



Reinforcement Learning

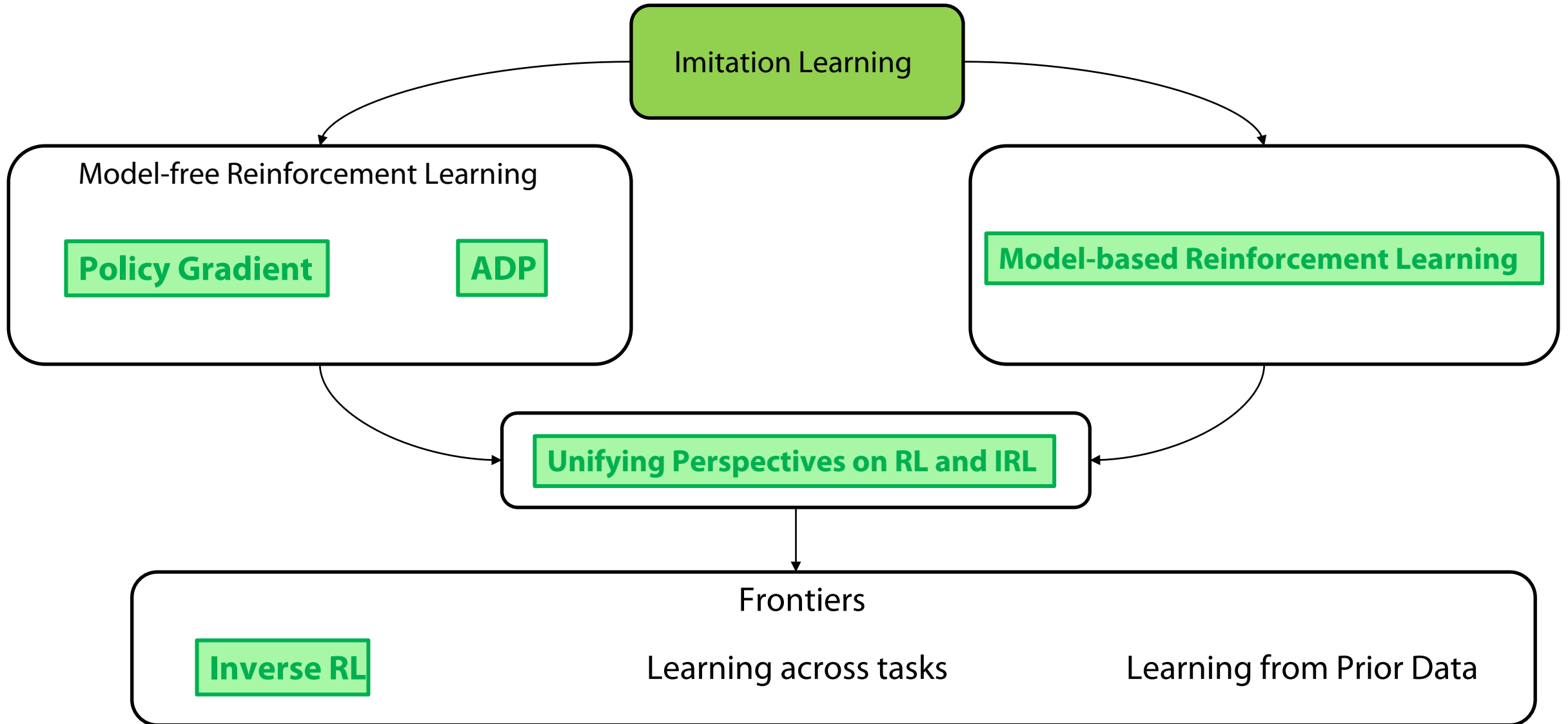
Autumn 2024

Abhishek Gupta

TA: Jacob Berg



Class Structure



Lecture Outline

Recap: Multi-task RL formalism



Multi-Task Reinforcement Learning



Meta-Reinforcement Learning



Why offline RL?



