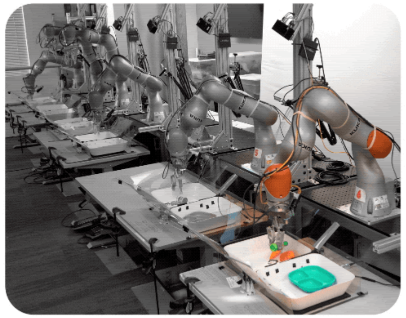
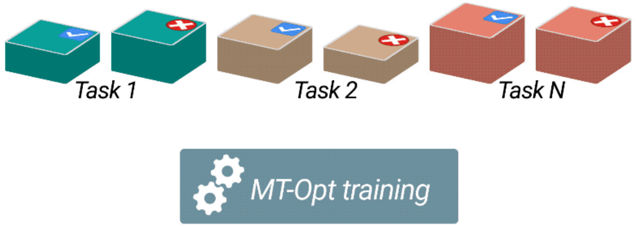
Reinforcement Learning
Autumn 2024

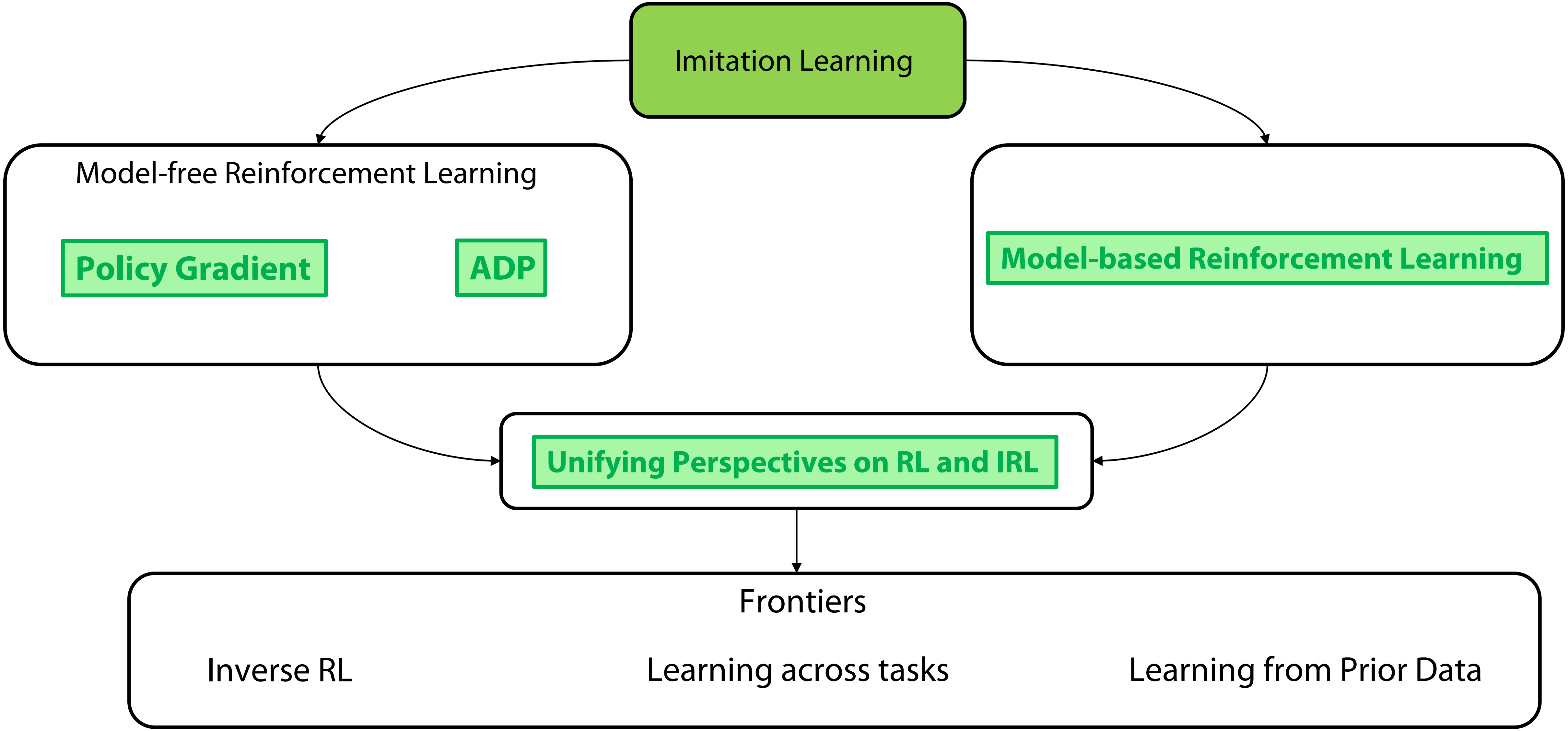
Abhishek Gupta

TA: Jacob Berg





Class Structure



Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

↓

Multi-Task Reinforcement Learning

↓

Meta-Reinforcement Learning

IRL problem statement + assumptions

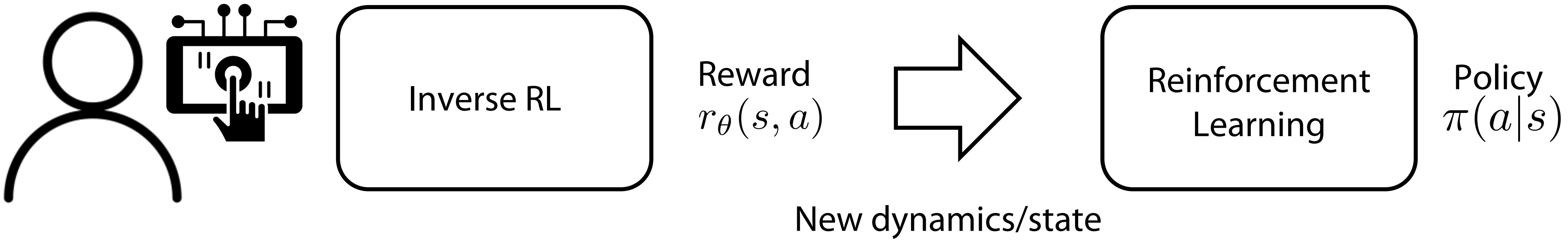
Reinforcement Learning

State: Known
Action: Known
Transition Dynamics: Unknown but can sample
Reward: **Known**
Expert policy: Unknown
Expert traces: **Unknown**

Inverse Reinforcement Learning

State: Known
Action: Known
Transition Dynamics: Unknown but can sample
Reward: **Unknown**
Expert policy: Unknown
Expert traces: **Known**

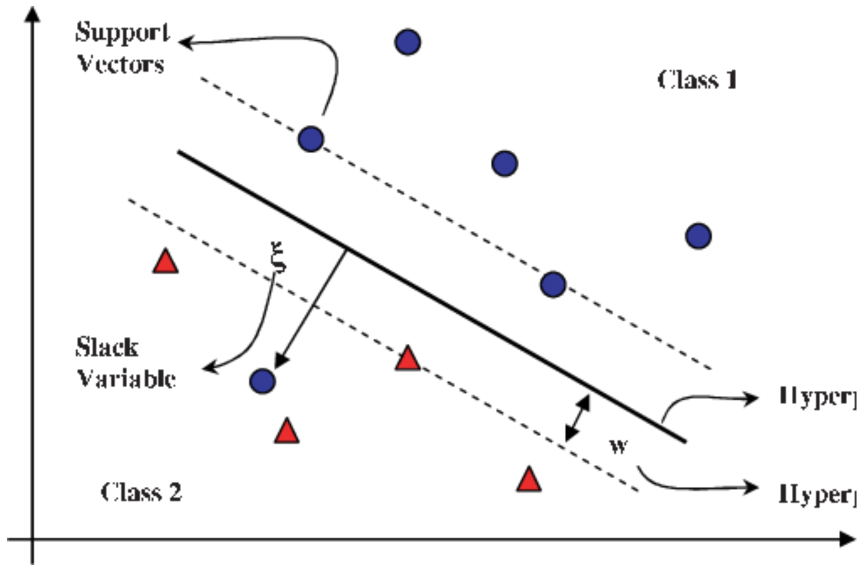
Find r that **explains** the demonstrator behavior as noisily optimal



IRL v1 – (Fancy) Max Margin Feature Matching

Maximum margin → Structured Max-Margin + Slack

$$\min \|w\|_2$$
$$\text{s.t} \;\; w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + 1, \forall \pi \in \Pi$$

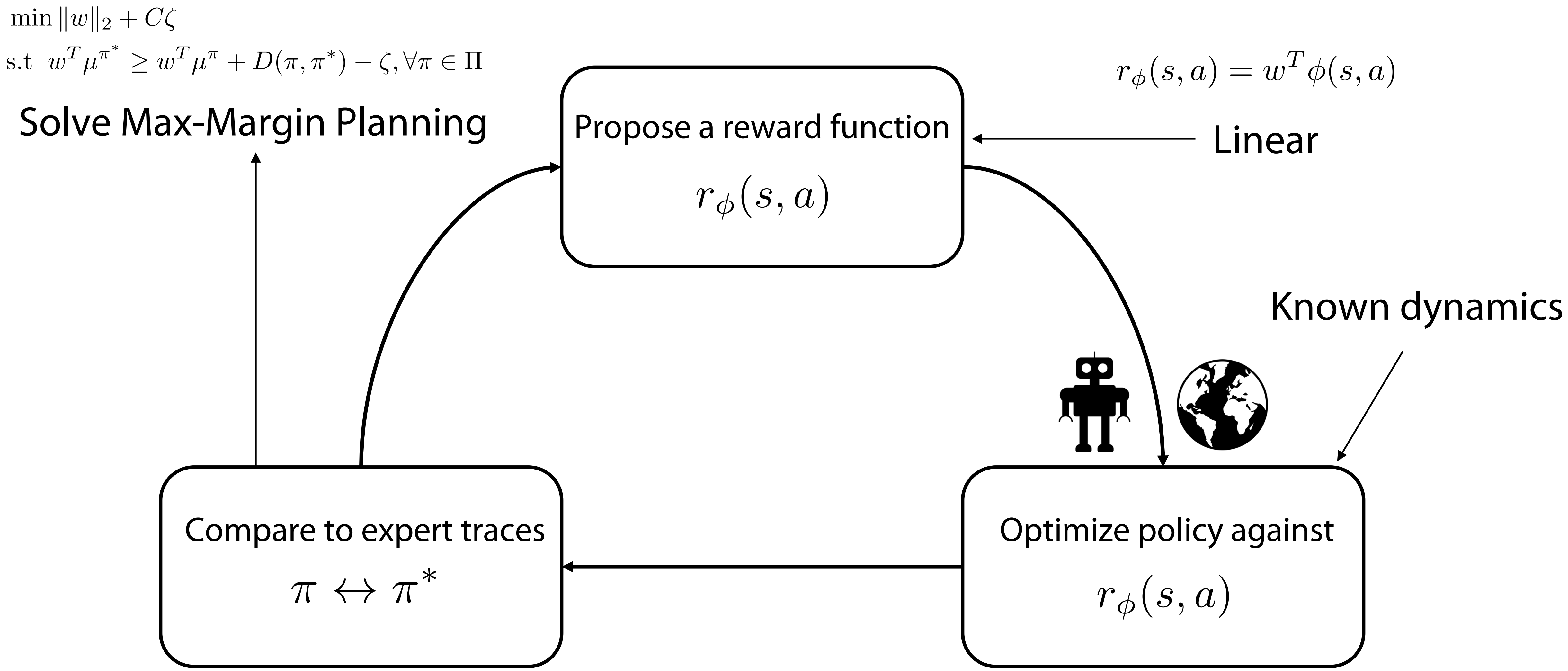


Bigger for more different policies

$$\min \|w\|_2 + C\zeta$$
$$\text{s.t} \;\; w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + D(\pi, \pi^*) - \zeta, \forall \pi \in \Pi$$

Slack allows for noisy optimality

IRL v1 – Max Margin Feature Matching



Maximum Entropy IRL Formulation

$$\max_p \; \mathcal{H}(p(\tau)) = -\int p(\tau)\log p(\tau)d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

Set up the Lagrangian

$$\max_p \min_{w,\lambda} \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

$$\min_{w,\lambda} \max_p \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

Solve wrt p

Solve wrt w, λ

Connect the dots!

Turns out this has nice intuitive properties

$$\max_p \ \mathcal{H}(p(\tau)) = -\int p(\tau) \log p(\tau) d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

$\Downarrow$

Objective reduces to $\min_w \log Z - w^T \mu(\pi^*)$

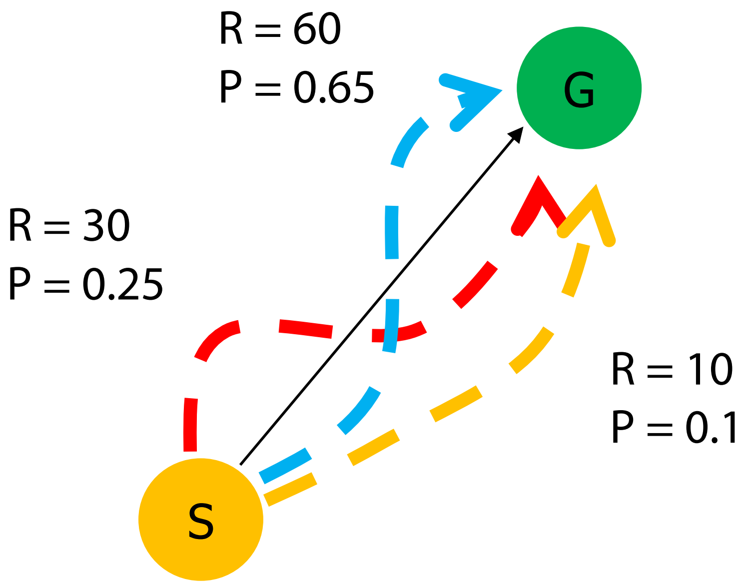
$$Z = \int \exp(w^T \mu(\tau)) d\tau$$

$\Downarrow$

$$\max_w \log \frac{\exp(w^T \mu(\pi^*))}{\int \exp(w^T \mu(\tau)) d\tau}$$

Maximum likelihood with exponential family

$$= \max_w \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\log \frac{\exp(w^T \mu(\tau^*))}{\int \exp(w^T \mu(\tau)) d\tau} \right]$$



Intuition: trajectories are chosen **proportional** to their reward

IRLv2 – Maximum Entropy Inverse RL

$$\nabla J(w) = \mathbb{E}_{\tau^* \sim \mathcal{D}^e}\left[\nabla_w w^T \mu(\tau^*)\right] - \mathbb{E}_{\tau \sim p_w^*(\tau)}\left[\nabla_w w^T \mu(\tau)\right]$$

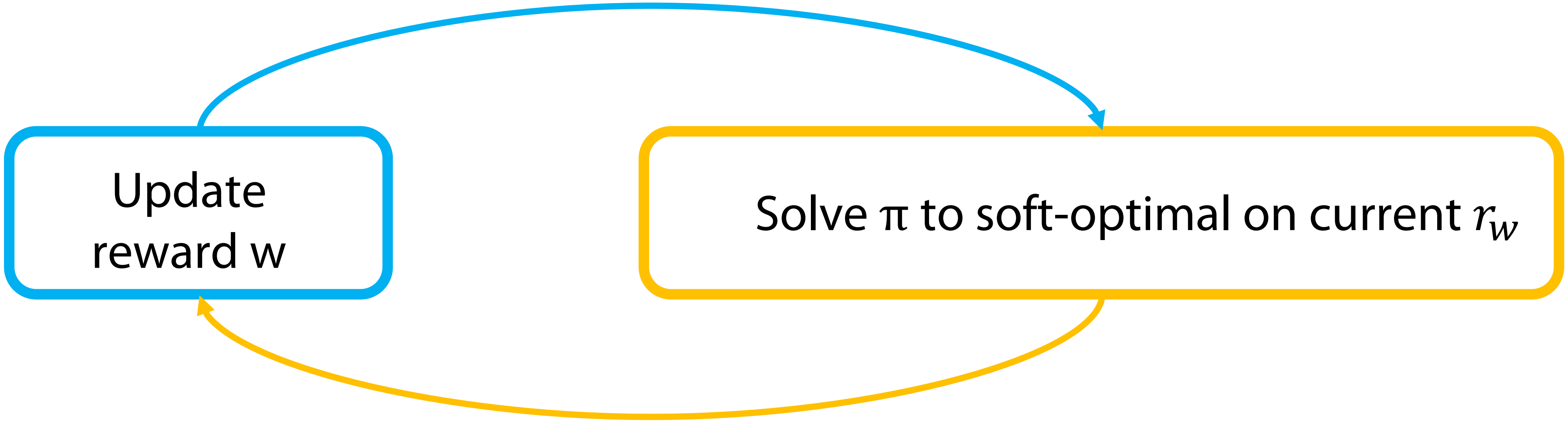
Push up on data

Push down on policy

Soft optimal policy for

$$r_w(s_t, a_t) = w^T \phi(s_t, a_t)$$

$$p_w^*(\tau) = \frac{\exp(w^T \mu(\tau))}{\int \exp(w^T \mu(\tau'))d\tau'}$$



IRL v2 – Max-Ent IRL – Put it together

Maximum Entropy

$$\nabla_w \mathcal{L} = \mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \phi(s_t, a_t)\right]$$
$$-\mathbb{E}_{p_w(\tau)}\left[\sum_t \gamma^t \phi(s_t^\tau, a_t^\tau)\right]$$

$$r_\phi(s,a) = w^T \phi(s,a)$$

Propose a reward function

$r_\phi(s,a)$

Linear

Known dynamics

Optimize policy against

$r_\phi(s,a)$

Compare to expert traces

$\pi \leftrightarrow \pi^*$

IRL v2 –Max-Entropy Inverse RL (Pseudocode)

1. Start with a random policy π_0 and weight vector w
2. Find the “soft” optimal policy under w – $p_w(\tau)$
3. Take a gradient step on w

$$\nabla_w \mathcal{L} = \mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \phi(s_t, a_t)\right] - \mathbb{E}_{p_w(\tau)}\left[\sum_t \gamma^t \phi(s_t^\tau, a_t^\tau)\right]$$

4. Repeat

Output the optimal reward function w*

Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

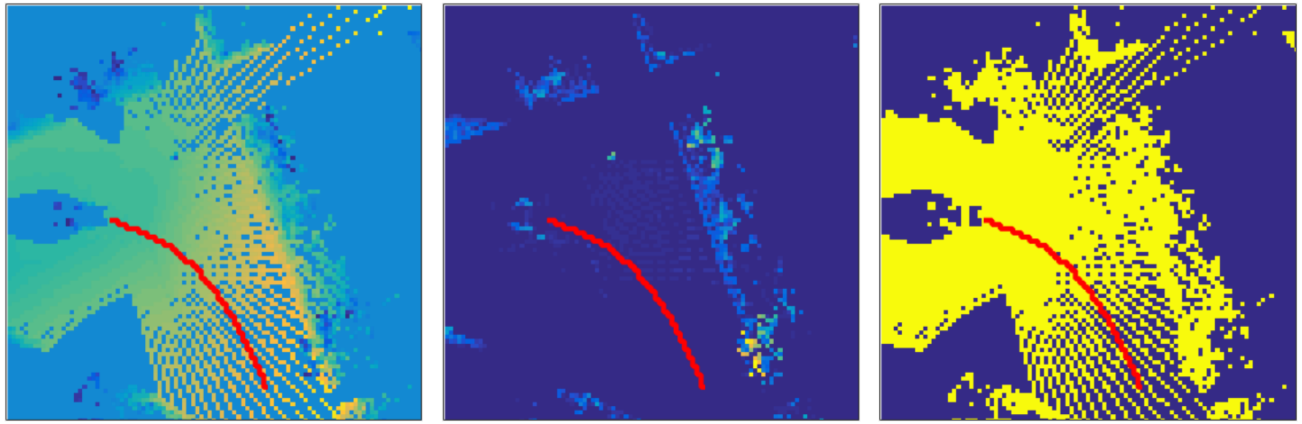
↓

Multi-Task Reinforcement Learning

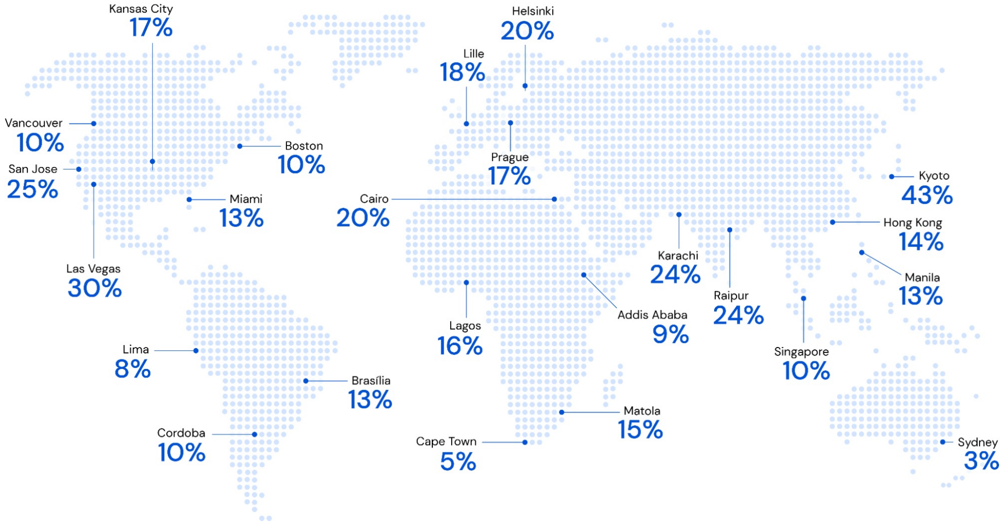
↓

Meta-Reinforcement Learning

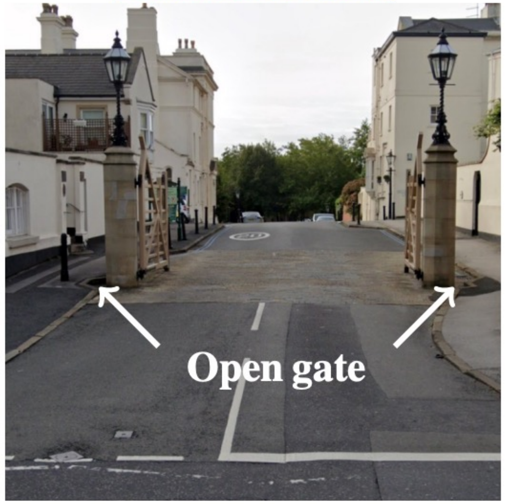
Max-Ent IRL in Action



Max-Ent IRL in Action



Preferred route



Detour route

Ok but no way this could work?

