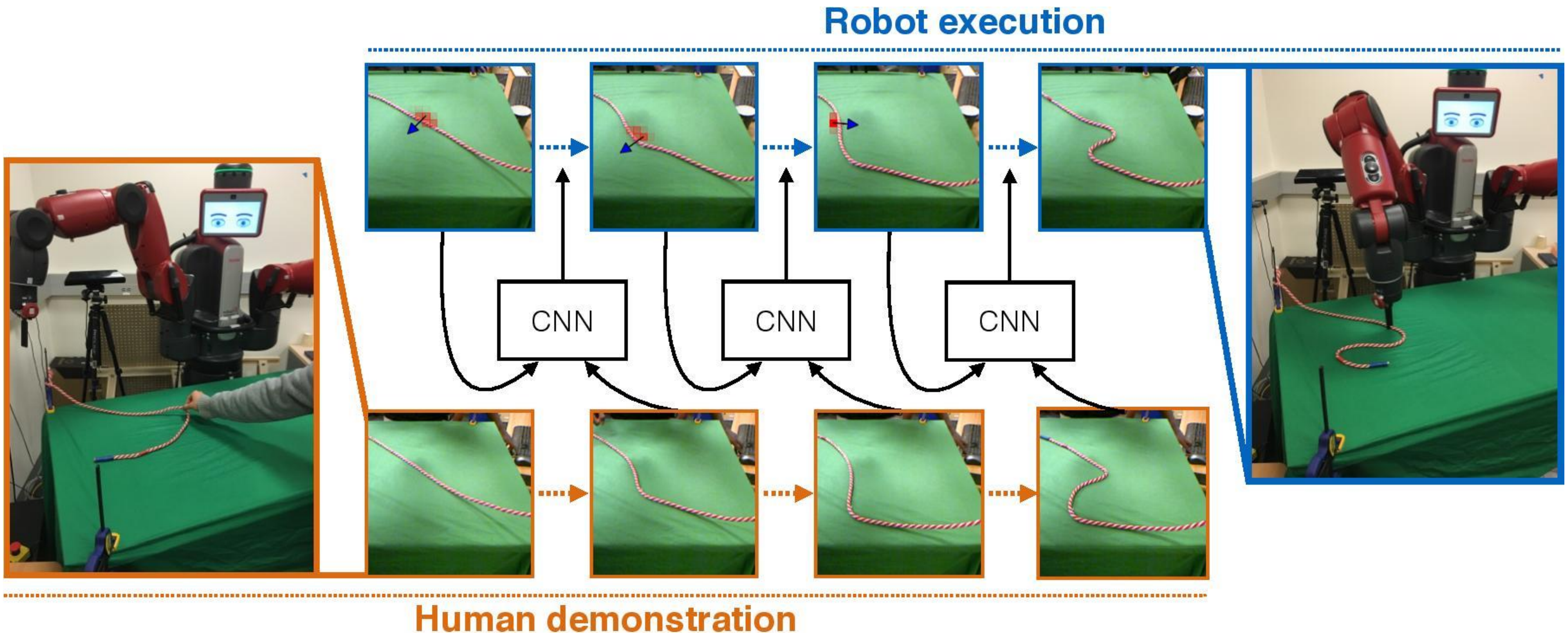
Robotics with Imitation Learning



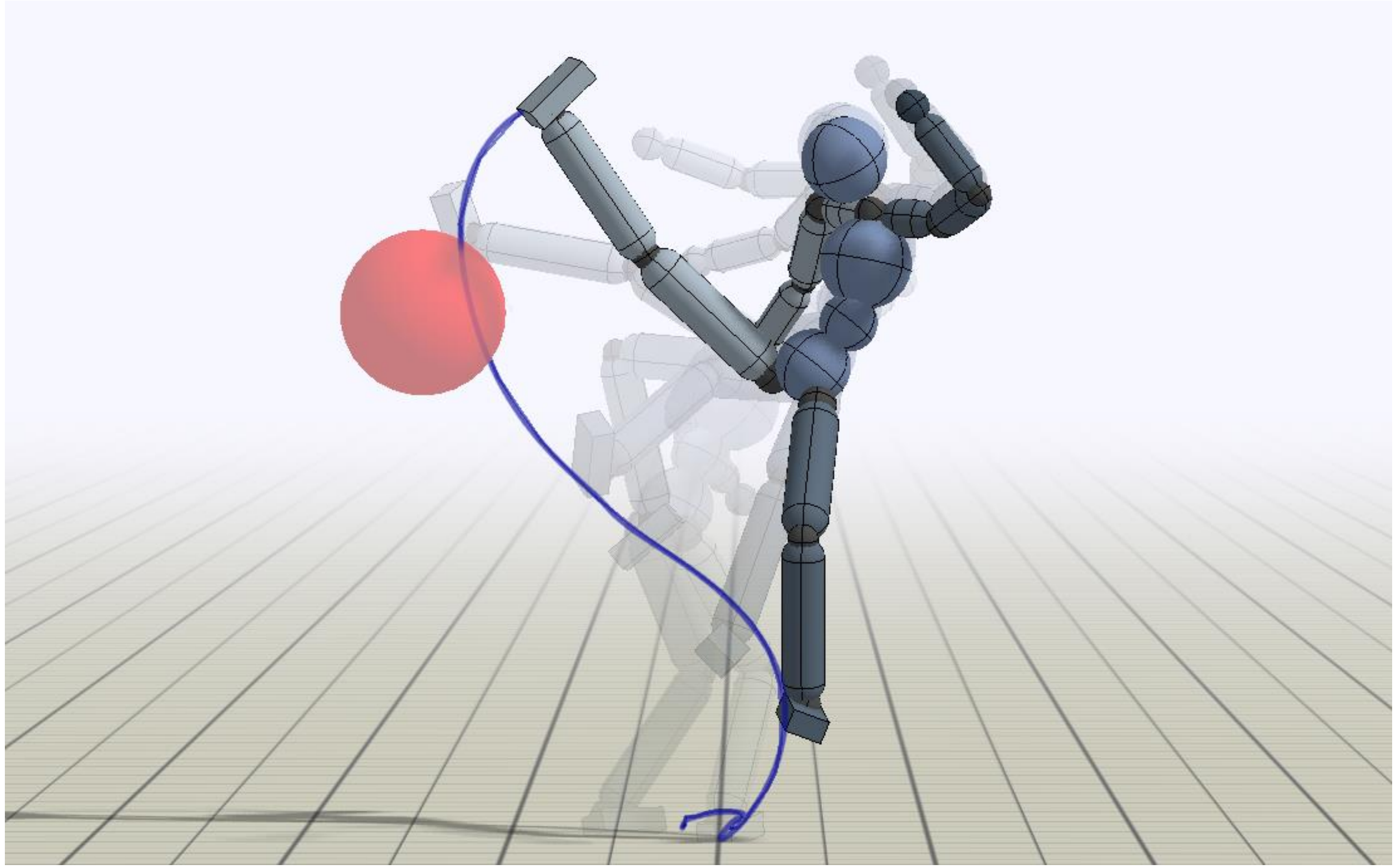