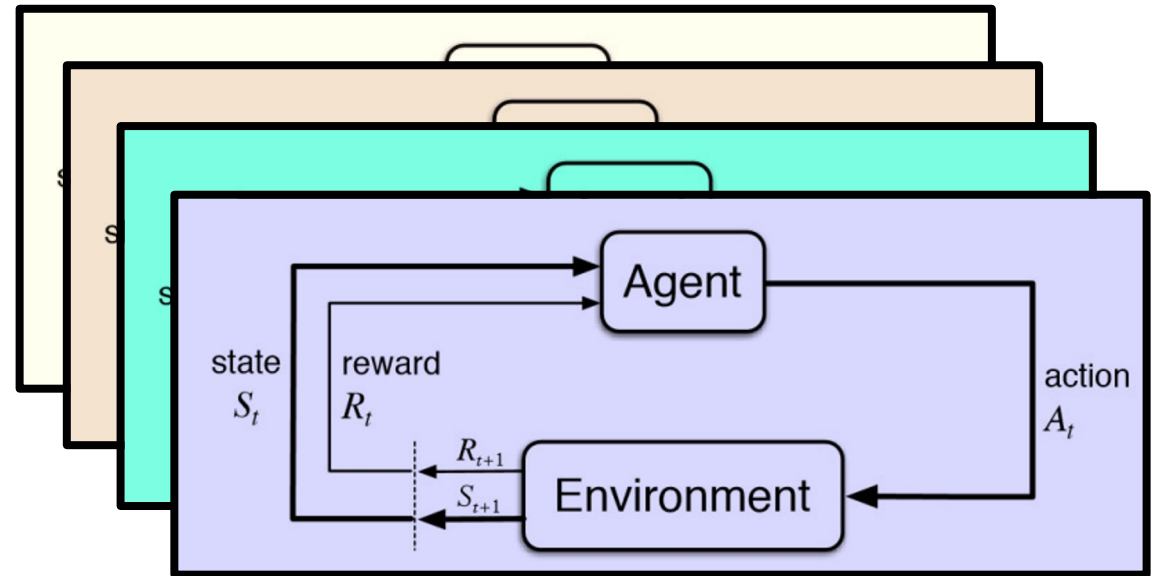
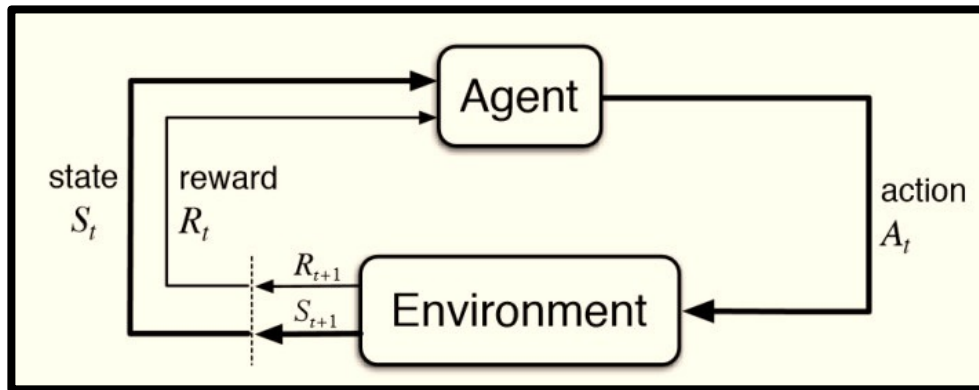
Methods for offline RL

Multi-Task RL – Distribution over MDPs

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

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

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

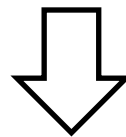


Multi-Task Meta-MDP

Let us assume the factor of variation across MDPs can be characterized by known ω_i
Eg: task ID, goal, video, language, ...