Maximum Entropy

$$\max_{p(\tau)} \quad \mathcal{H}(p(\tau))$$
$$\text{s.t} \quad \mathbb{E}_{p(\tau)}[\phi(s,a)] \approx \mathbb{E}_{\pi^*}[\phi(s,a)]$$

Propose a reward function

$r_\phi(s,a)$

$r_\phi(s,a) = w^T\phi(s,a)$

Linear

Known dynamics

Optimize policy against

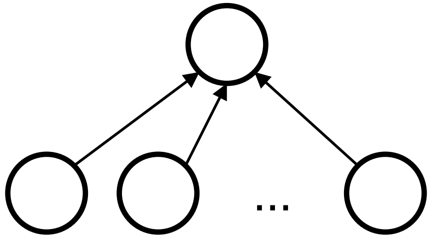
$r_\phi(s,a)$

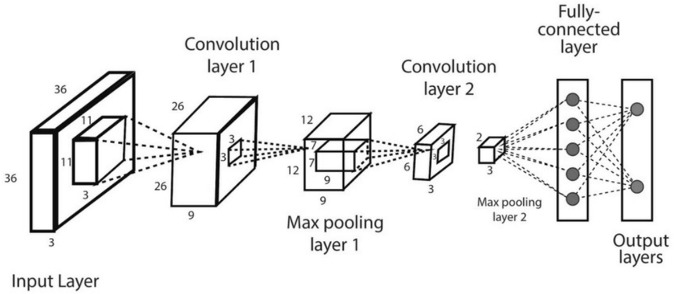
Compare to expert traces

$\pi \leftrightarrow \pi^*$

Linear Rewards → Neural Net Rewards

Max-ent IRL allows us to go from linear rewards to arbitrary neural network rewards





Linear Max-Ent IRL

$$\max_{w} \mathbb{E}_{\pi^*}\left[\sum_t w^T \gamma^t \phi(s_t, a_t)\right] - \log \int_\tau \left[\exp\left(\sum_t w^T \gamma^t \phi(s_t, a_t)\right)\right] d\tau$$

⇓

Non-Linear Max-Ent IRL

$$\max_{\theta} \mathbb{E}_{\pi^*}\left[\sum_t \gamma^t r_\theta(s_t, a_t)\right] - \log \int_\tau \left[\exp\left(\sum_t \gamma^t r_\theta(s_t, a_t)\right)\right] d\tau$$

Can simply replace, w with arbitrary θ and use autodiff!

Avoiding Complete Policy Optimization

Optimize policy against

$r_\phi(s, a)$

Assumes dynamics are known so we can just do (fast) planning

What happens when dynamics are unknown!

$$\mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$

$$-\mathbb{E}_{p_w(\tau)}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$

What if we only **<u>improved</u>** the policy a little bit

Biased!

Requires complete "soft" policy optimization

Avoiding Complete Policy Optimization

Importance sampling to the rescue!

$$\mathbb{E}_{p(x)}\left[f(x)\right] = \mathbb{E}_{q(x)}\left[\frac{p(x)}{q(x)} f(x)\right]$$

$$\mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right] - \mathbb{E}_{p_w(\tau)}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$

Importance Sampling

$$\mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right] - \mathbb{E}_{q}\left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$

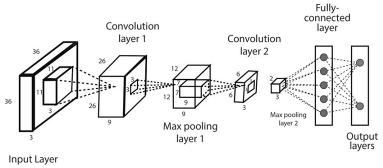
$$\frac{\exp(\sum_t r_\theta(s_t, a_t))}{\Pi_t \pi_\theta(a_t|s_t)}$$

Can transfer significantly more from iteration to iteration rather than doing full nested optimization

IRLv4 – Guided Cost Learning

Gradient Step on Reward

$$\mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$
$$-\mathbb{E}\left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$



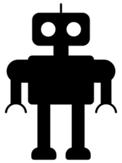
Propose a reward function

$r_\phi(s, a)$

$r_\phi(s, a)$

Neural network

Gradient step on policy



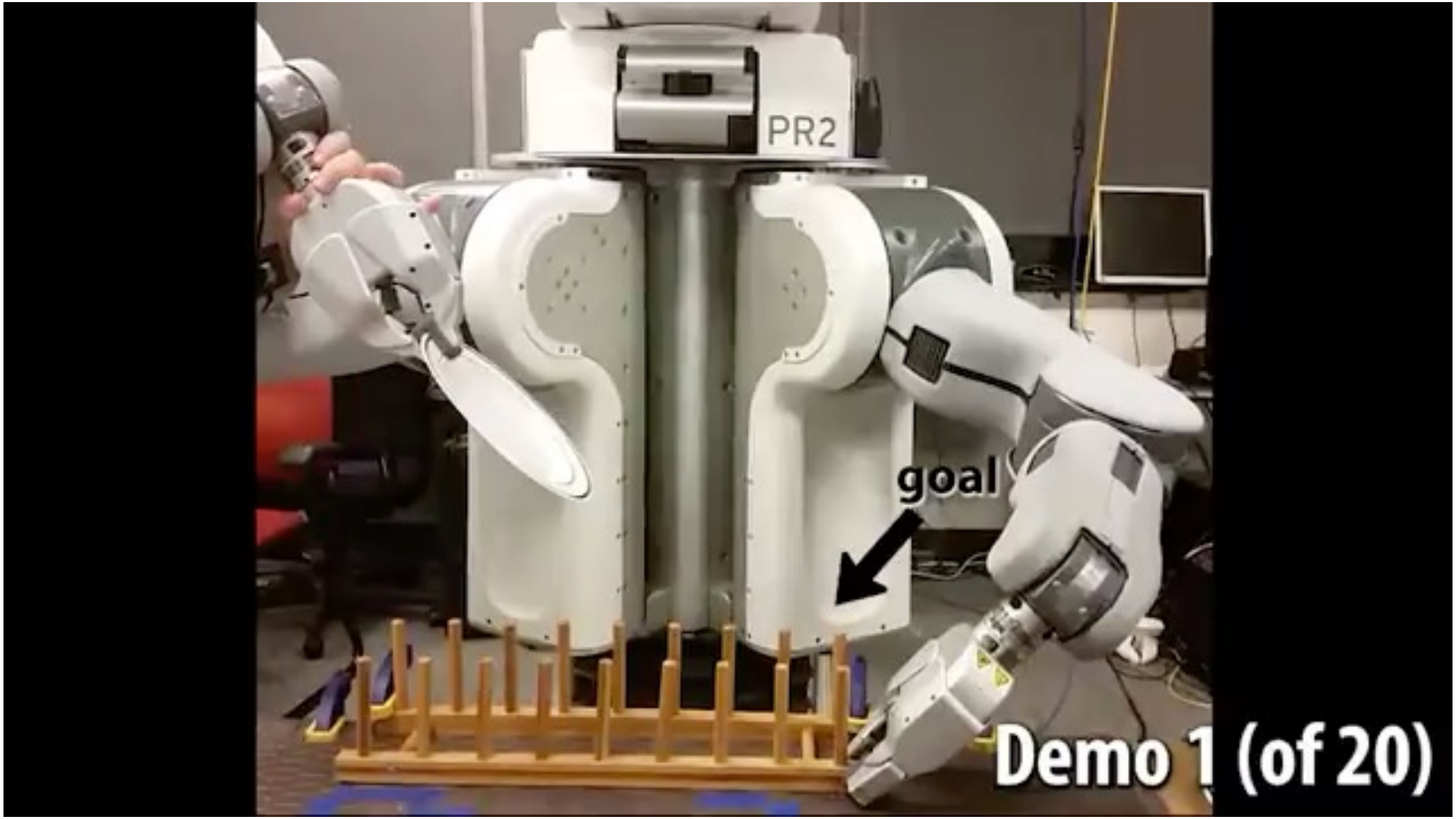
Optimize policy against

$r_\phi(s, a)$

Compare to expert traces

$\pi \leftrightarrow \pi^*$

IRLv4 – Guided Cost Learning



Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

↓

Multi-Task Reinforcement Learning

↓

Meta-Reinforcement Learning

Connecting Maximum-Entropy RL to GANs

Looks like a game

Generator

Discriminator

1. Start with a random policy π_0 and weight vector w
2. Take a step on "soft" optimal policy under w – $p_w(\tau)$
3. Take a gradient step on w

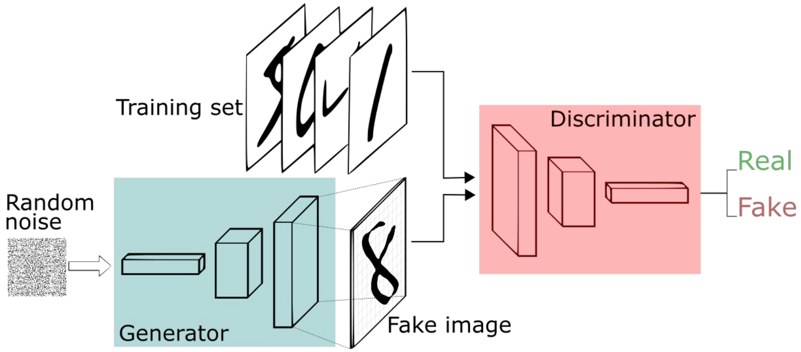
$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right] - \mathbb{E}_q\left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$

4. Repeat

Output the optimal reward function w*

Reminder: Generative Adversarial Networks

Technique to learn generative models via a 2 player game



https://sthalles.github.io/intro-to-gans/

Key idea: Generator tries to "confuse" the discriminator.
At convergence generated samples indistinguishable from real samples

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p(z)}\left[\log(1-D(G(z)))\right]$$

Often approximate generator loss as:

$$\min_G \mathbb{E}_{z\sim p(z)}\left[\log(1-D(G(z)))\right] - \mathbb{E}_{z\sim p(z)}\left[\log(D(G(z)))\right]$$

Can inverse RL be considered a GAN?

Generator = policy
Discriminator = reward (kinda)

Find a policy which makes a discriminator unable to tell if the samples came from the policy or the demos

$$\min_G \max_D V(D, G) = \mathbb{E}_{\tau \sim p_{\text{demo}}(\tau)} [\log D(\tau)] + \mathbb{E}_{\tau \sim \pi} [\log(1 - D(\tau))]$$

Push up real data

Push down policy data

Discriminator trained with classification between expert/non-expert

Generator trained to max log D with RL

Challenge: only policy, not really a reward

Generative Adversarial Imitation Learning

Jonathan Ho
Stanford University
hoj@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Tweaking GAIL to connect with IRL

We can make simple tweaks to GAIL to get back to max-ent IRL

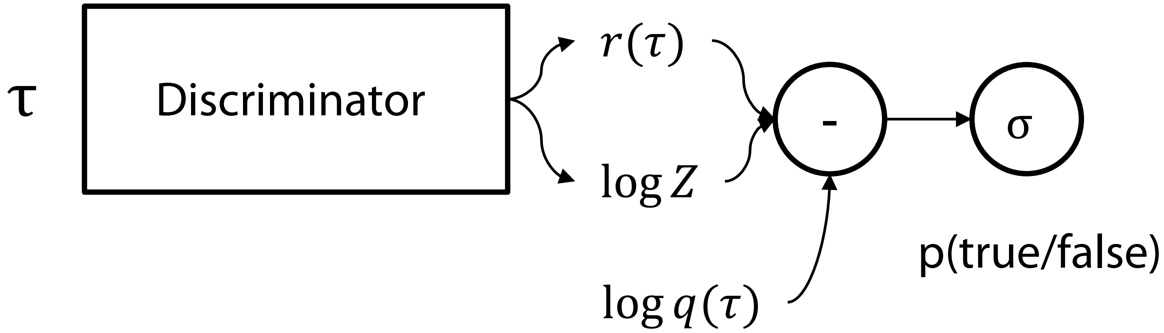
Optimal discriminator

$$D^*(x) = \frac{p(x)}{p(x) + q(x)}$$

Choose a particular form of discriminator $\longrightarrow$

Policy informed discriminator

$$D_\theta(\tau) = \frac{\frac{1}{Z}\exp(r_\theta(\tau))}{\frac{1}{Z}\exp(r_\theta(\tau)) + q(\tau)}$$



Tweaking GAIL to connect with IRL

For a particular parameterization of the discriminator, we can show that GAN = max-ent IRL

Max-Ent Inverse RL

$$\mathbb{E}_{\pi^*}\left[\sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right] - \mathbb{E}_q\left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_\theta r_\theta(s_t, a_t)\right]$$

Push up demos, push down policy

$\Longleftrightarrow$

With some massaging

GAN

$$\mathbb{E}_{\tau \sim p_{\text{demo}}(\tau)}\left[\log D(\tau)\right] + \mathbb{E}_{\tau \sim \pi}\left[\log(1 - D(\tau))\right]$$

Push up real data, push down generated

$$D_\theta(\tau) = \frac{\frac{1}{Z}\exp(r_\theta(\tau))}{\frac{1}{Z}\exp(r_\theta(\tau)) + \Pi_t \pi_\theta(a_t|s_t)}$$

Generator Optimization as Max-Ent RL

$$\min_G \mathbb{E}_{z\sim p(z)}\left[\log(1 - D(G(z)))\right] - \mathbb{E}_{z\sim p(z)}\left[\log(D(G(z)))\right]$$

$$D_\theta(\tau) = \frac{\frac{1}{Z}\exp(r_\theta(\tau))}{\frac{1}{Z}\exp(r_\theta(\tau)) + q(\tau)}$$

$$\Downarrow$$

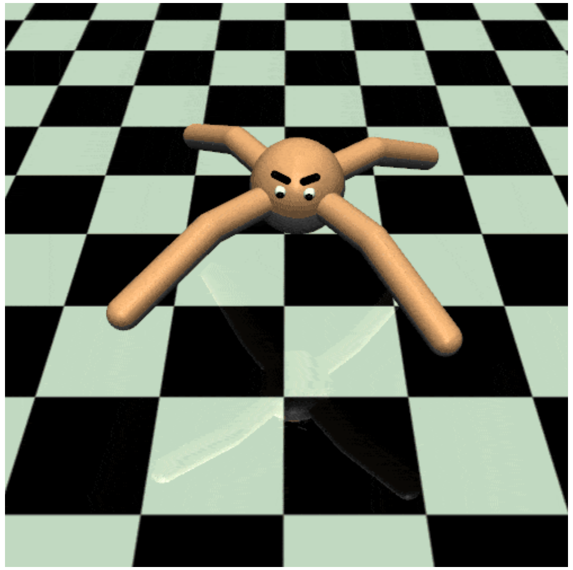
$$\min \mathbb{E}_{\tau\sim q(\tau)}\left[\log \frac{q(\tau)}{\cancel{\frac{1}{Z}\exp(r_\theta(\tau)) + q(\tau)}} - \log \frac{\frac{1}{Z}\exp(r_\theta(\tau))}{\cancel{\frac{1}{Z}\exp(r_\theta(\tau)) + q(\tau)}}\right]$$

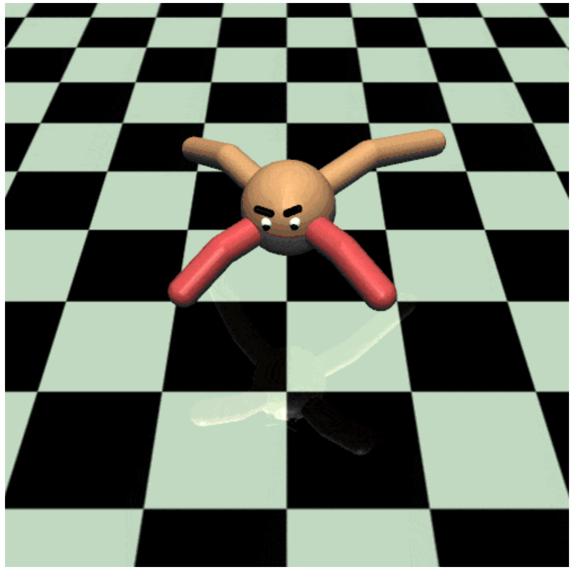
$$\max \mathbb{E}_{\tau\sim q(\tau)}\left[r_\theta(\tau) - \cancel{\log Z} - \log q(\tau)\right]$$

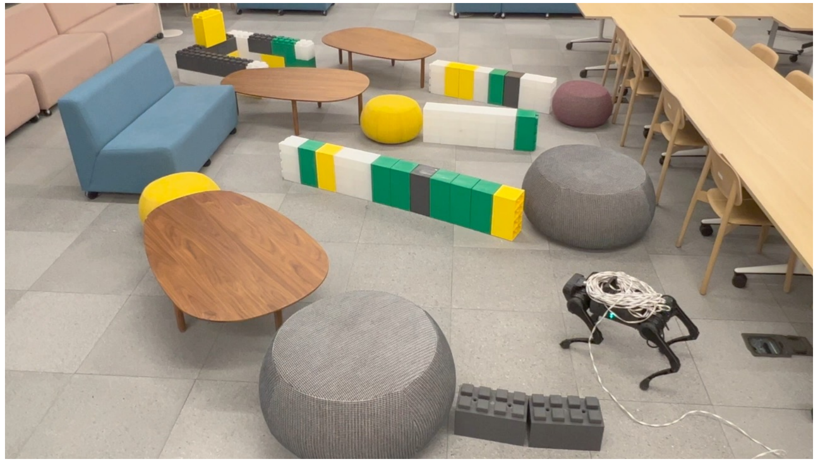
Maximum entropy RL with current reward!