Robotics with Imitation Learning



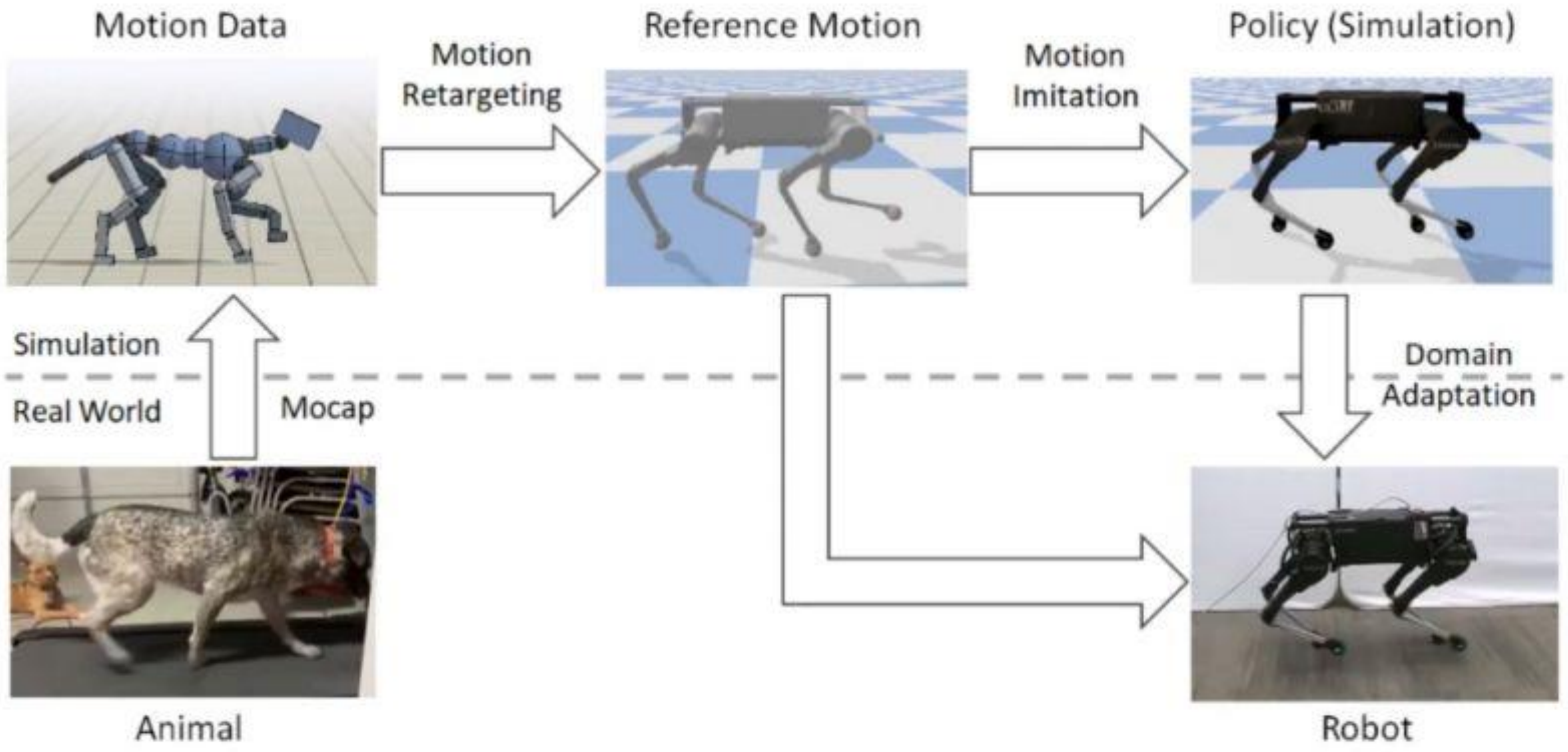
Robotics with Imitation Learning