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

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



Slight reformulation

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

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

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

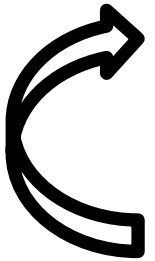
$$\mu \rightarrow \mu(s_0)p(\omega_i)$$

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

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 $(s, a, s', \text{task ID})$ pairs
 3. Perform actor and critic updates on the shared actor and critic with losses summed up across tasks

Critic

Actor

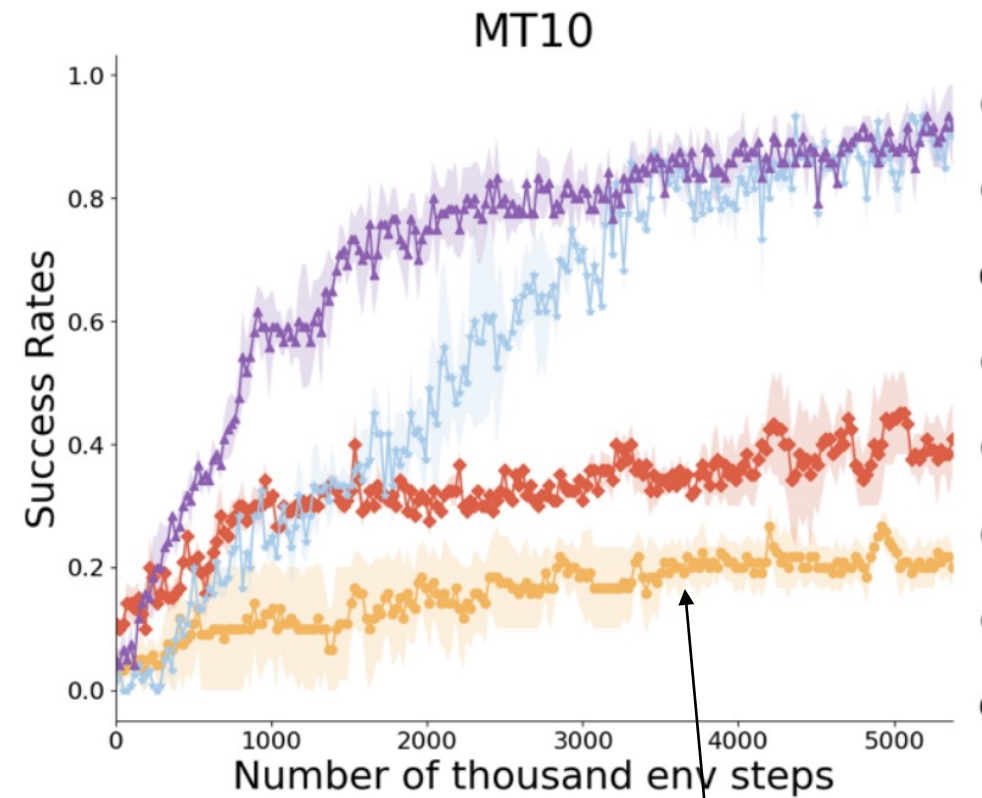
$$\pi \leftarrow \arg \max \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{a \sim \pi} [Q^\pi(s, a, \tau)]$$