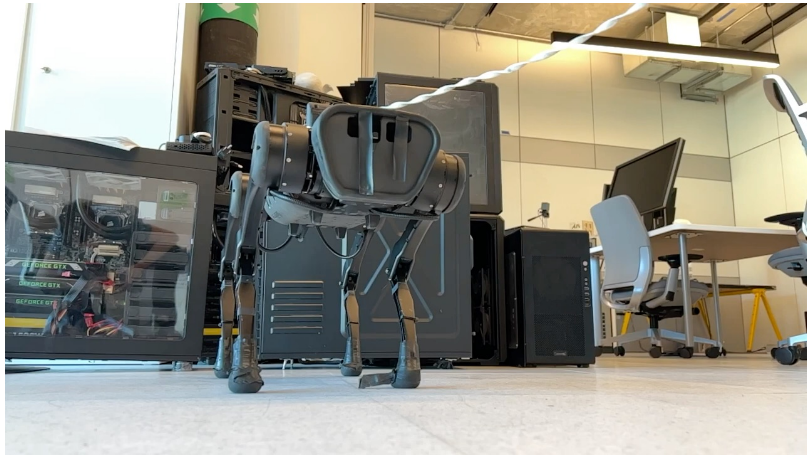
Similar proof holds for the discriminator optimization – refer to https://arxiv.org/pdf/1611.03852

Adversarial IRL in Action



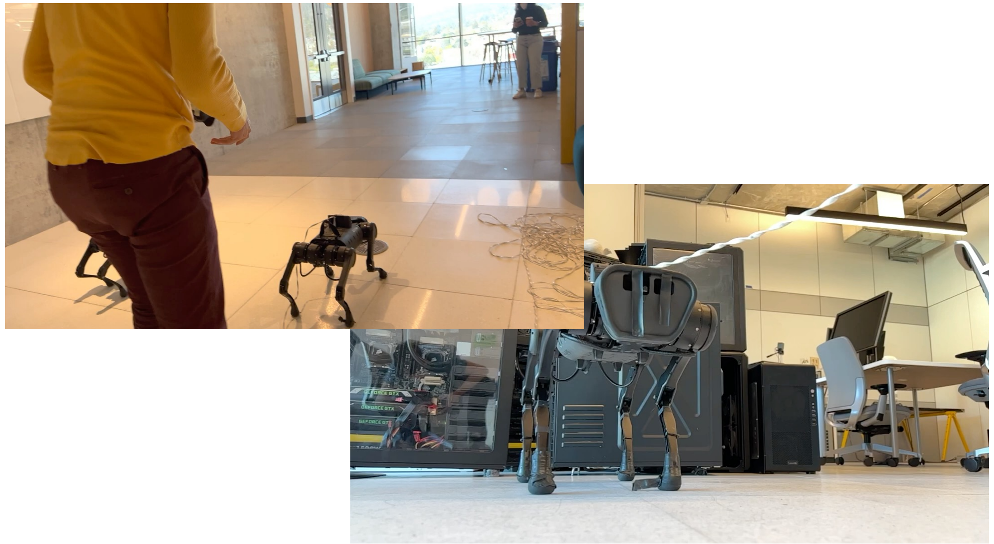




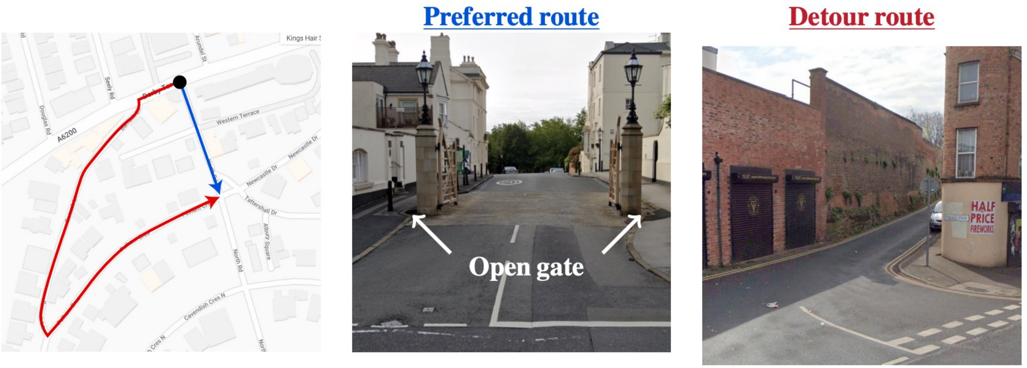


Takeaways on IRL

Pros:
1. Potentially generalizable
2. Can continue improving beyond BC
3. Avoids compounding error
4. Often only option for RL in hard to specify scenarios



Cons
1. Expensive nested optimization
2. Inherent ambiguity
3. Hard to scale reliably



Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

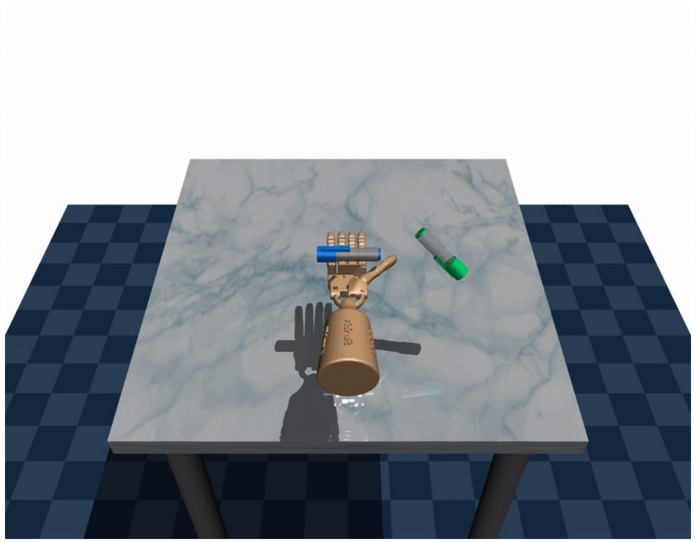
↓

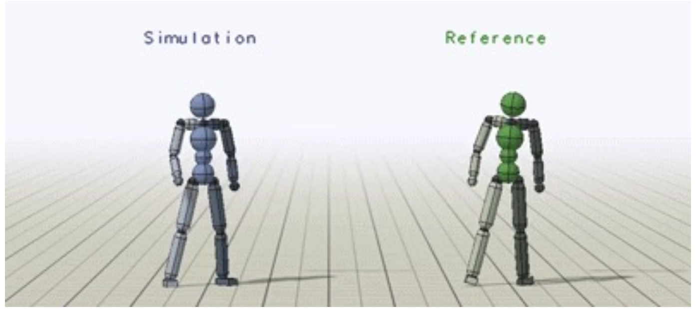
Multi-Task Reinforcement Learning

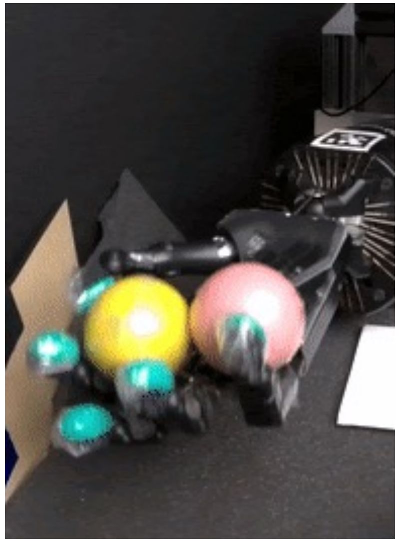
↓

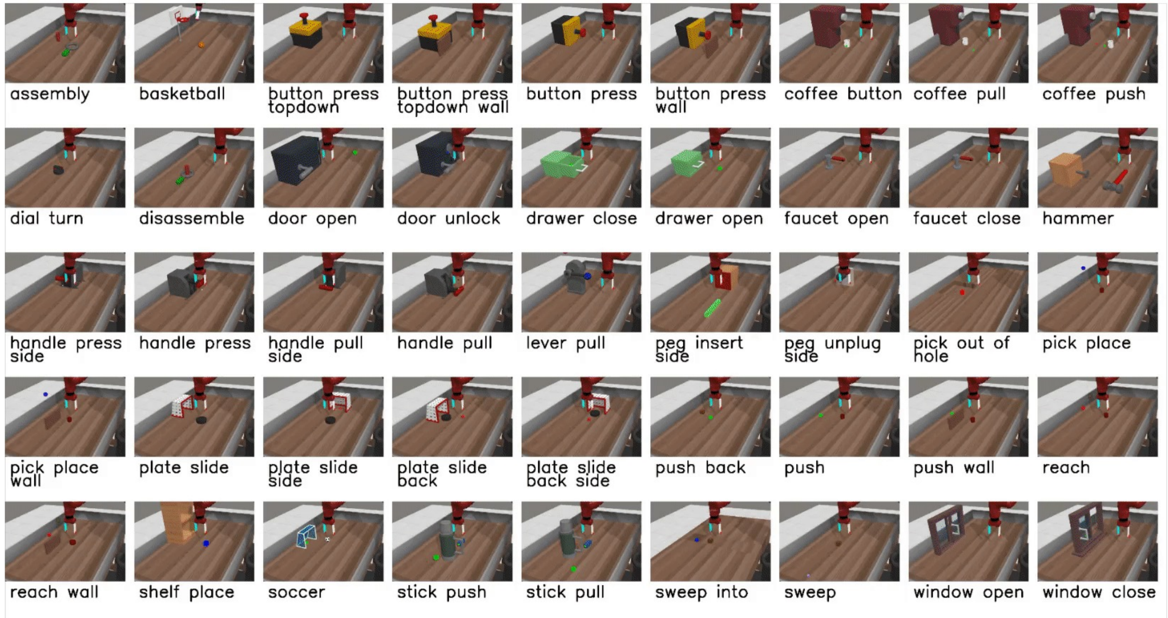
Meta-Reinforcement Learning

From Single Task to Multi-Task RL



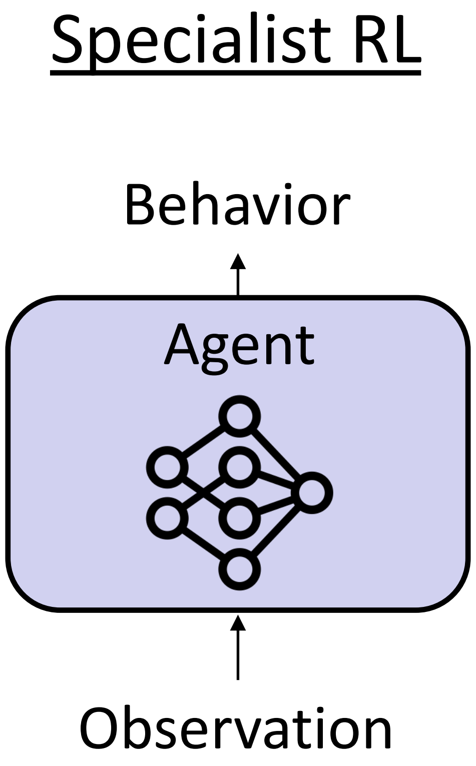


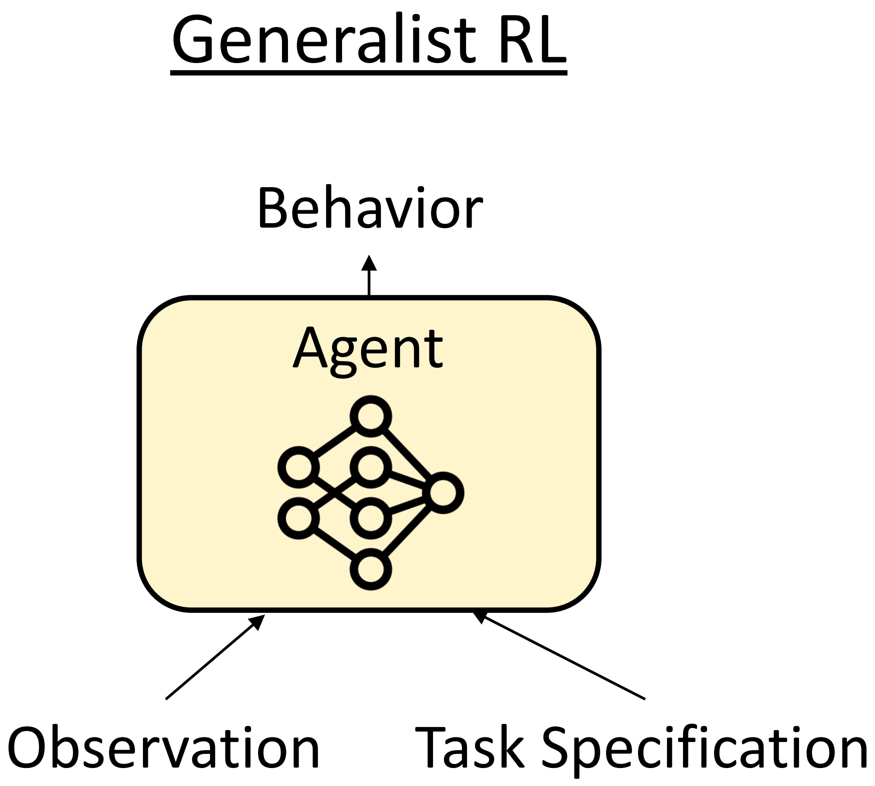




Can we make RL algorithms generalists?

We need a single agent to be able to (quickly or directly) solve multiple different tasks





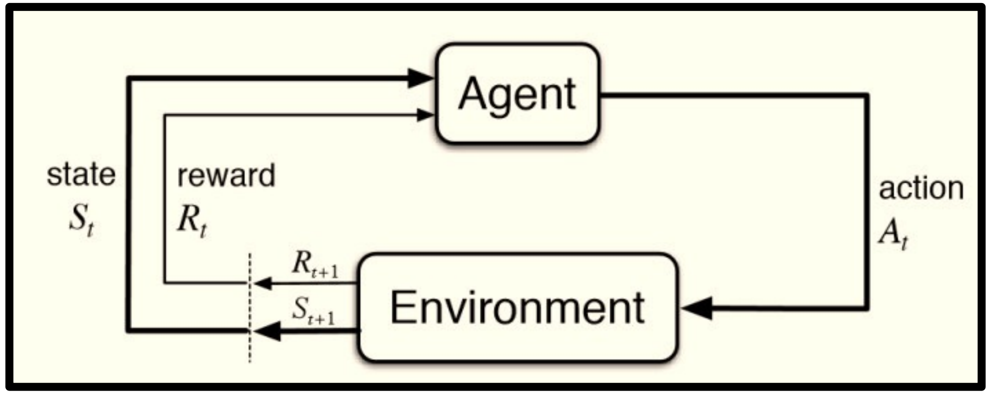
Multi-Task RL – Distribution over MDPs

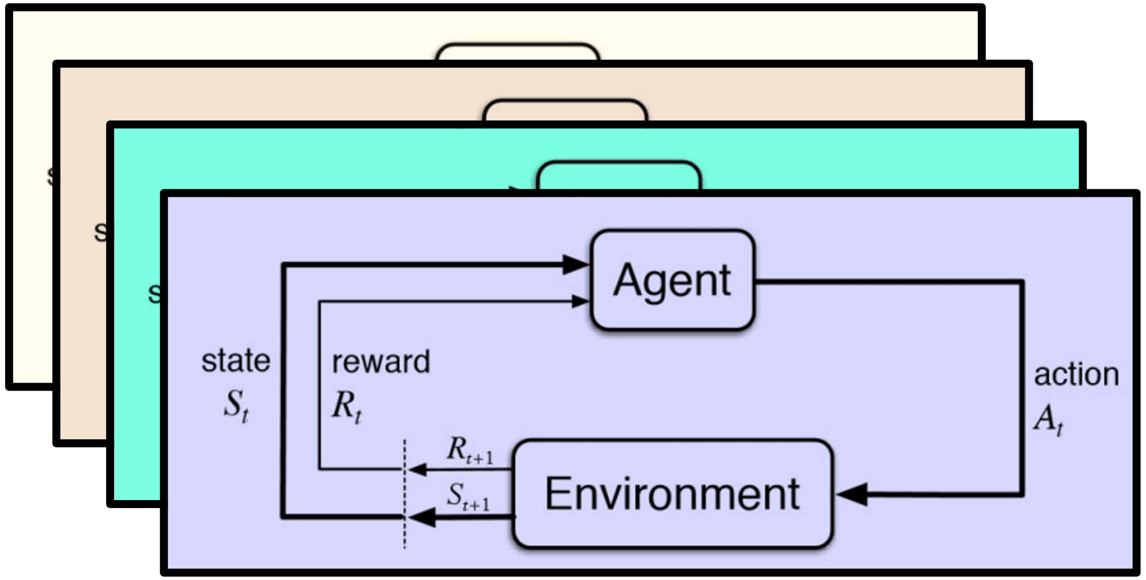
Assumption: Same state/action space, varying dynamics and rewards

$$p(\mathcal{M}_i)$$

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mu, \gamma)$$

$$\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{T}_i, \mathcal{R}_i, \mu, \gamma)$$



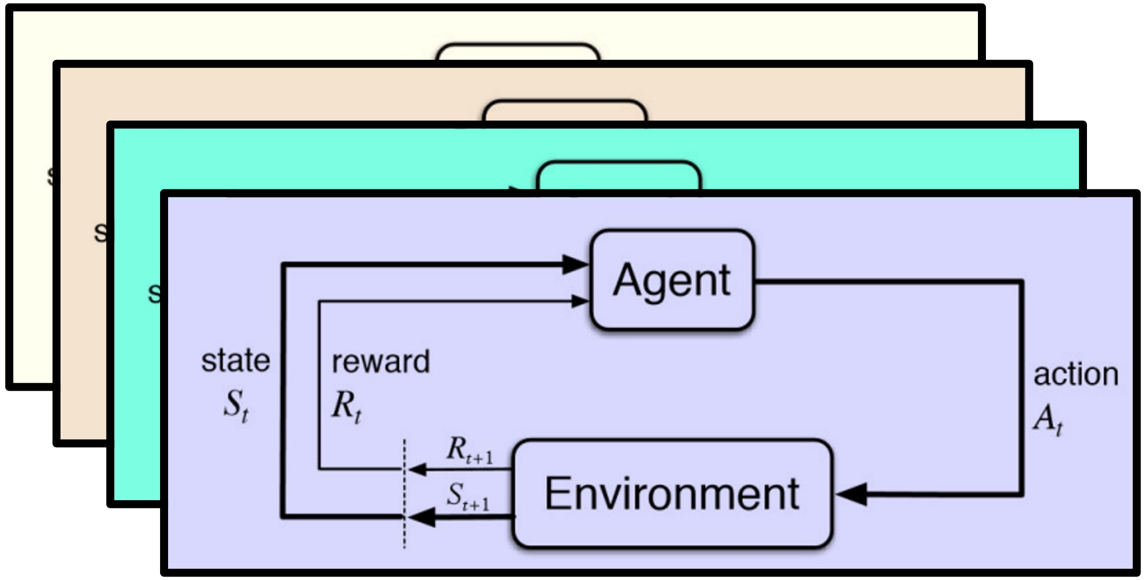


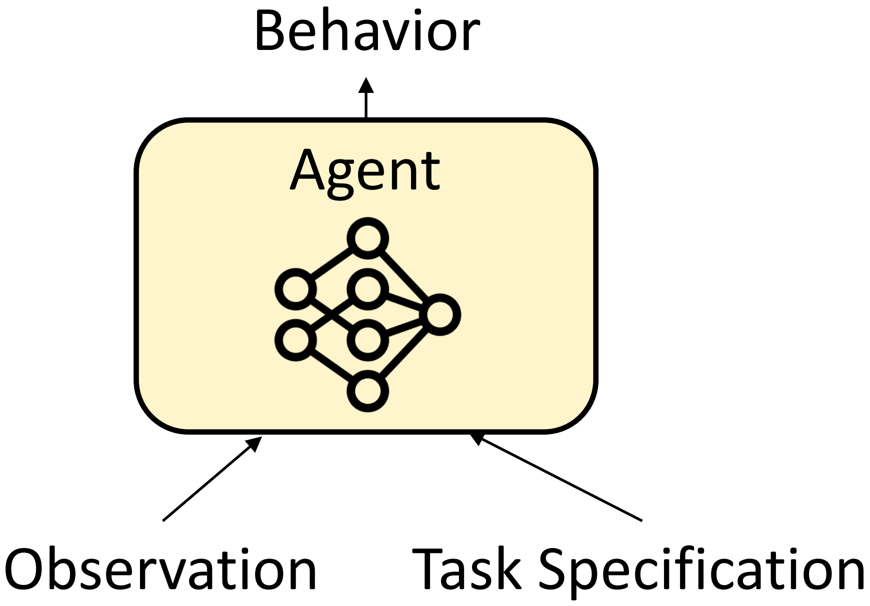
Goals for Today

Our goal: understand different ways to solve meta-MDP/multi-task RL problem

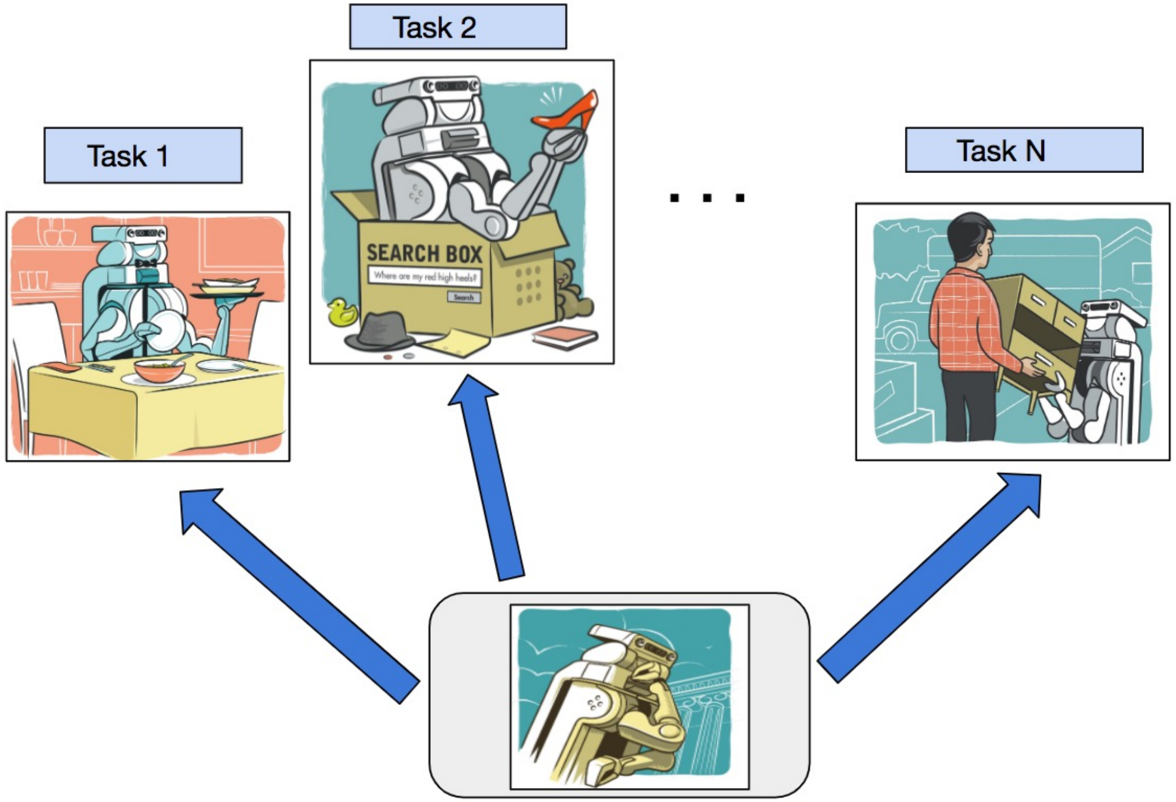
$$p(\mathcal{M}_i)$$

$$\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{T}_i, \mathcal{R}_i, \mu, \gamma)$$





Why should we do this?