Robotics with Imitation Learning



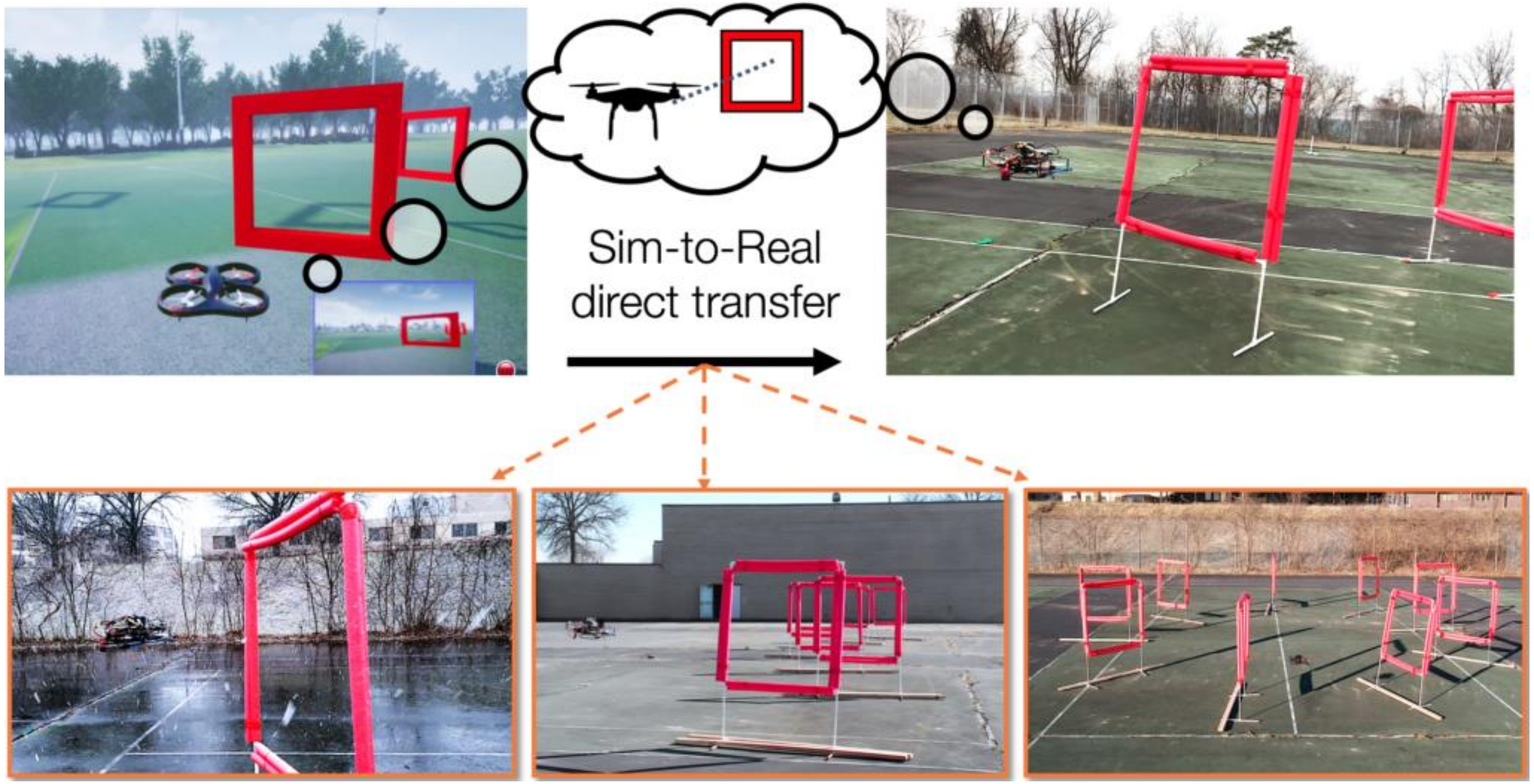
Robotics with Imitation Learning