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

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

Lecture Outline

Recap: Multi-task RL formalism



Multi-Task Reinforcement Learning



Meta-Reinforcement Learning



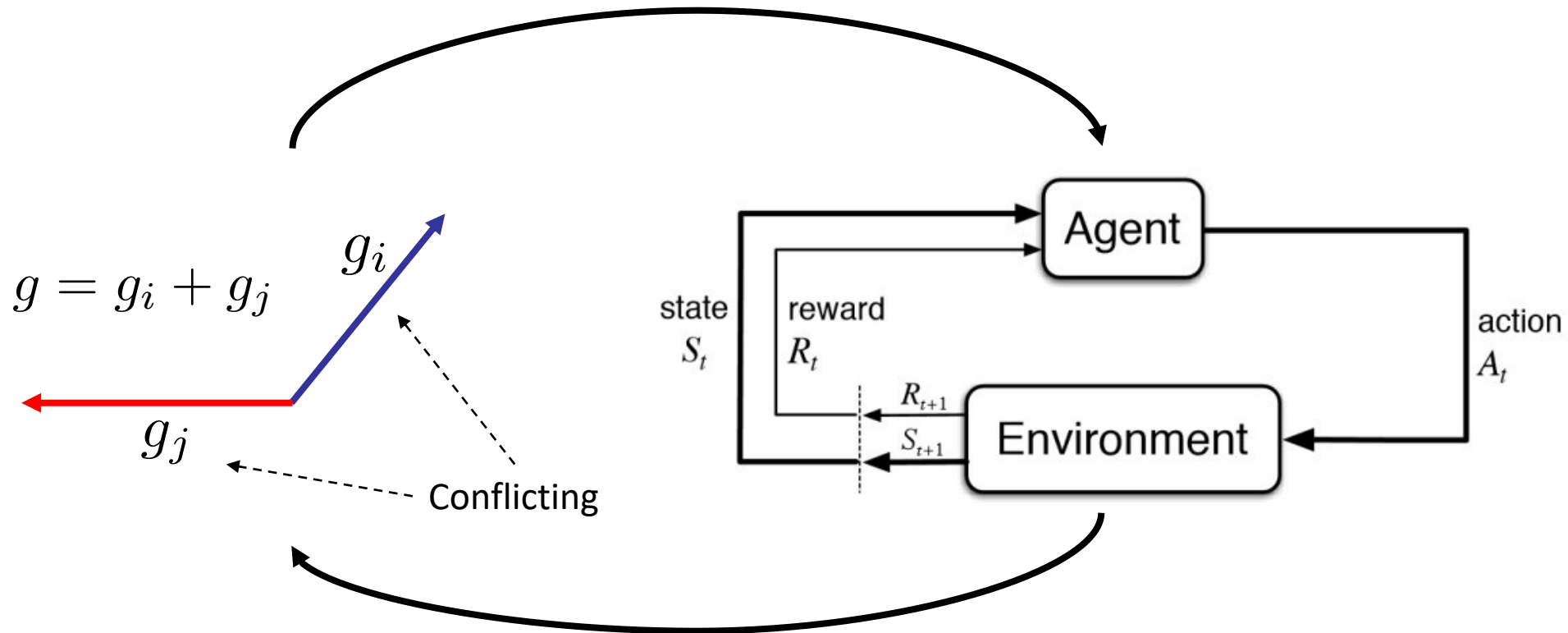
Why offline RL?