- Learn faster by sharing data
- Generalize immediately (or quickly) to new, unseen tasks

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

↓

Multi-Task Reinforcement Learning

↓

Meta-Reinforcement Learning

Multi-Task Meta-MDP

Let us assume the factor of variation across MDPs can be characterized by known ω_i

Eg: task ID, goal, video, language, …

$$p(\omega_i)$$

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mu, \gamma) \qquad \mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{T}_{\omega_i}, \mathcal{R}_{\omega_i}, \mu, \gamma)$$

$\Downarrow$ Slight reformulation

$$s \rightarrow (s, \omega_i)$$

$$\mathcal{T} \rightarrow p(s' | s, a, \omega_i)$$

$$\mathcal{R} \rightarrow r(s, a, \omega_i)$$

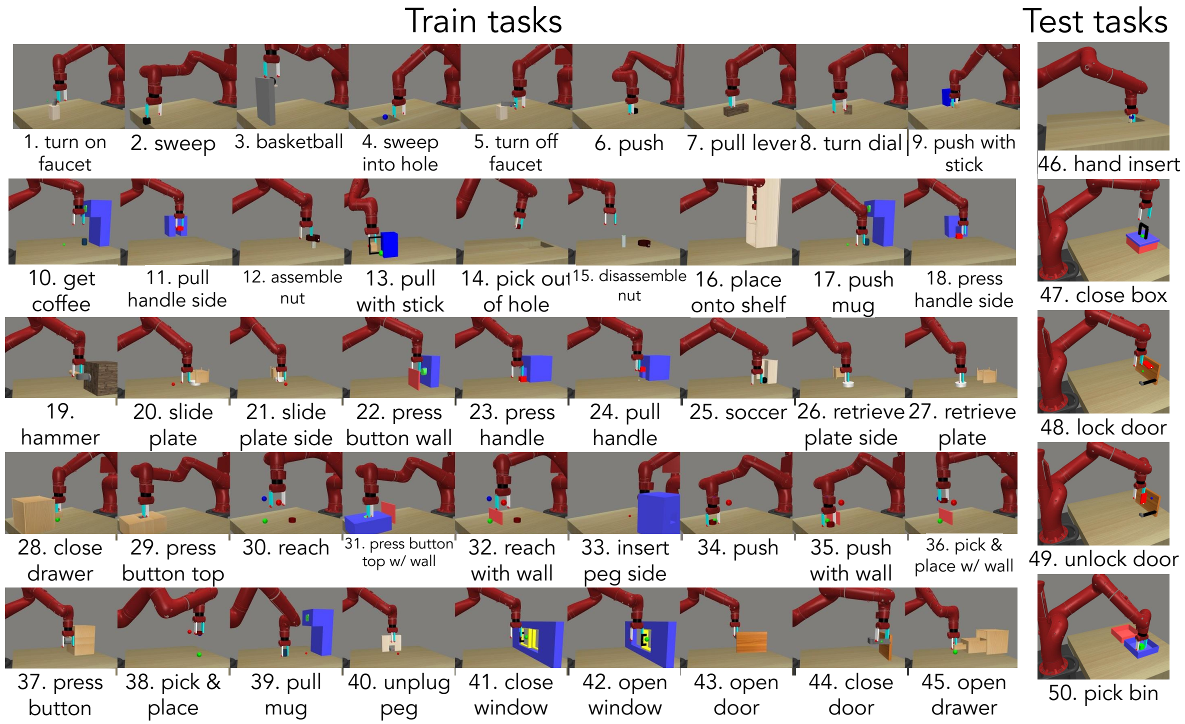
$$\mu \rightarrow \mu(s_0) p(\omega_i)$$

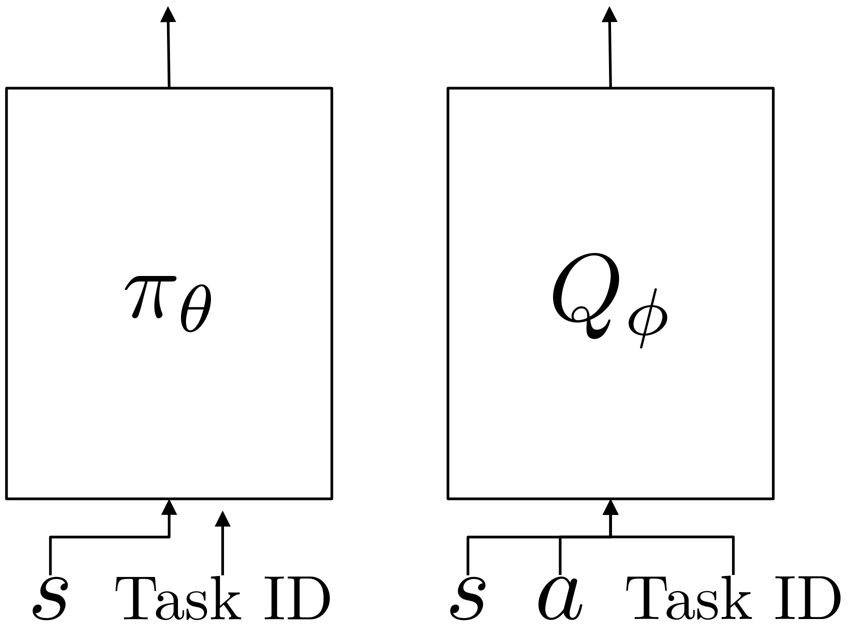
Key idea: Multi-task RL == Single task RL in modified MDP

Just include ω_i in state and run standard RL, solve new ω_i 0-shot

Multi-Task Actor-Critic

We often want to learn a single policy, Q function which can solve multiple tasks.





$$\max_{\phi} \mathbb{E}_{\omega \sim p(\omega)} \left[\mathbb{E}_{\pi_\phi(a|s,\omega)} \left[\sum_t \gamma^t r(s, a, \omega) \right] \right]$$

Template for Multi-Task RL

Canonical paradigm for doing multi-task RL via RL

1. Sample data from all tasks using the same actor with different task ID
2. Collect all data into a single batch with (s, a, s’, task ID) pairs
3. Perform actor and critic updates on the shared actor and critic with losses summed up across tasks

Actor

$$\pi \leftarrow \arg\max \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{a \sim \pi} \left[Q^{\pi}(s, a, \tau) \right]$$

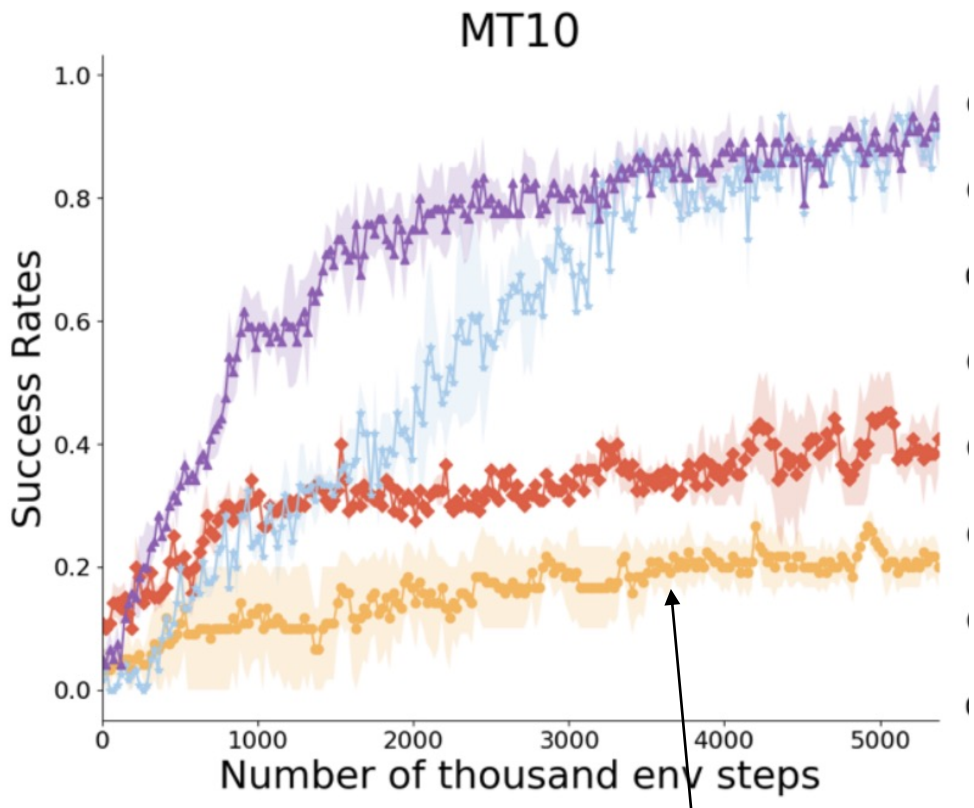
Critic

$$Q^{\pi} \leftarrow \arg\min \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{(s,a,s') \sim p} \left[\left(Q(s, a, \tau) - \left(r(s, a, \tau) + \gamma \mathbb{E}_{a' \sim \pi(.|s', \tau)} Q(s', a', \tau) \right) \right)^2 \right]$$

Does it work?

Let’s not even study generalization, let’s understand if this fits the train set

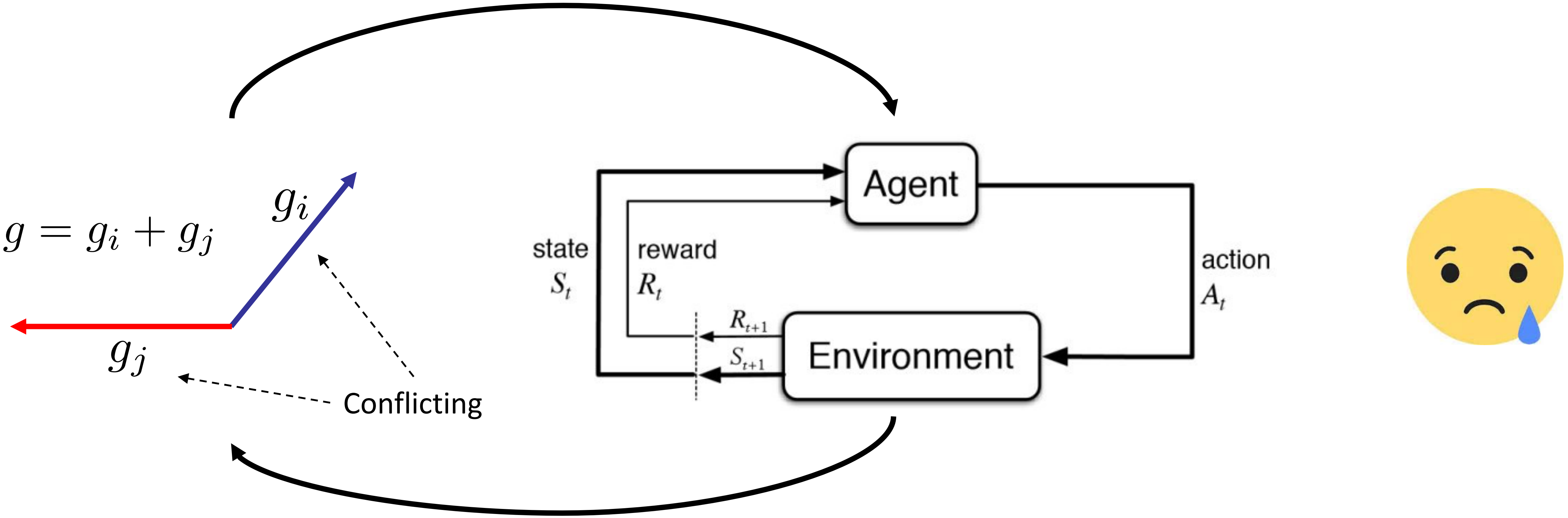
Methods	MT50
Multi-task PPO	8.98%
Multi-task TRPO	22.86%
Task embeddings	15.31%
Multi-task SAC	28.83%
Multi-task multi-head SAC	**35.85%**



Multi Task RL

Why is it hard to do Multi-Task RL?

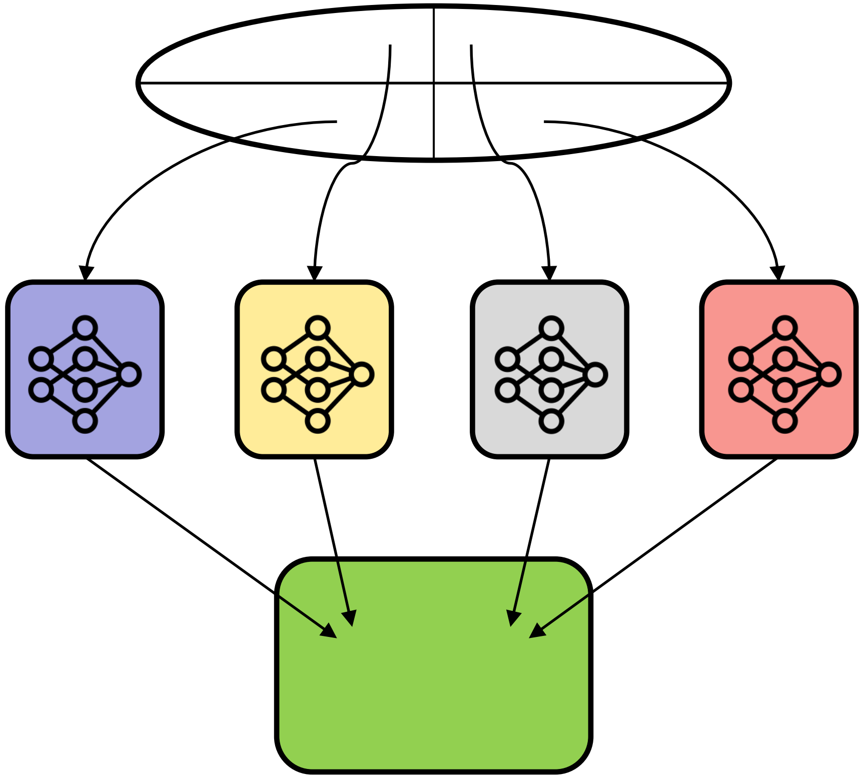
Gradients from different tasks often conflict and hamper performance of all tasks, especially when coupled with exploration



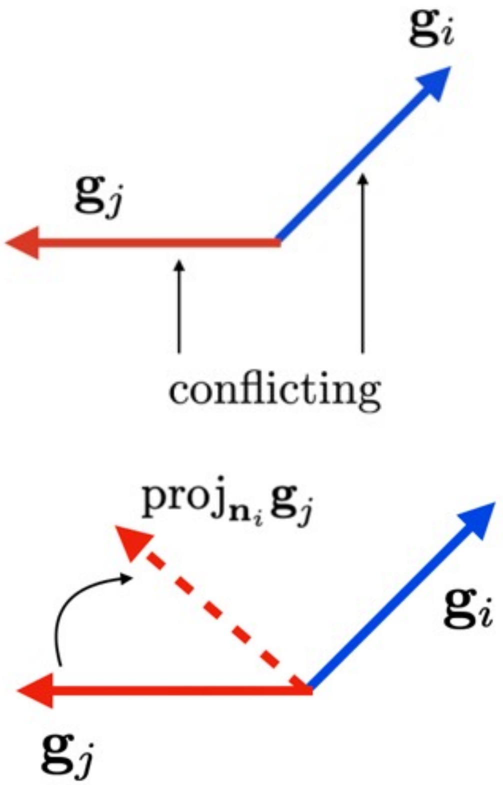
How can we deal with gradient interference in RL?

If issue is exploration + conflicting gradients is bad

Idea 1: Remove exploration from MTRL



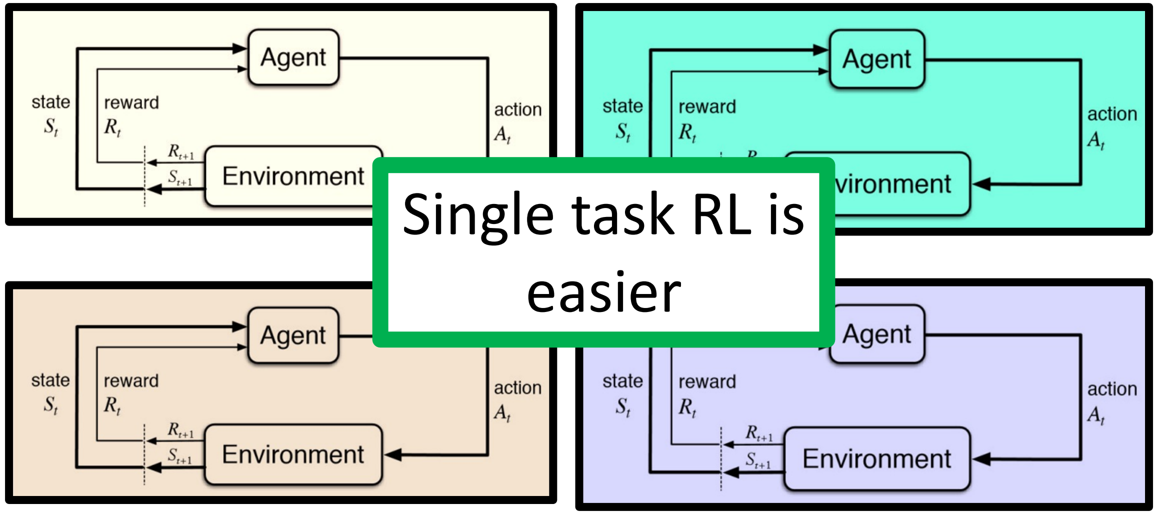
Idea 2: Modify gradients

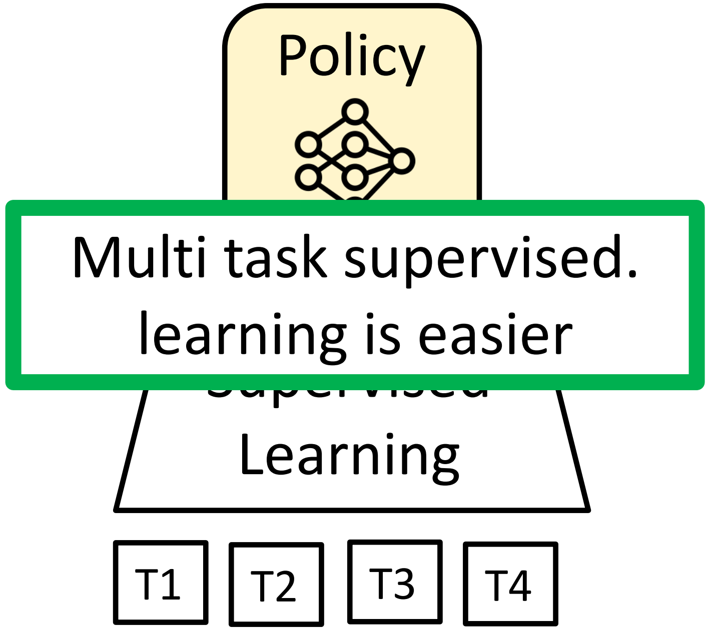


Resolving Gradient Interference with Distillation

Empirical observation:

Multi-task SL (no exploration) is stable, multi-task RL (exploration) is unstable