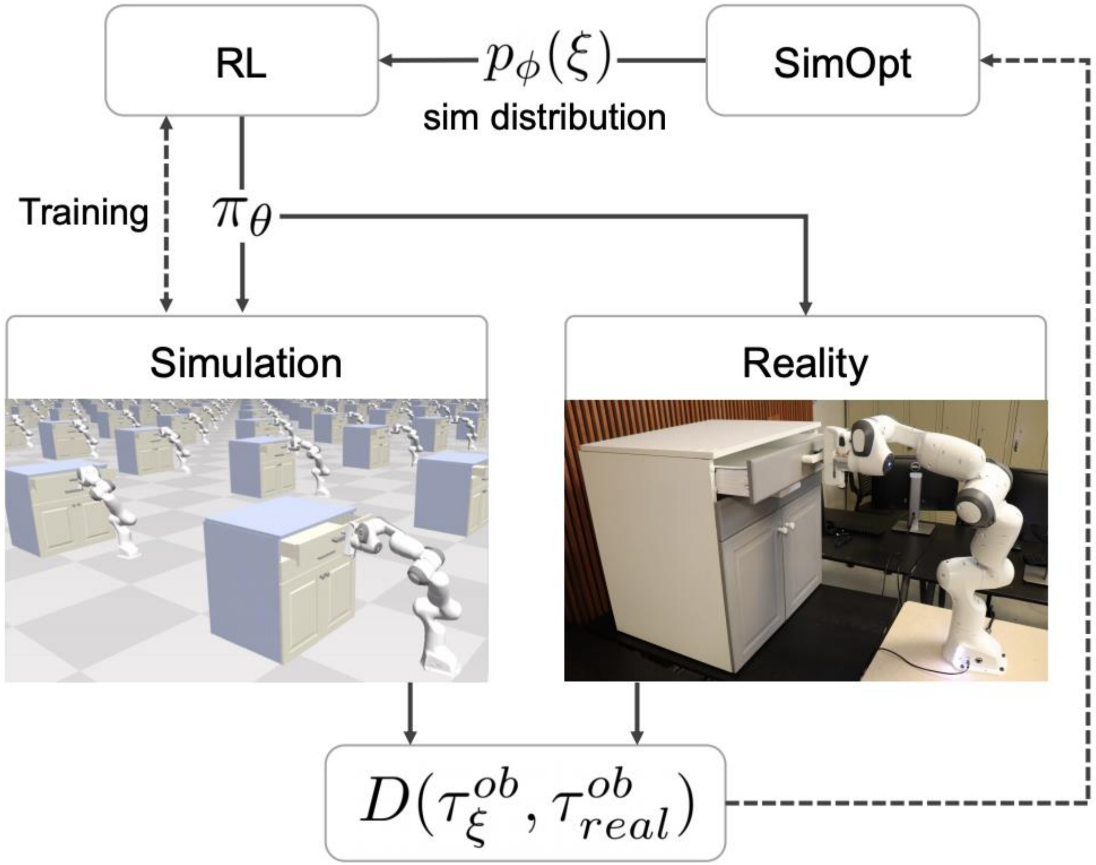
Learning Agile Robotic Locomotion Skills by Imitating Animals (RSS 2020)