Methods for offline RL

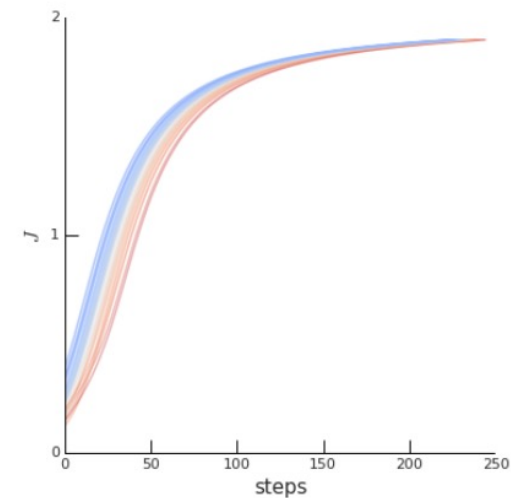
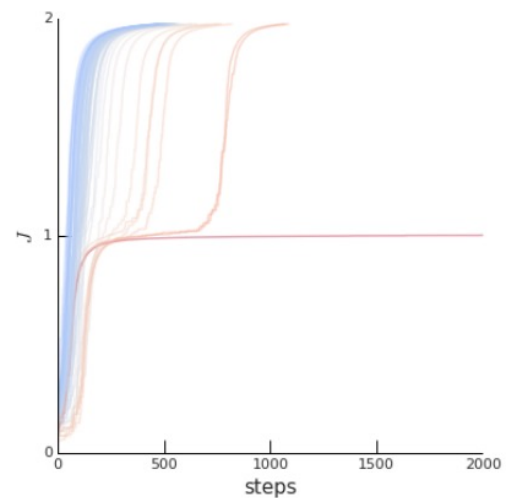
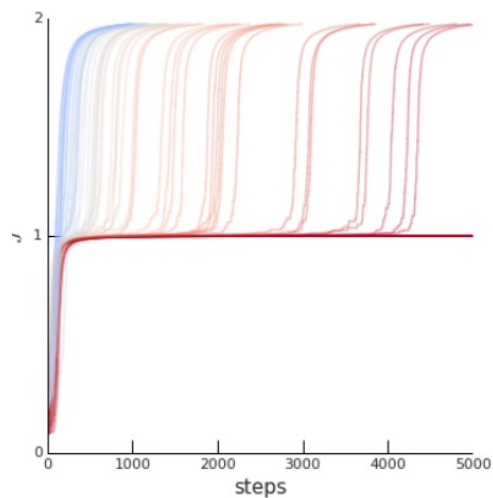
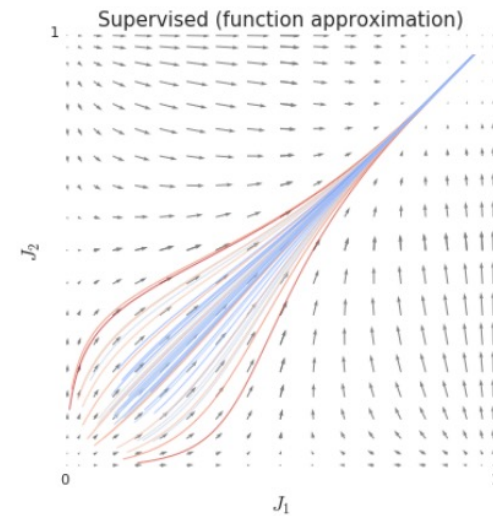
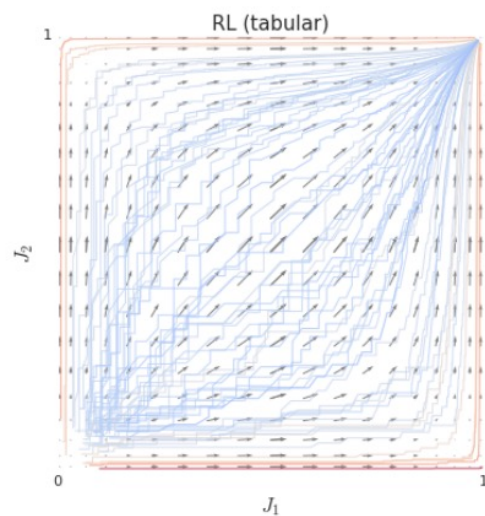
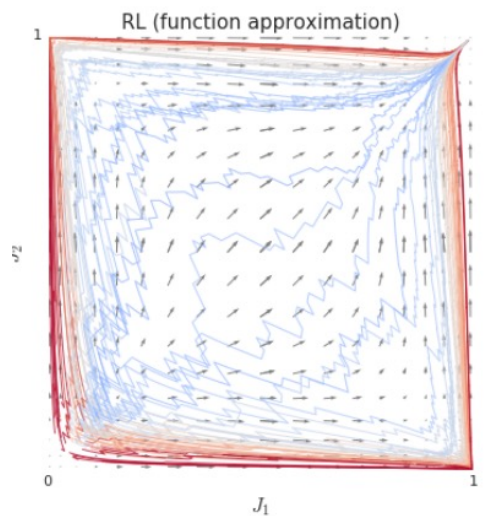
Why is it hard to do Multi-Task RL?