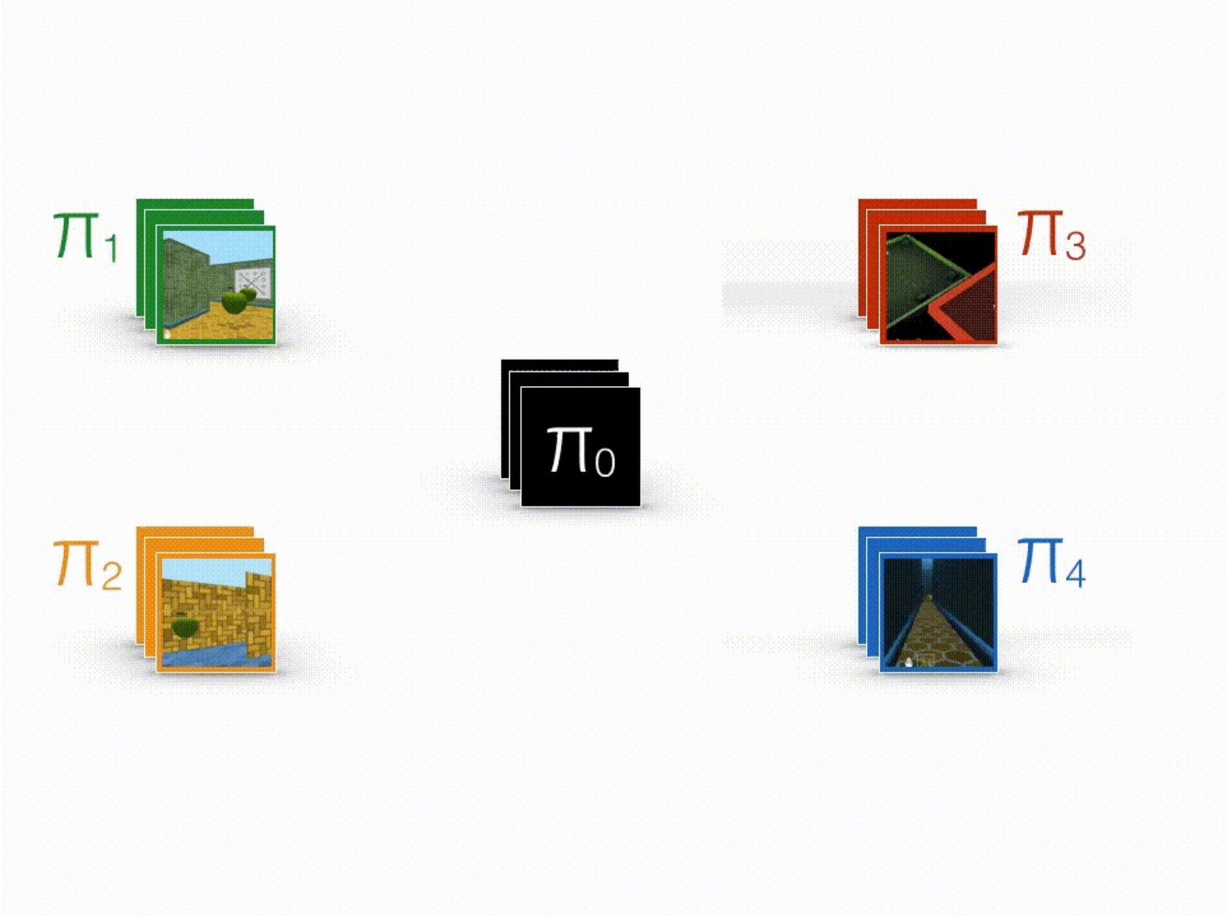




Idea: convert multi-task RL into single task RL + multi task SL

Divide and Conquer Approach to RL

Divide into multiple single task RL problems, "distill" into a single solution



Single task RL → standard RL

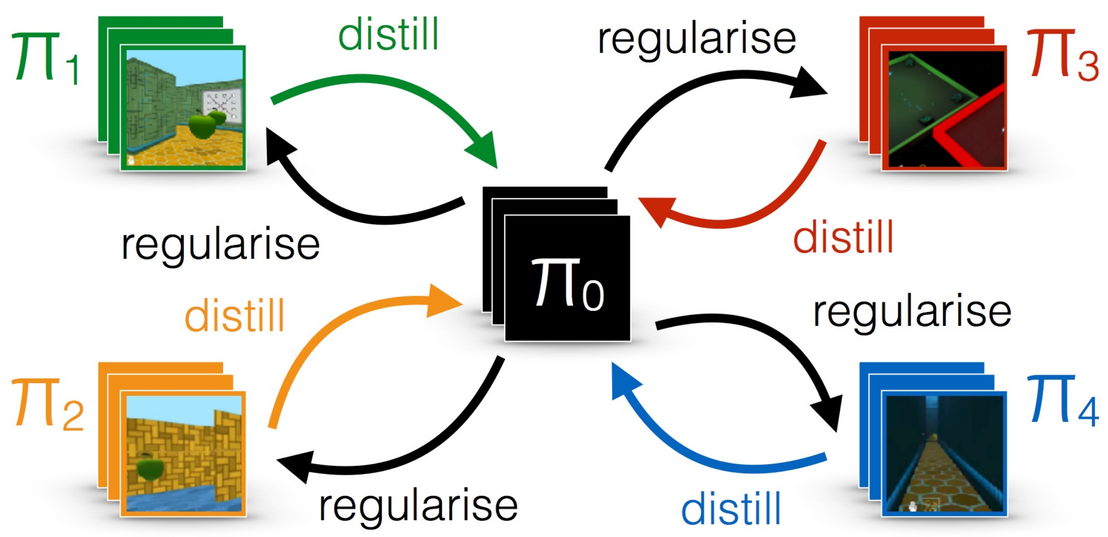
Distillation → supervised learning

Distral: Robust multi-task RL Teh et al 2017

Divide and Conquer RL: Mathematical Formulation

Shared policy   Per-task policy   Per-task reward   Match shared/individual policy   Entropy

$$J(\pi_0, \{\pi_i\}_{i=1}^n) = \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0} \gamma^t R_i(a_t, s_t) - c_{\mathrm{KL}}\gamma^t \log \frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} - c_{\mathrm{Ent}}\gamma^t \log \pi_i(a_t|s_t)\right]$$



Single task RL → standard RL

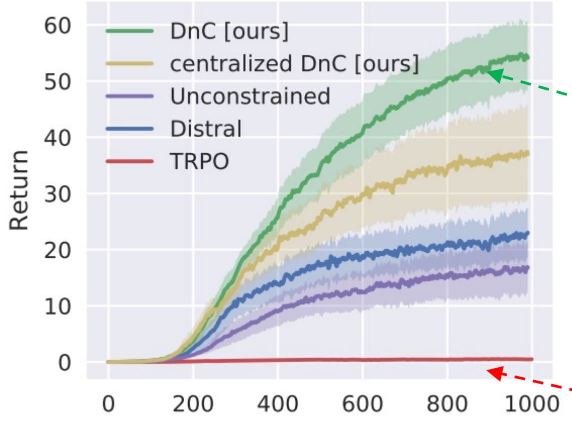
$$\max_{\pi_i} J(\pi_0, \{\pi_i\}_{i=1}^n)$$

Distillation → supervised learning

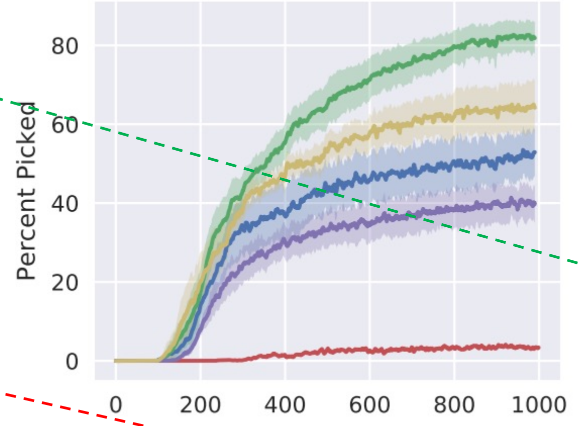
$$\max_{\pi_0} J(\pi_0, \{\pi_i\}_{i=1}^n)$$

Experimental Validation

(a) *Picking task*



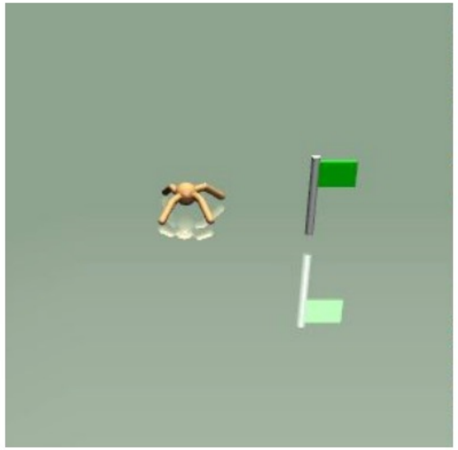
(b) Average Return on *Picking*



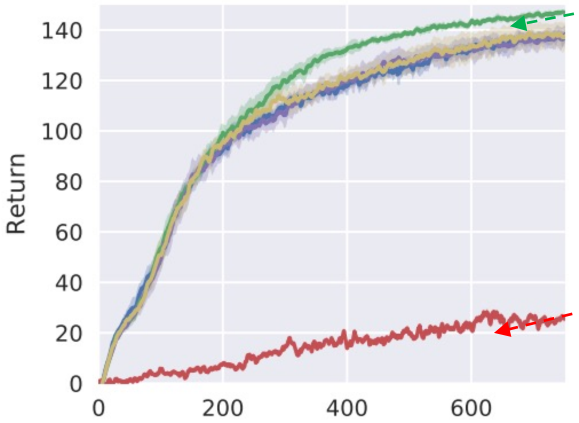
(c) Success rate on *Picking*

Divide and Conquer

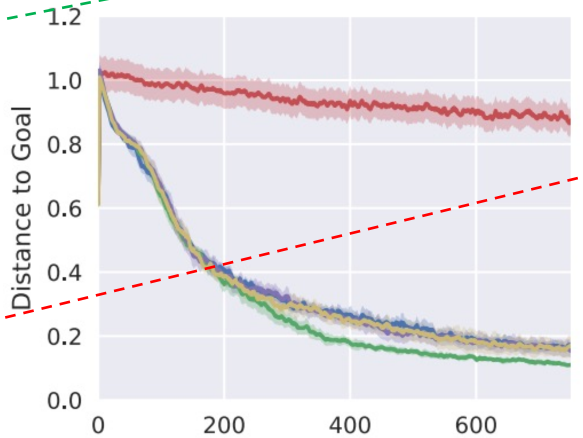
Monolithic RL



(j) *Ant*



(k) Average Return on *Ant*



(l) Distance from goal on *Ant*

Divide and Conquer RL, Ghosh et al 2018

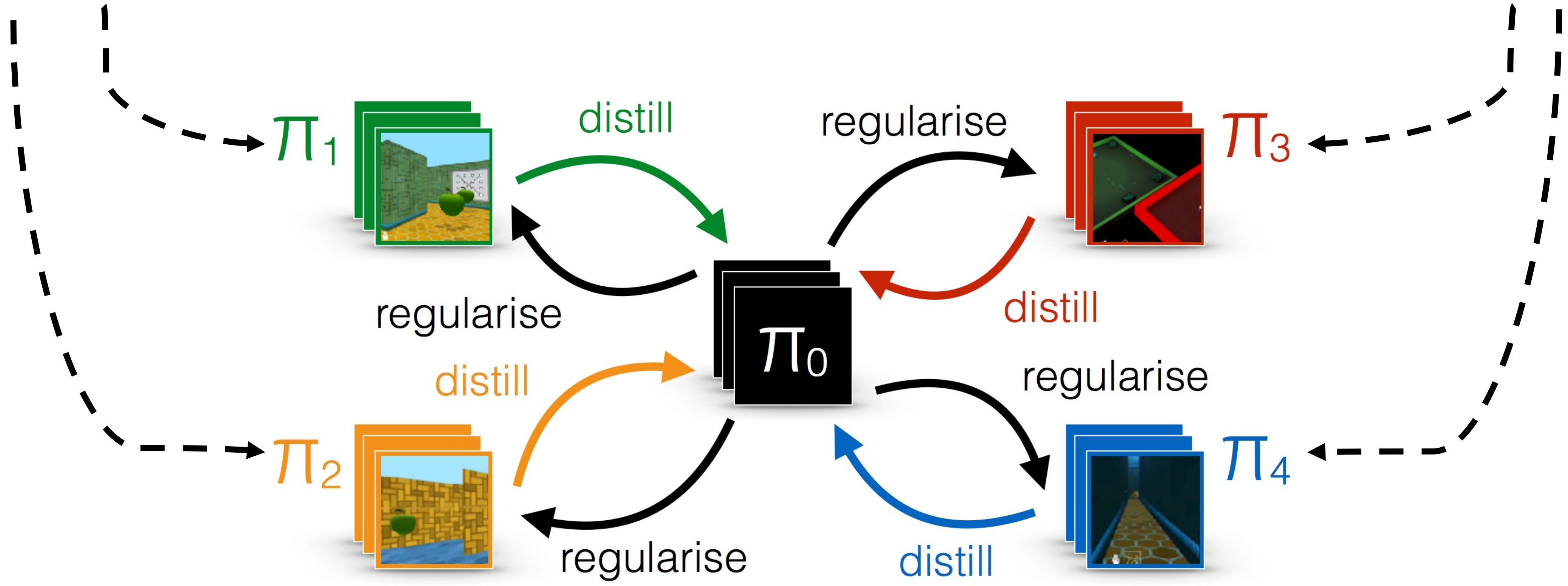
Experimental Validation

Divide and Conquer
Reinforcement
Learning

Divide and Conquer RL, Ghosh et al 2018

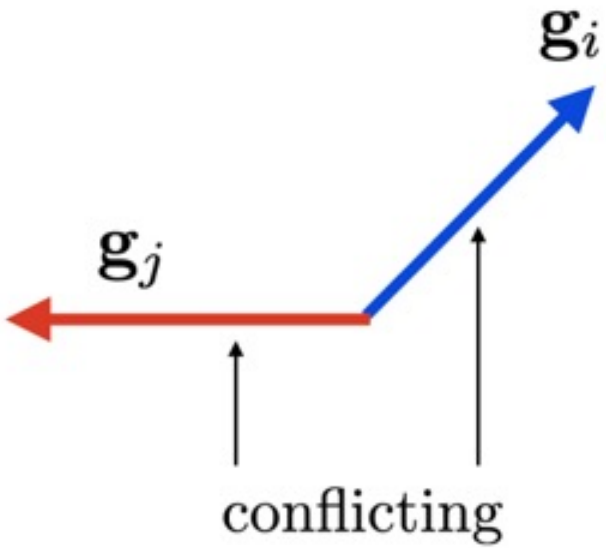
Is this enough?

Lot of the learning is done independently, limited data/parameter sharing



Can we do better?

What if we directly modified the gradients?



Replace g_i by g_i'

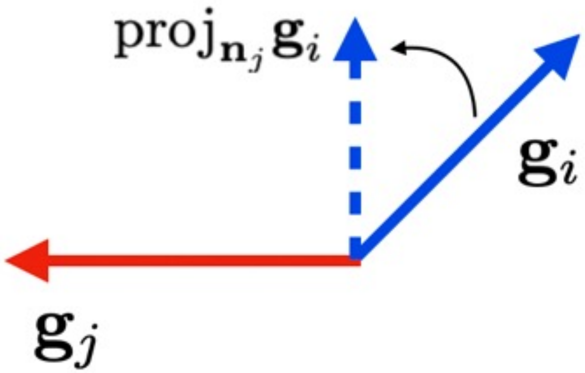
Replace g_j by g_j'

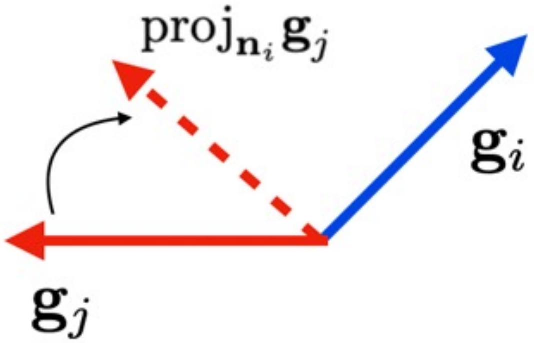
What should g_i' and g_j' be?

Idea: When gradients conflict, project them to deconflict

Deconflicting gradients with PCGrad

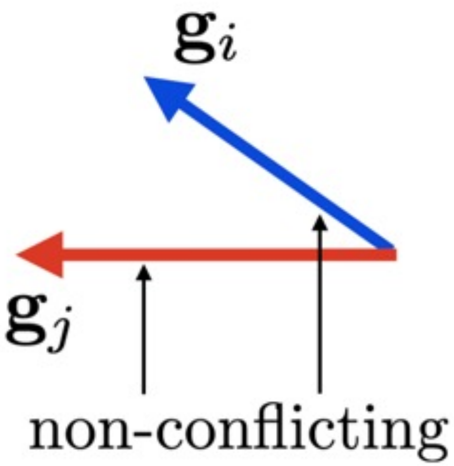
If gradients conflict: project them onto the normal plane





$$g_i = g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} \cdot g_j$$

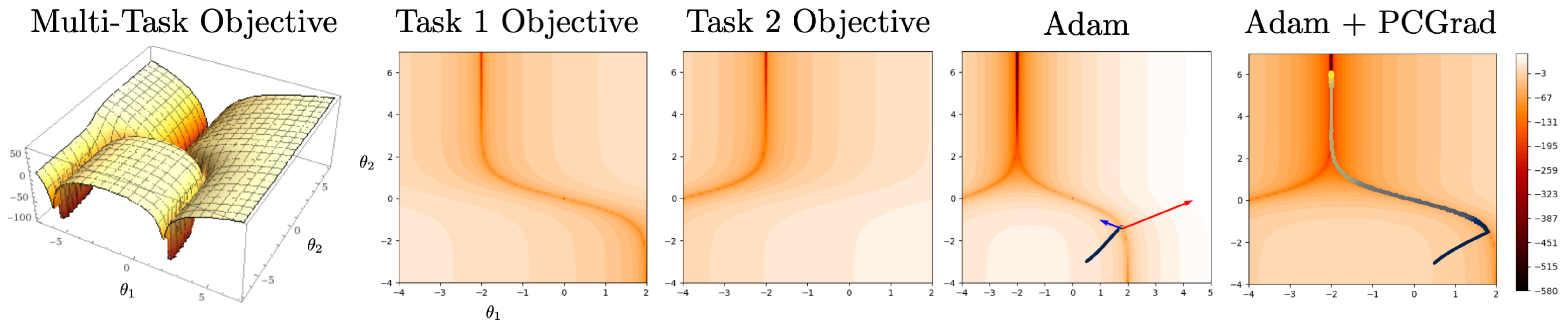
Otherwise: leave them alone



Gradient Surgery for Multi-Task RL, Yu et al 2018

Does this empirically help?

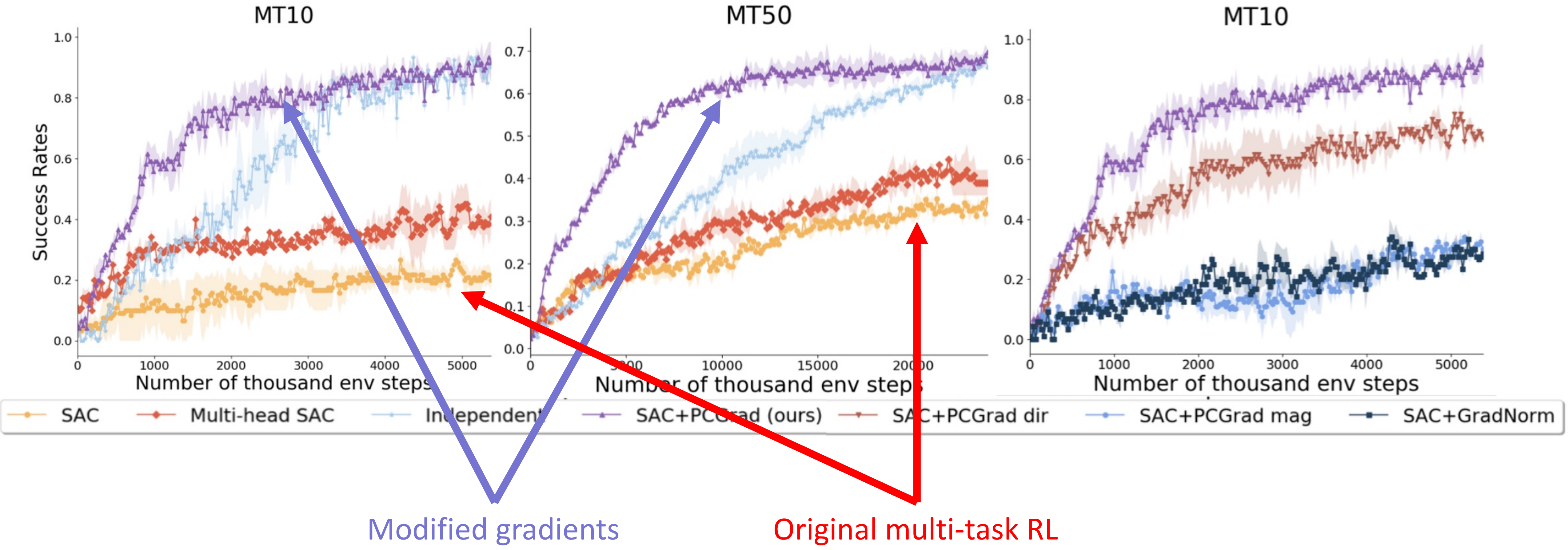
$$\mathcal{L}_1(\theta) = 20\log(\max(|.5\theta_1 + \tanh(\theta_2)|, 0.000005))$$
$$\mathcal{L}_2(\theta) = 25\log(\max(|.5\theta_1 - \tanh(\theta_2) + 2|, 0.000005))$$



(a) (b) (c) (d) (e)

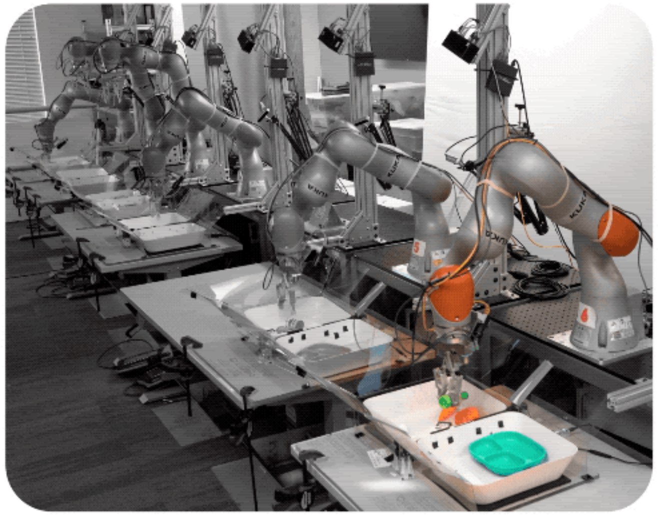
Gradient Surgery for Multi-Task RL, Yu et al 2018

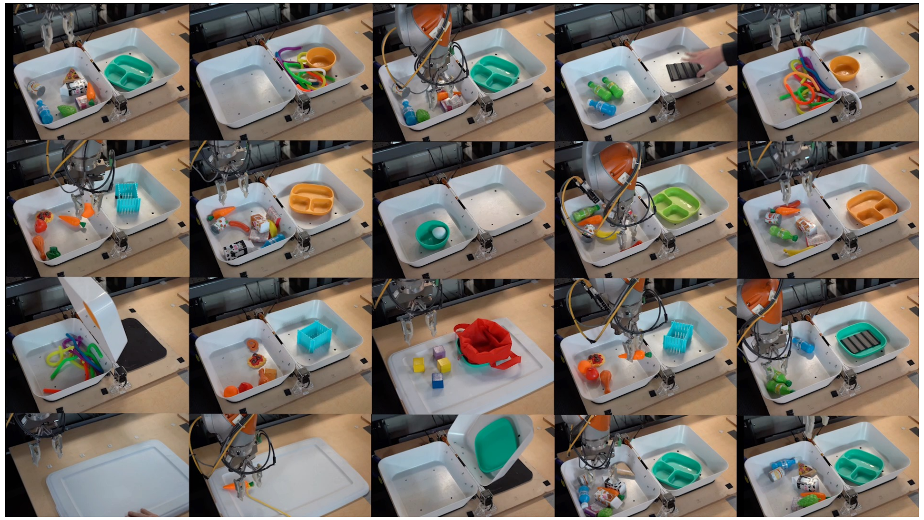
Does this empirically help?