Sim to Real

Sim to Real

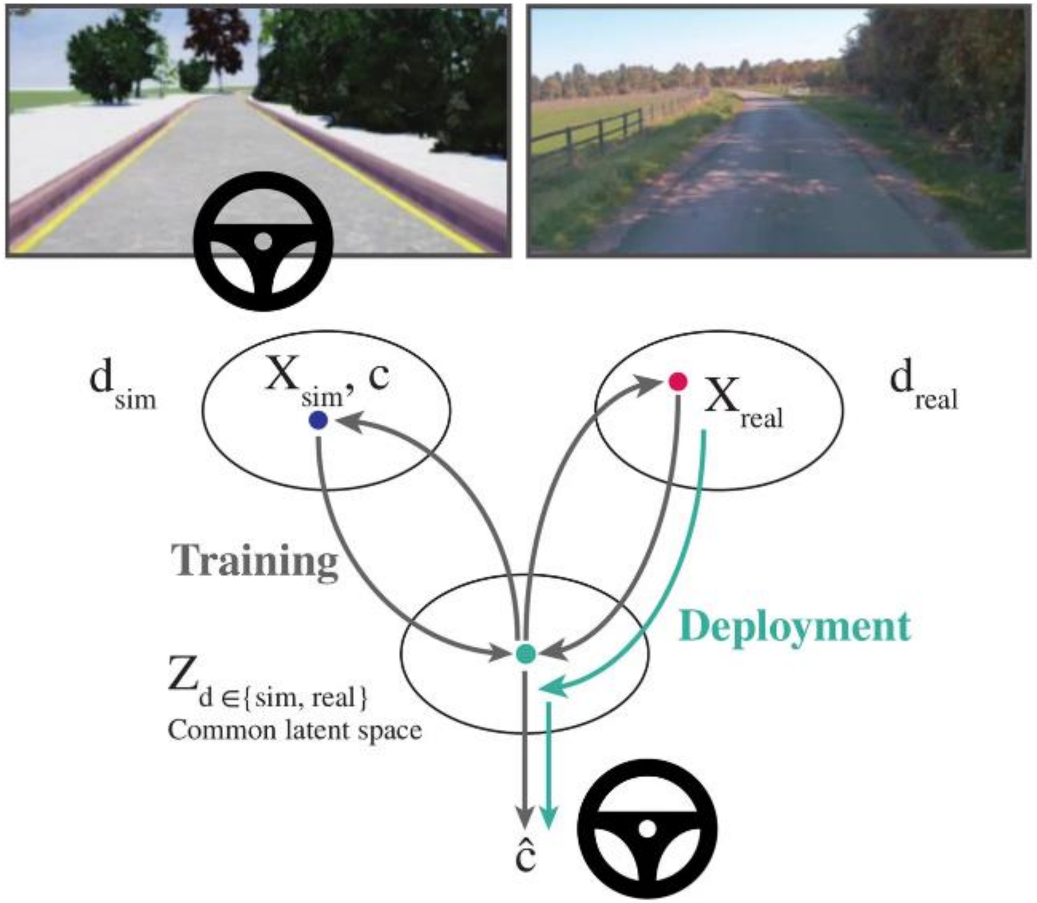


Sim to Real



Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience (ICRA 2019)

Sim to Real



https://youtu.be/D7ZglEPu4lM

Q&A