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



Winner-Take-All Phenomena

Vicious cycle of data collection and policy optimization

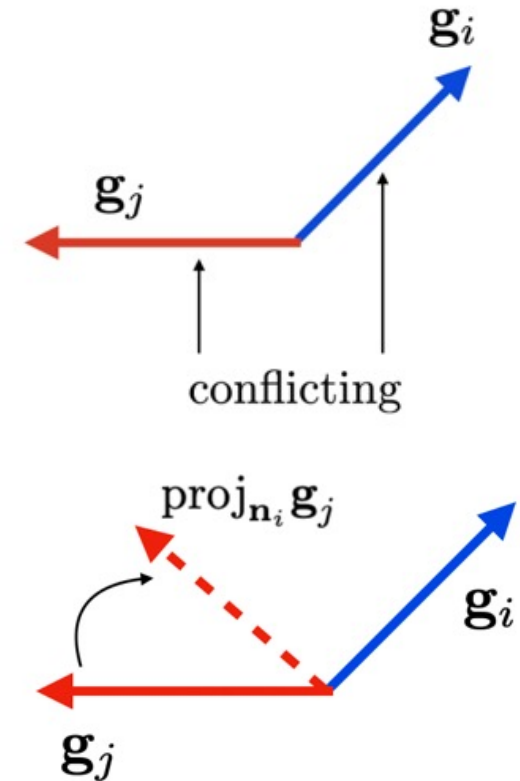
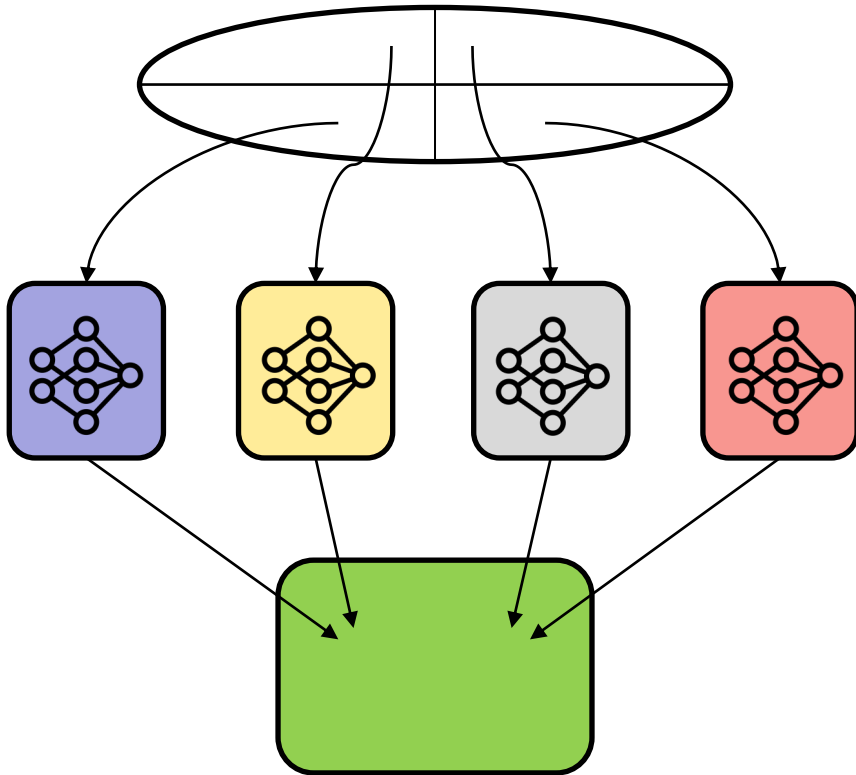


How can we deal with gradient interference in RL?