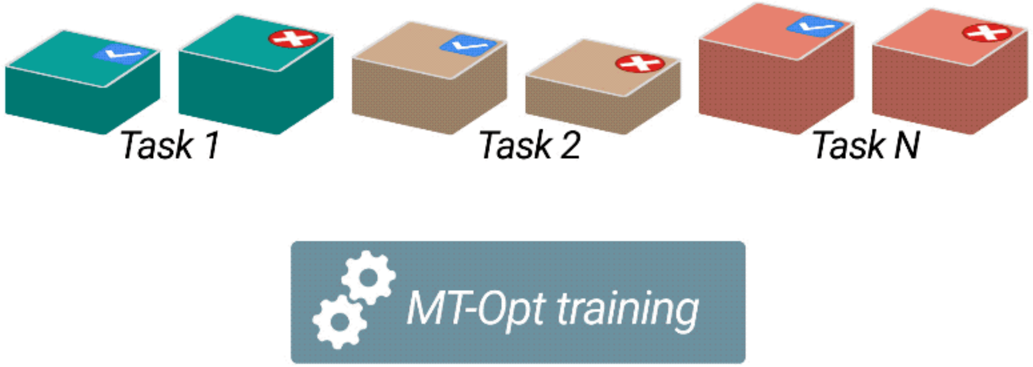


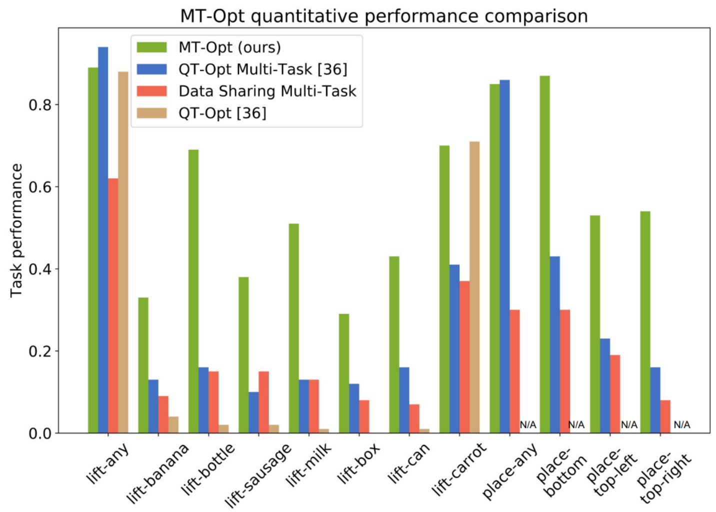
Gradient Surgery for Multi-Task RL, Yu et al 2018

So multi-task RL is pretty cool, does it work?









Mt-Opt: Kalashnikov et al 2021

So multi-task RL is pretty cool, does it work?

ω_i can be language too!

Do As I Can, Not as I Say: Ahn et al 2022

Takeaways

1. Multi-task RL solves a contextual meta-MDP for 0-shot generalization
   - Can help with efficiency and generalization
2. Optimization in multi-task RL can be challenging:
   - Gradient interference during optimization
   - Winner take all during optimization
3. Solutions to multi-task optimization include:
   - Divide and conquer
   - Gradient projection
   - …

Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

↓

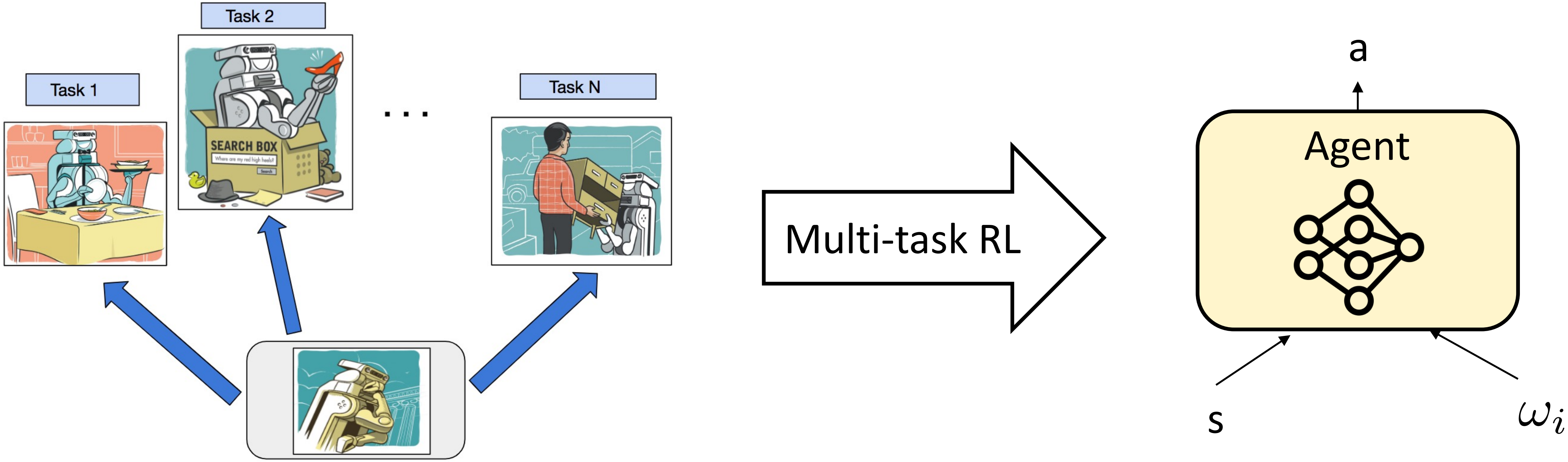
Multi-Task Reinforcement Learning

↓

Meta-Reinforcement Learning

Recap: Multi-task RL Setup, 0-shot generalization

Factor of variation across MDPs can be characterized by ω_i, which is known
Eg: task ID, goal, video, language, …



Good 0-shot performance

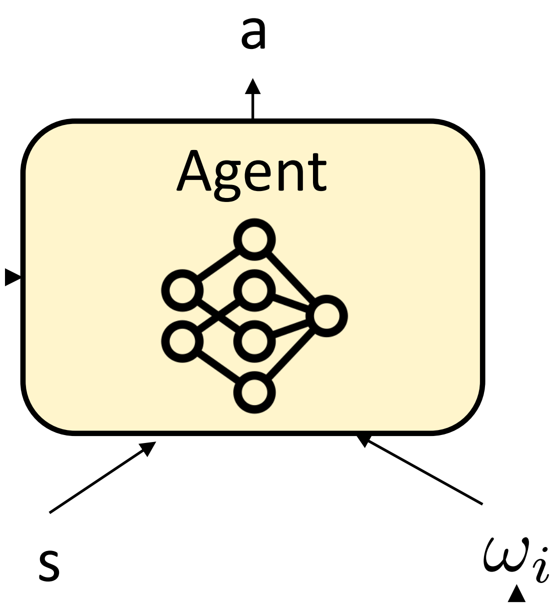
When is this not enough?

Visual from C Finn

From 0–shot learning to few-shot learning

Factor of variation across MDPs can be characterized by ω_i, which is known
Eg: task ID, goal, video, language, …

Policy is not good enough 0-shot, need to finetune

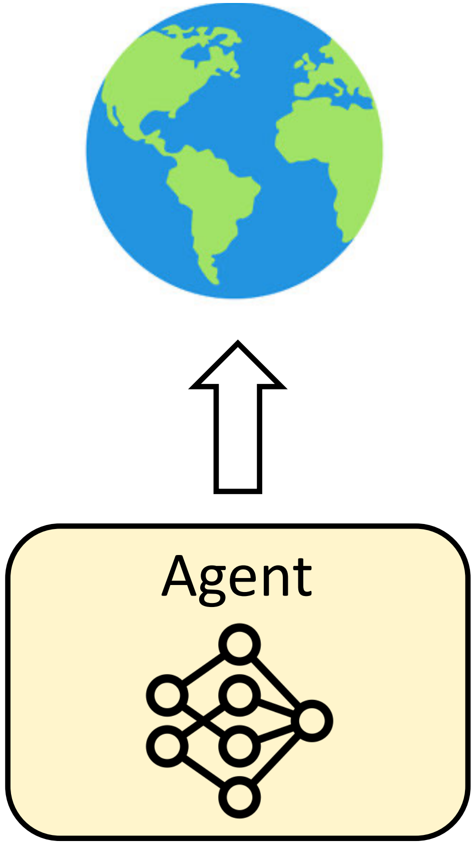


Context is unknown or hard to specify analytically

When is this not enough?

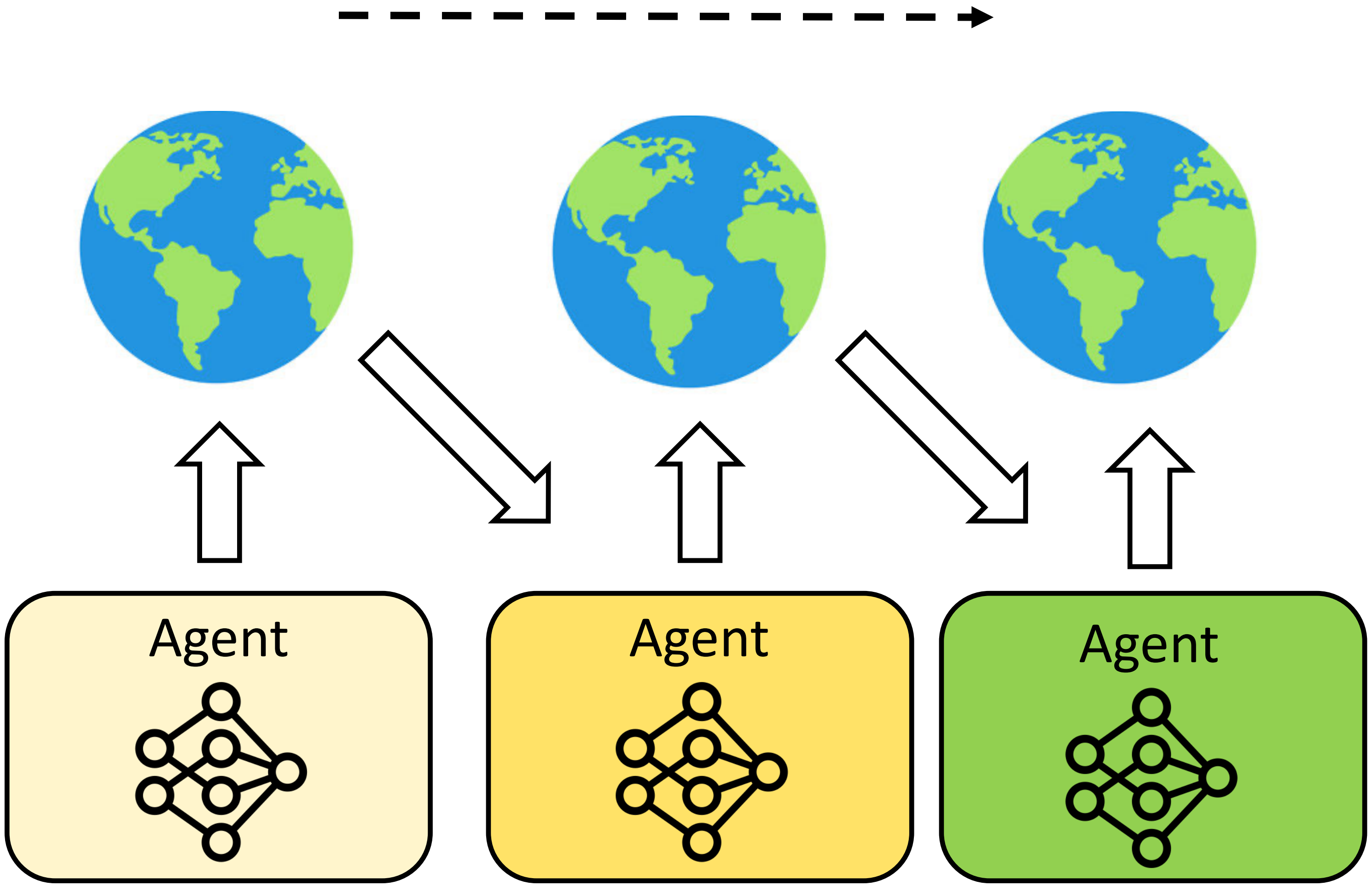
From 0–shot learning to few-shot learning

0-shot MTRL: No experience at test time

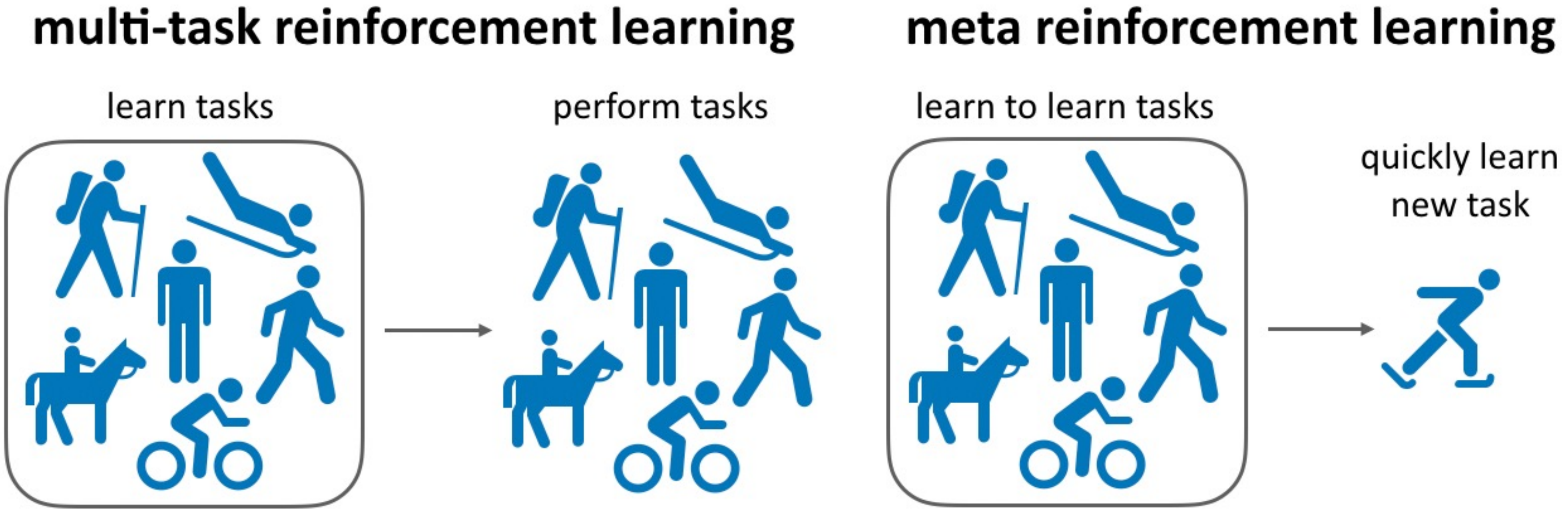


Meta-RL: Small amount of experience at test time

Fast adaptation with experience

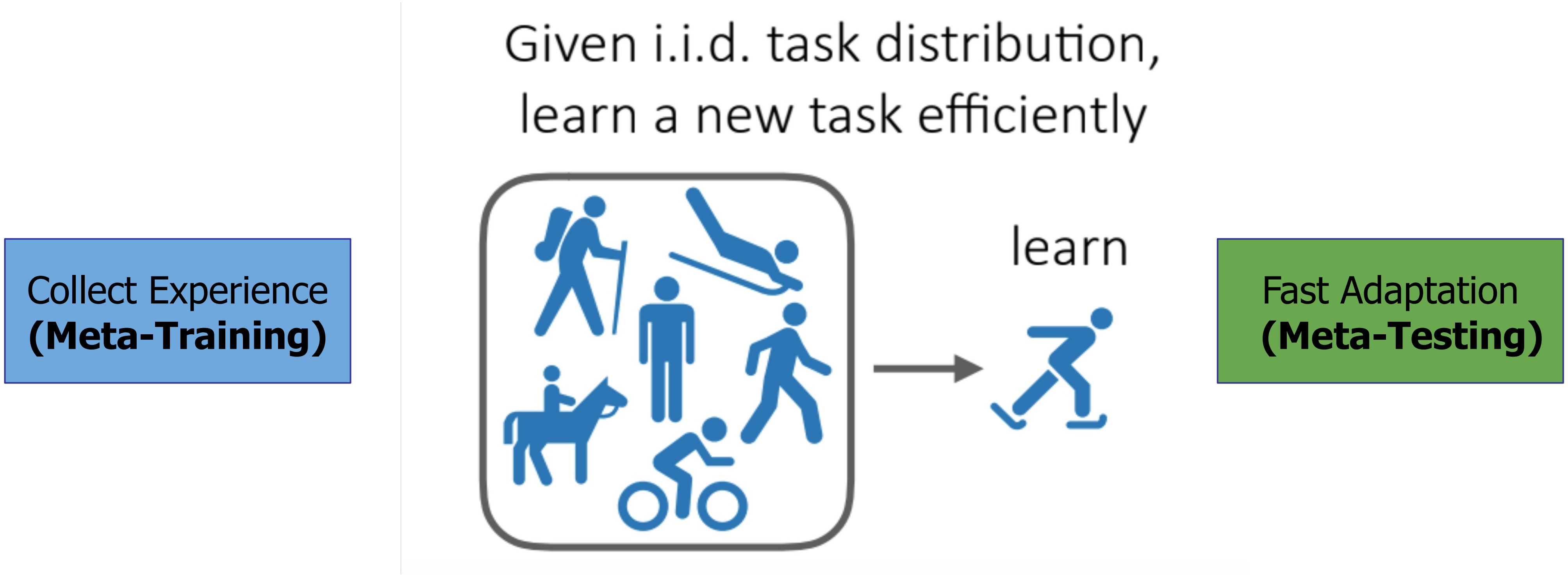


Connection to Contextual Multi-Task RL



- Multi-task policy evaluates 0-shot performance
- Meta-RL trains for good k-shot policy by ”learning to learn”

Meta-Learning Problem for RL



- Given a distribution over tasks $p(\tau)$, learn an update function f_θ that can learn tasks drawn from $p(\tau)$ quickly!
- Leverage regularity across tasks to optimize for a fast RL algorithm

Visual from C Finn

Meta-Learning Problem for RL

Standard RL:

Single reward function, single dynamics $\arg\max_{\theta} \mathbb{E}_{\pi_\theta}\left[\sum_t r(s_t, a_t)\right]$

Meta RL:

Distribution of tasks $p(\tau)$, optimize for update function f_θ

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\tau\sim p(\tau)}\left[\mathbb{E}_{\pi_{\phi_i}}\left[\sum_t r(s_t, a_t)\right]\right]$$

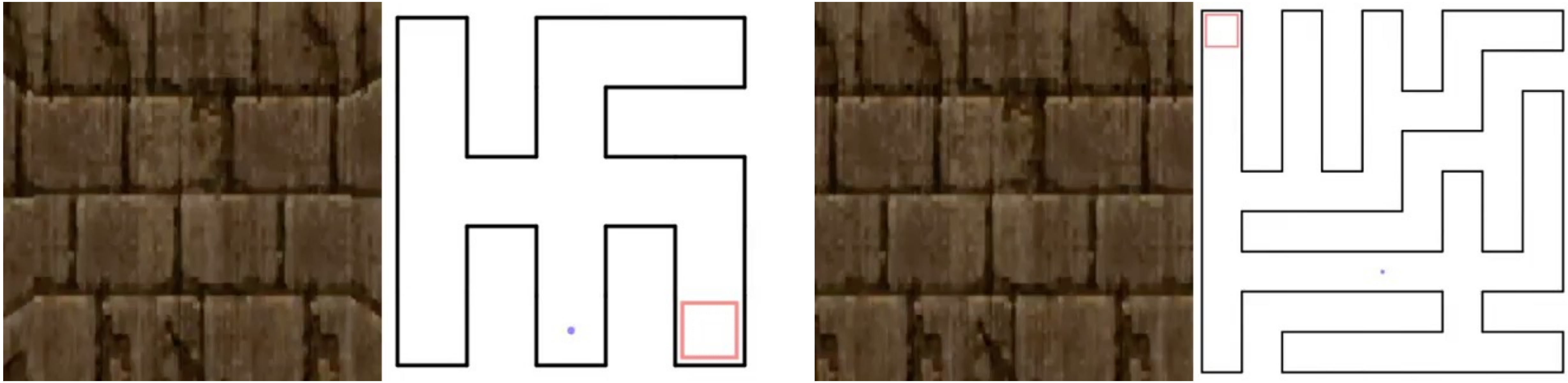
Encourages quick update

Per-task updated policy

where $\phi_i = f_\theta(\mathcal{D}_\tau)$

Shared update function

Intuition behind Meta-RL



- Leverage regularity in task distribution to speed up learning
- Explore for some time before exploiting
- Minimizes regret not just maximizes reward

Duan et al 2016

General Structure of Meta-RL Algorithms

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)} \left[\mathbb{E}_{\pi_{\phi_i}} \left[\sum_t r(s_t, a_t) \right] \right]$$ ← Outer loop

where $\phi_i = f_\theta(\mathcal{D}_\tau)$ ← Inner loop

1. Sample a batch of tasks from $p(\tau)$
2. collect data <u>pre-update</u>
3. Compute update according to $\phi_i = f_\theta(\mathcal{D}_\tau)$
4. Sample data from ϕ_i post-update to evaluate the update
5. Optimize for update function f_θ

Solution Techniques for Meta-RL Problems

Main design choices:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)} \left[\mathbb{E}_{\pi_{\phi_i}} \left[\sum_t r(s_t, a_t) \right] \right] \longleftarrow \text{Outer loop}$$

where $\phi_i = f_\theta(\mathcal{D}_\tau) \longleftarrow$ Inner loop

- Parameterization of f_θ for inner loop
- Algorithm for outer loop optimization

<u>Inner loop</u>

Memory based ⟷ Latent Variable ⟷ Gradient Descent

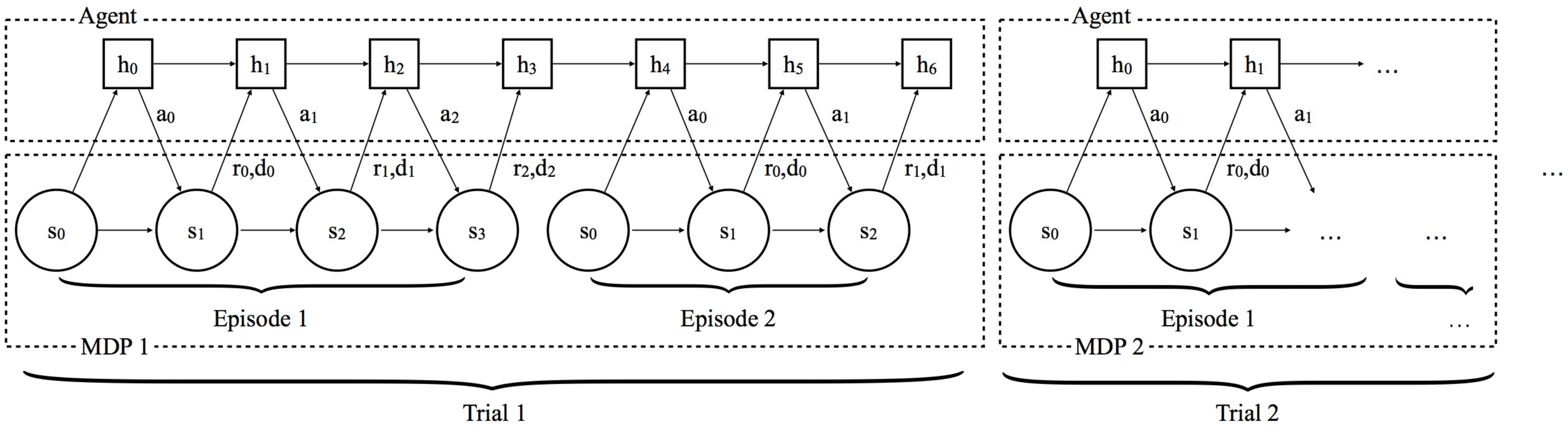
<u>Outer loop</u>

Policy Gradient ⟷ Off-Policy RL ⟷ Model-Based RL

Memory Based Meta-RL

Idea: Make the update function forward pass of an RNN

- Learn RNN that takes in past s, a, **r(s, a)**, produce action.
- Maintain hidden state across episodes
- Maximize sum of returns across episodes



RL[2] Duan et al, Learning to RL Wang et al

Memory Based Meta-RL

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)} \left[\mathbb{E}_{\pi_{\phi_i}} \left[\sum_t r(s_t, a_t) \right] \right]$$

where $\phi_i = f_\theta(\mathcal{D}_\tau)$

Combine inner and outer loop into black box RNN

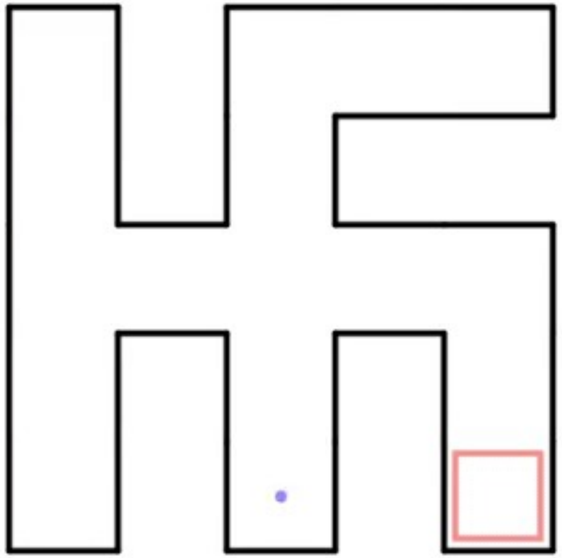
<u>Meta-Training</u>

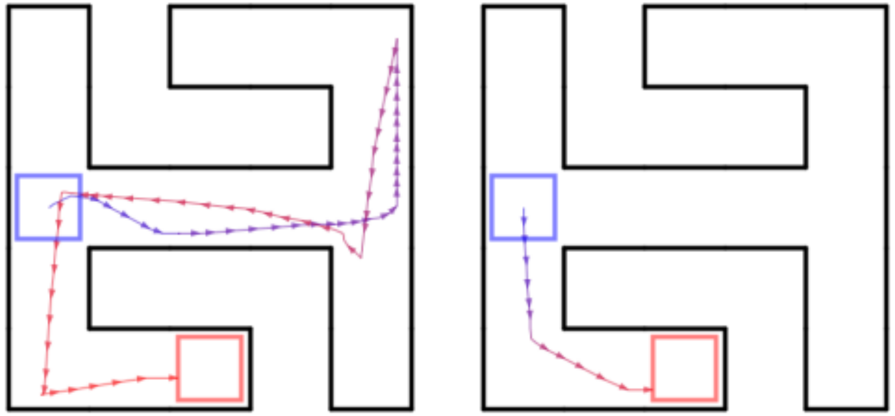
1. Sample a batch of tasks from $p(\tau)$
2. Collect data using RNN across episodes for each task, with persistent hidden state and rewards available to the policy
3. Optimize RNN policy via policy gradient BPTT

<u>Meta-Testing</u>

1. Simply run the RNN forward pass across episodes

Memory Based Meta-RL





(a) Good behavior, 1st episode (b) Good behavior, 2nd episode

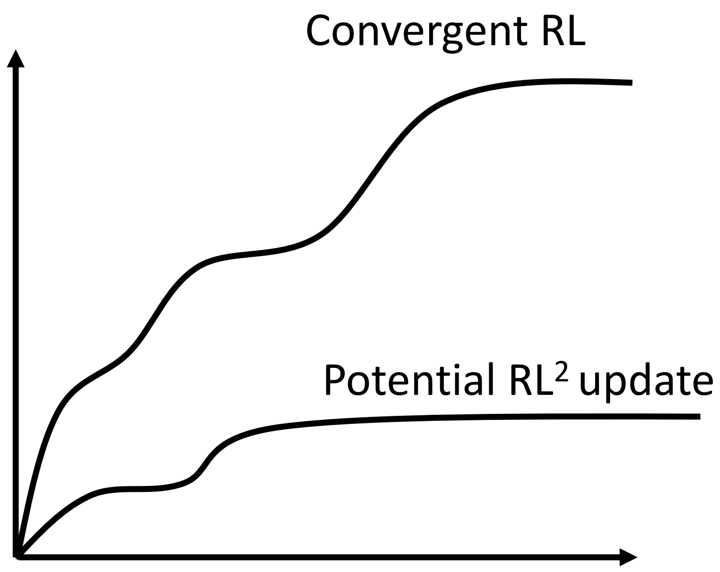
How well does memory based meta-RL work?

Pros:

Simple, easy to implement

Arbitrarily flexible inner loop

Generally stable optimization



Cons:

No guaranteed improvement during meta-test time

Poor performance OOD

Optimization Based Meta-RL

Idea:

What if we force $f(\theta)$ to be convergent?

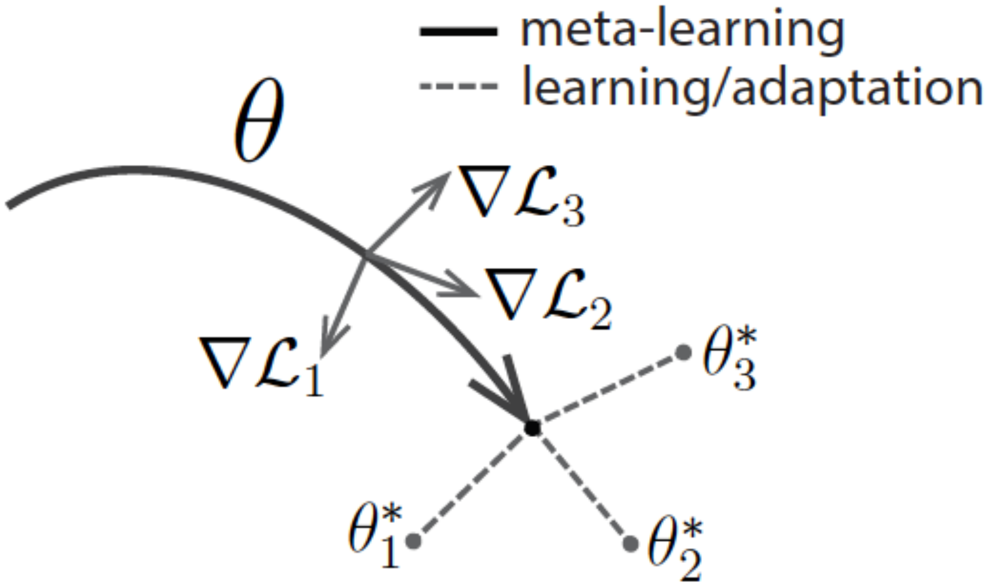
Force $f(\theta)$ to be a convergent optimization algorithm like SGD

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)} \left[\mathbb{E}_{\pi_{\phi_i}} \left[\sum_t r(s_t, a_t) \right] \right]$$

$$\phi_i = f_\theta(\mathcal{M}_i)$$

Restrict to be convergent optimization

MAML: Gradient Based Meta-RL



$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\tau\sim p(\tau)}\left[\mathbb{E}_{\pi_{\phi_i}}\left[\sum_t r_\tau(s_t, a_t)\right]\right]$$

$$\phi_i = \theta + \alpha\nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_t r_\tau(s_t, a_t)\right]$$

Learn most fine-tunable initial parameters, such that 1-step of SGD is good

Pseudocode for Gradient Based RL

1. Sample a batch of tasks from $p(\tau)$
2. collect data <u>pre-update</u> from π_θ
3. Compute update according to $\phi_i = \theta + \alpha \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_t r_\tau(s_t, a_t)\right]$
4. Sample data from ϕ_i post-update
5. Optimize for initial parameters by PG in outer loop

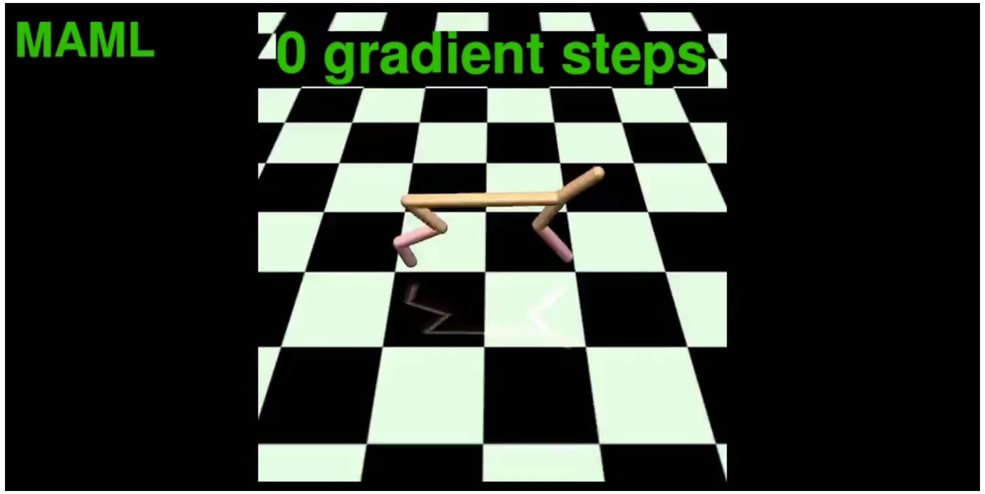
$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)}\left[\mathbb{E}_{\pi_{\phi_i}}\left[\sum_t r_\tau(s_t, a_t)\right]\right]$$

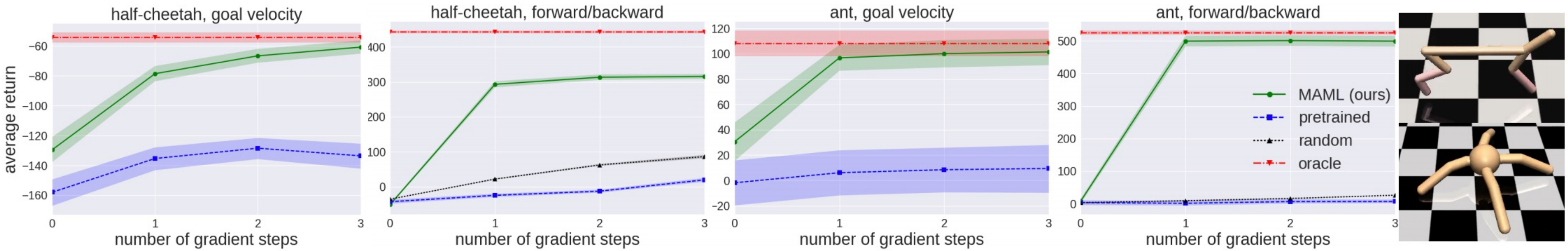
$$\phi_i = \theta + \alpha \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_t r_\tau(s_t, a_t)\right]$$

Second order gradients via bi-level optimization

How well does it work?

<u>Tasks:</u>
Half cheetah: goal velocity,
Half cheetah: forward/backward
Ant: forward/backward





Finn et al 2017

How well does it work?

Pros:

Consistent, worst case performance is PG

Only need to learn initialization

Cons:

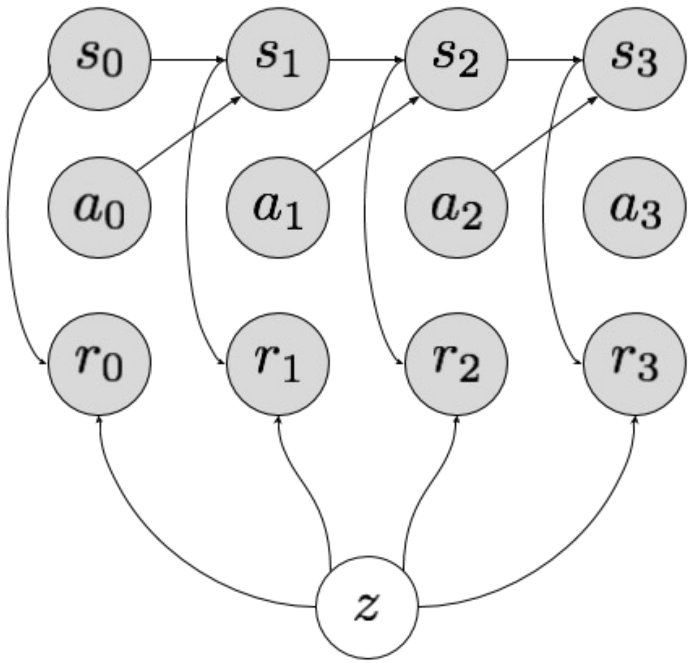
Second order gradients needed

Potentially less expressive update

Latent Variable Models for Meta-RL

Think of meta-RL similar to multi-task RL, but context ω_i is a hidden variable that must be inferred

Meta-RL as a POMDP



Need to infer this

Recasting meta-RL as context inference

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)} \left[\mathbb{E}_{\pi_{\phi_i}} \left[\sum_t r(s_t, a_t) \right] \right]$$

where $\phi_i = f_\theta(\mathcal{D}_\tau)$

Infer latent variable from experience

$$q_\theta(z | s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$$

Deploy latent conditioned policy

$$\pi_\theta(a | s, z)$$

Recasting meta-RL as context inference

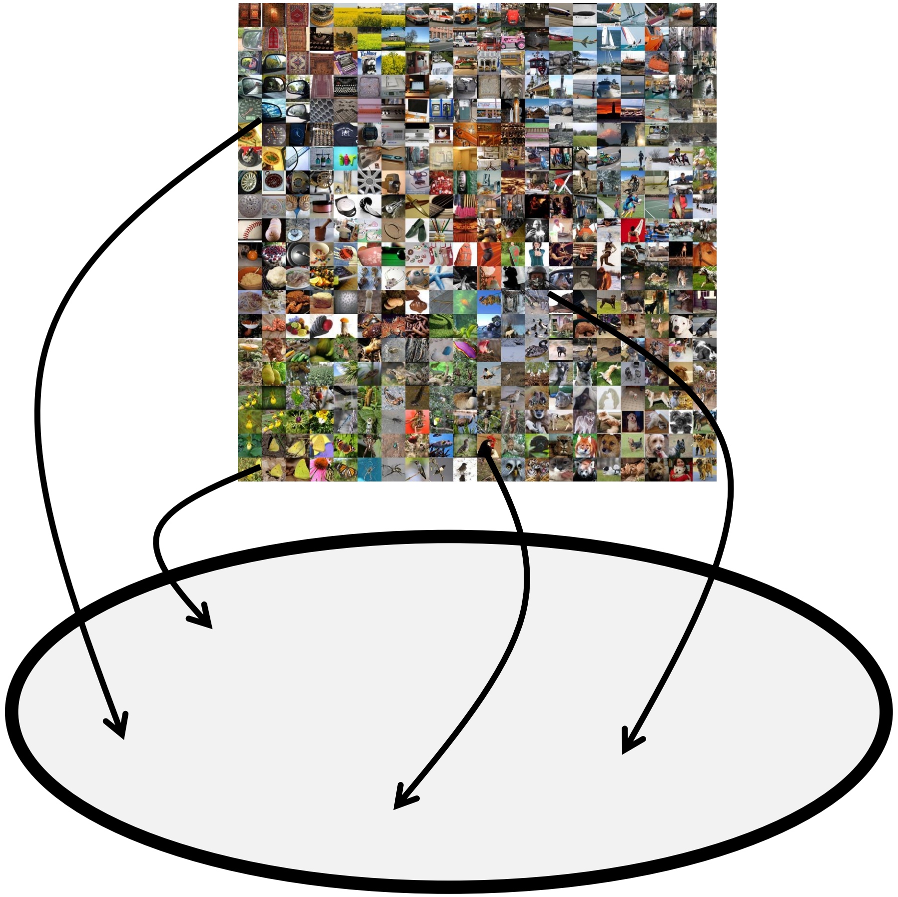
<u>Meta-Training</u>

1. Sample a batch of tasks from $p(\tau)$
2. Sample trajectories $\{s_0, a_0, r_0, \ldots, s_T, a_T, r_T\}_{I=1}^{N}$
3. Train $q_\theta(z|s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ and $\pi_\theta(a|s, z)$ to maximize rewards via RL (+ some regularization)

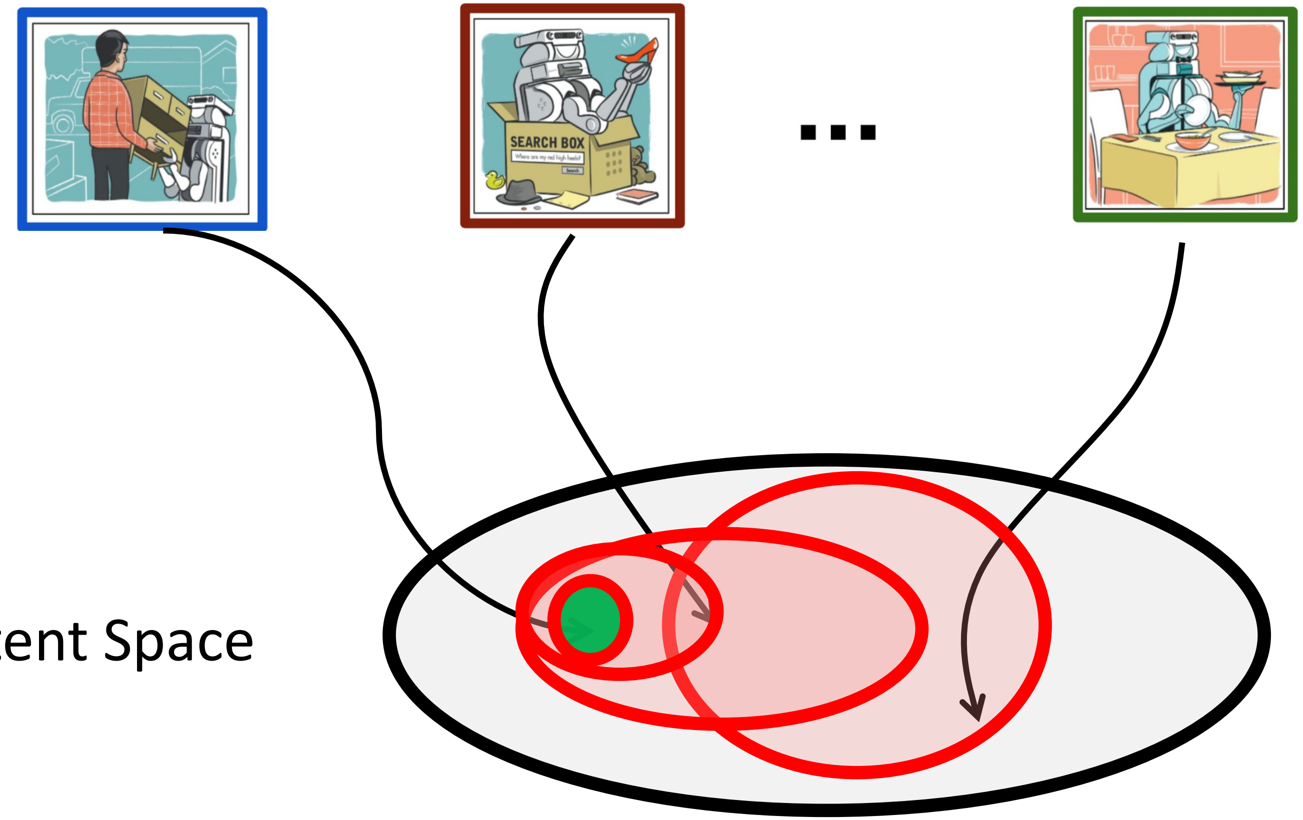
<u>Meta-Testing</u>

1. Sample z from prior $p(z)$
2. Sample trajectories from $\pi_\theta(a|s, z)$ and z
3. Update $p(z)$ to posterior $q_\theta(z|s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$

Latent Variable Model Intuition



Latent Space

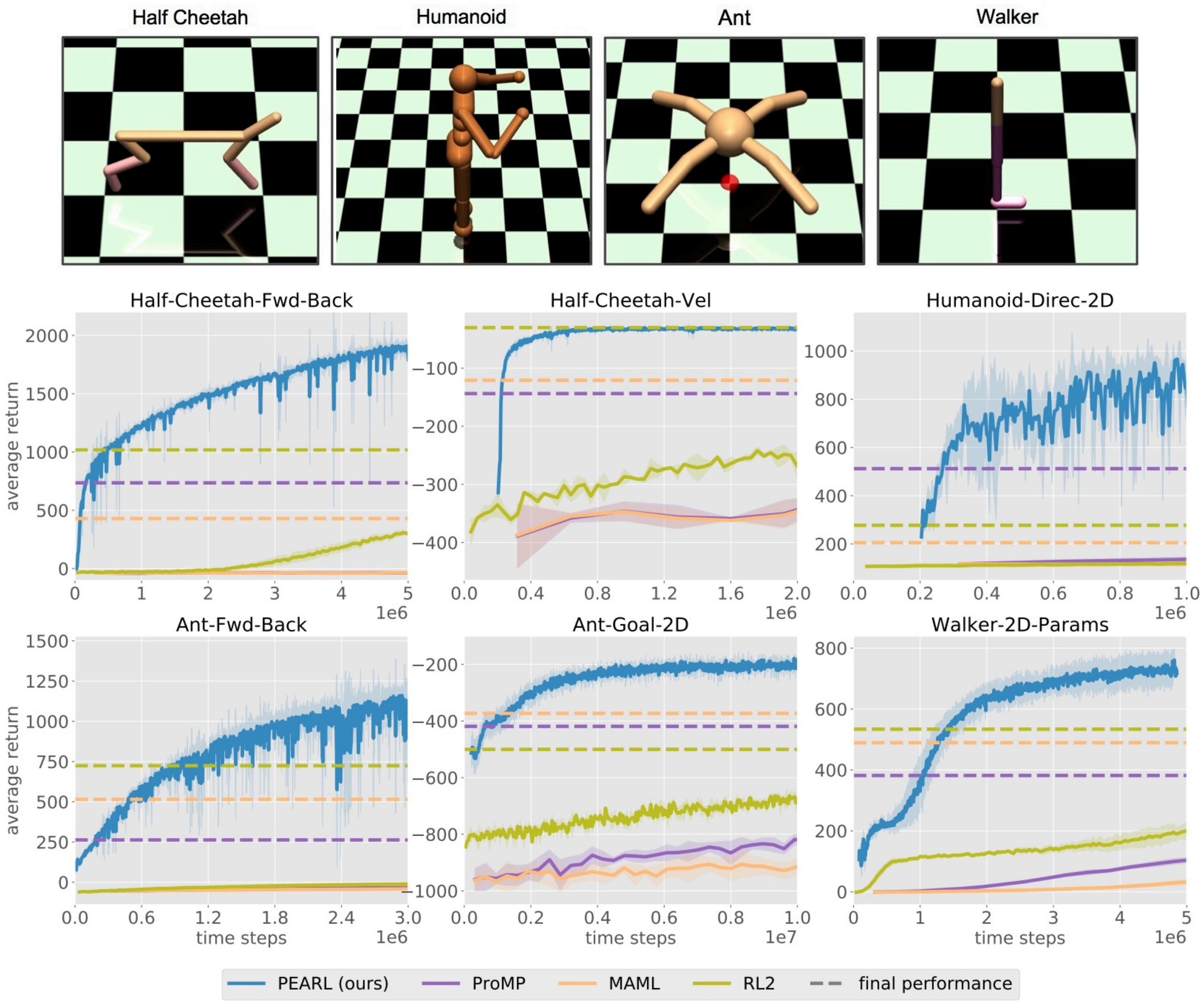


Different images correspond to different z

Different **tasks** correspond to different z
Quick search happens in z space

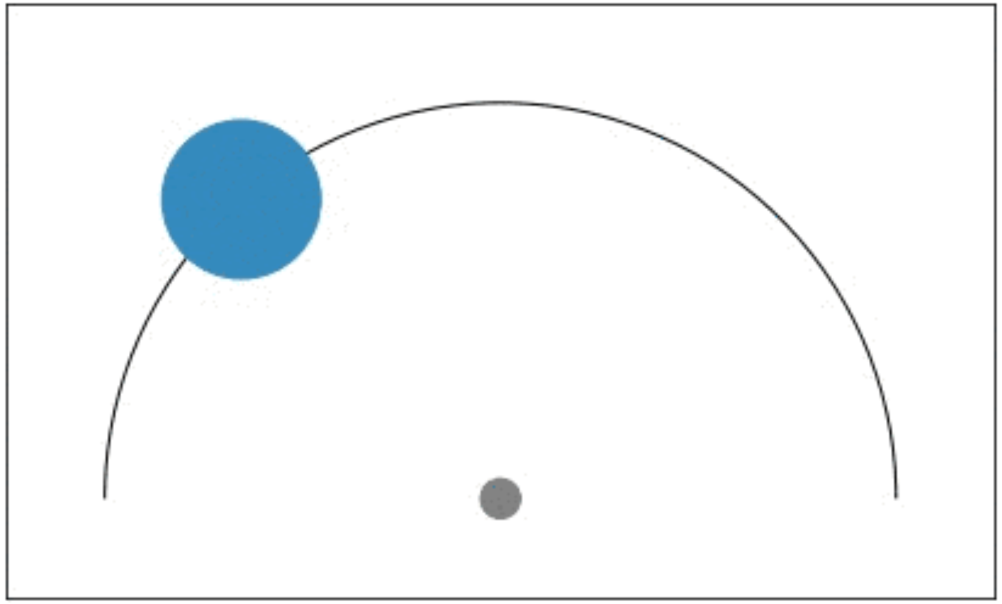
How well does it work?

Quantitative:



Gains mainly from off-policy RL

Exploration:



PEARL: Off-Policy Meta RL via Probabilistic Context Variables, Rakelly et al 2019

How well does it work?

Rapid Motor Adaptation: Kumar et al 2021

How well does it work?

Pros:

Easy to run with off-policy RL

Can be very efficient, trained offline, etc

Might be easy to incorporate priors into inference network

Cons:

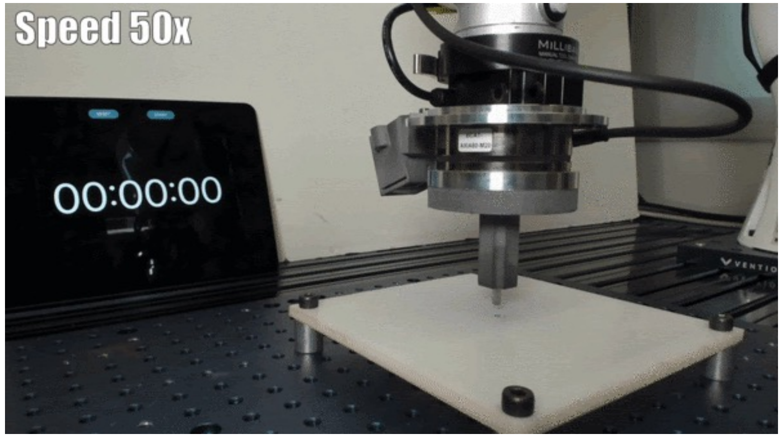
Exploration may be suboptimal

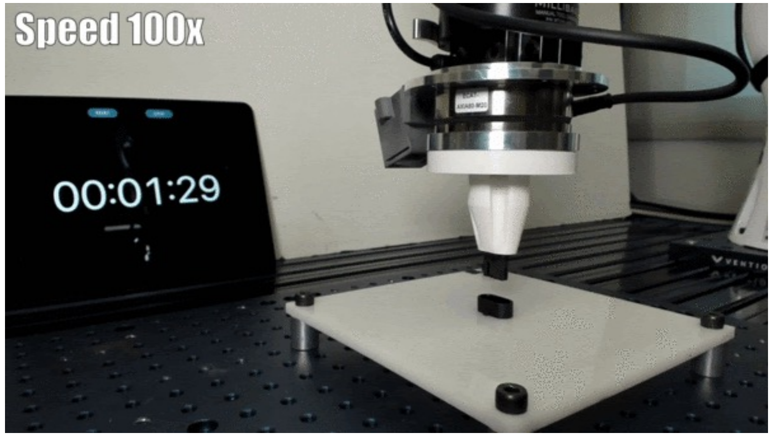
May need a huge context variable, hard to optimize/generalize

So meta-RL is cool, does it actually work?

Industrial insertion → adapting to different plug shapes

	US-AC-plug	NEMA14-30P	Metal-peg-rec	Metal-peg-rd	UK-AC-plug	Car-plug-4p	Metal-peg-sq	Car-plug-3p	EU-AC-plug
Ours	**100/100**	**100/100**	**100/100**	**100/100**	**100/100**	**100/100**	**99/100**	**75/100**	99/100
AWAC	87/100	93/100	96/100	99/100	**100/100**	**100/100**	90/100	64/100	**100/100**

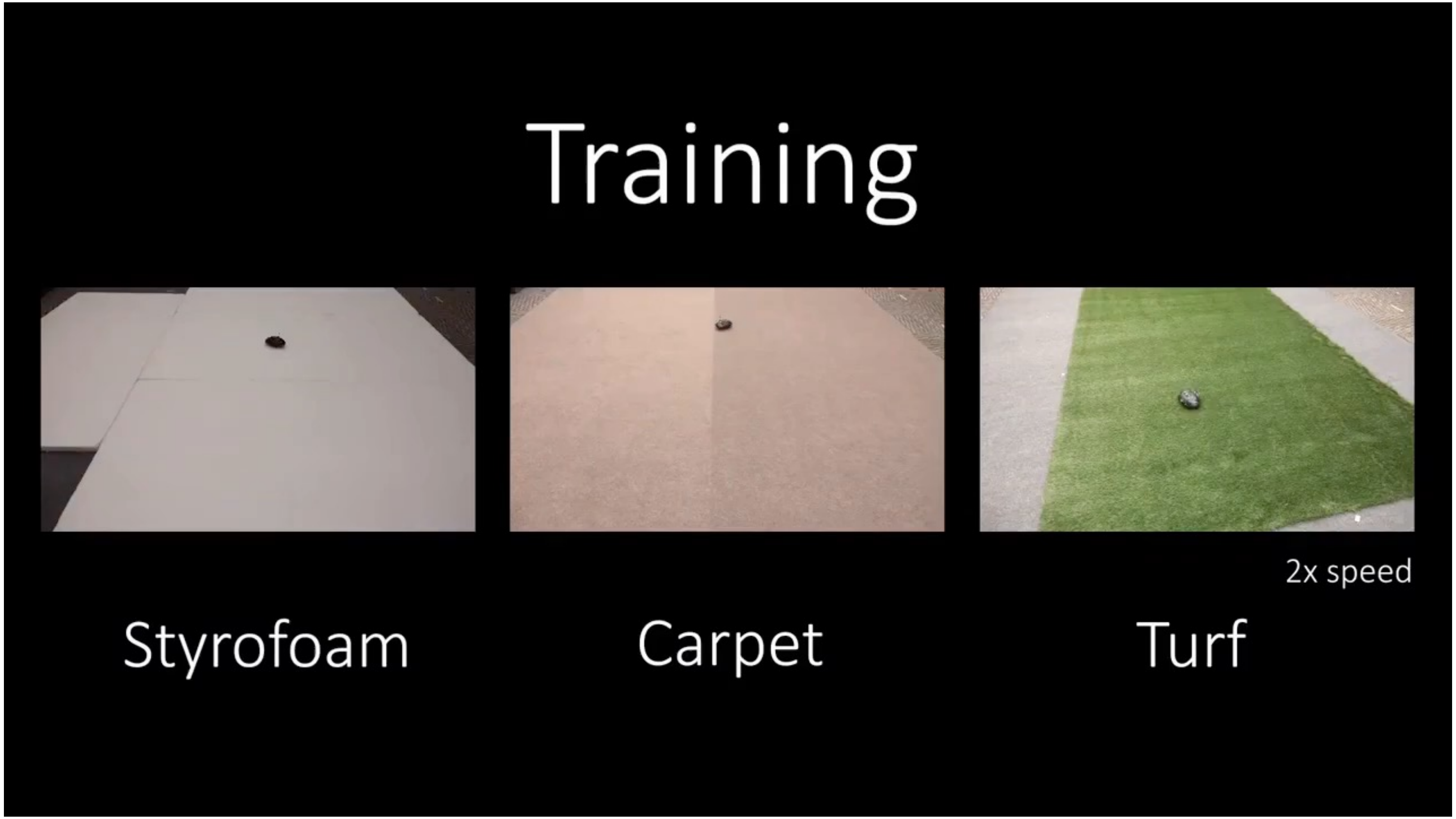




Offline Meta-Reinforcement Learning for Industrial Insertion, Zhao 2021

So meta-RL is cool, does it actually work?

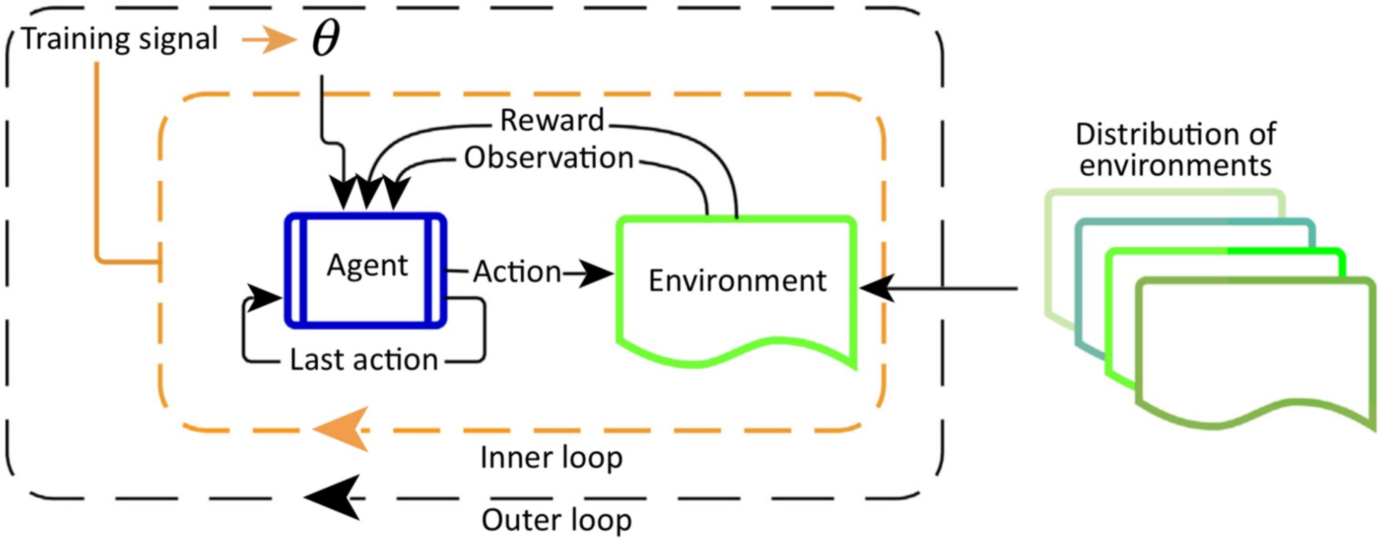
Adapting to different terrains/robot conditions



Learning to Adapt in Dynamic, Real-World Environments, Nagabandi. 2018

Takeaways from meta-RL

- Meta-RL takes multi-task RL from 0-shot to few-shot
- Meta-RL algorithms can be viewed as choices on top of bi-level optimization – memory based, gradient based, latent variable
- Meta-RL can allow adaptation when context is unknown or hard to describe



Putting things in perspective

- Multi-task (and meta) RL takes RL from specialists to generalists (well, kind of)
- The landscape can be understood along 2 axes

Amount of test-time experience

0-shot ⟷ few-shot ⟷ many-shot

Multi-task | Meta-Learning | RL from scratch

Inference Scheme

Gradient Based ⟷ Memory-based ⟷ LVM

Some heavily biased readings

Multi-Task RL

1. Gradient conflict: Gradient Surgery for Multi-Task Learning (Yu et al 2020), Multi-Task Learning as Multi-Objective Optimization (Sener et al 2019)
2. Divide and Conquer: Distral: Robust Multitask Reinforcement Learning (Teh et al 2017), Divide-and-Conquer Reinforcement Learning (Ghosh et al 2018)
3. Multi-task RL at scale: MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale (Kalashnikov et al 2021), BC-Z: (Jang et al 2022), Do As I Can, Not As I Say: Grounding Language in Robotic Affordances (Ahn et al 2022)

Meta-RL

4. Meta-RL overview, older papers by Schimdhuber/Hochreiter
5. Recurrent meta-RL: RL^2 (Duan et al), L2RL (Wang et al), SNAIL (Mishra et al), CNP (Garnelo et al 2018)
6. Gradient-based meta-RL: MAML (Finn et al), REPTILE (Nichols et al), ProMP (Clavera et al), Antoniu 2018, Bechtle 2019
7. Latent variable meta-RL: PEARL (rakelly et al), VariBAD (zintgraf et al), MAESN (Gupta et al), Zhang et al 2020
8. Model-based meta-RL: Clavera and Nagabandi 2019, Harrison and Sharma 2020, MIER (Mendonca et al)
9. Exploration in meta-RL: MAESN (Gupta et al), DREAM (Liu et al), GMPS (Mendonca et al)
10. Supervision in meta-RL: UMRL (Gupta et al), CARML (Jabri et al), UML (Hsu et al)

Lecture Outline

Recap – Max-margin and Max-ent IRL

↓

Making max entropy IRL practical

↓

IRL as a GAN

↓

Why multi-task or meta-RL?

↓

Multi-Task Reinforcement Learning

↓

Meta-Reinforcement Learning