If issue is exploration + conflicting gradients is bad

Idea 1: Remove exploration from MTRL

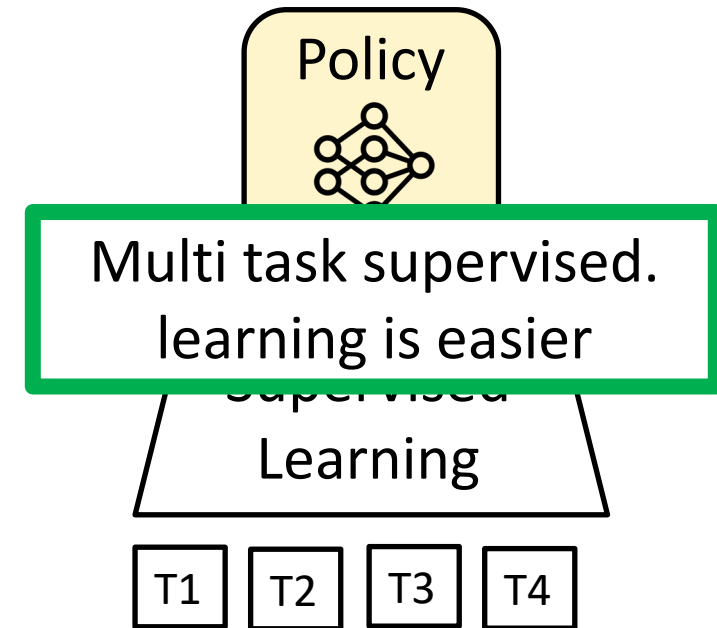
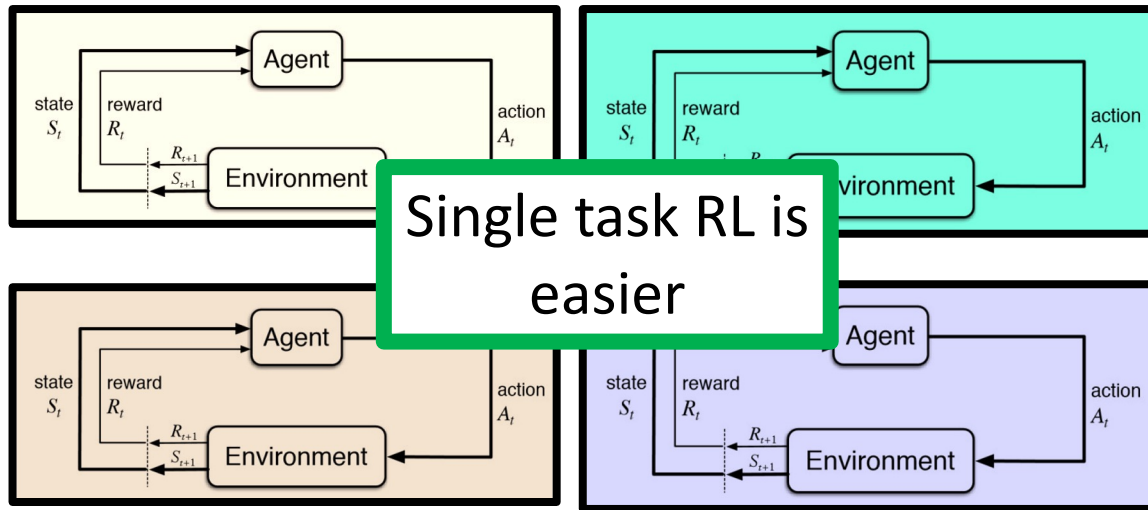
Idea 2: Modify gradients



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 \rightarrow standard RL

Distillation \rightarrow supervised learning

Divide and Conquer RL: Mathematical Formulation

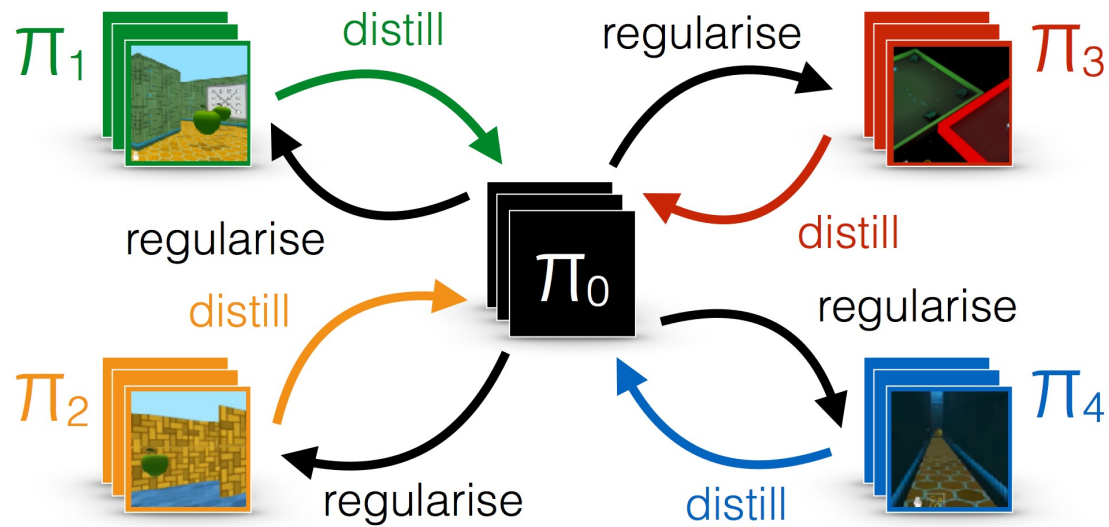
Shared policy Per-task policy

Per-task reward

Match shared/individual policy

Entropy

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



Single task RL → standard RL

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

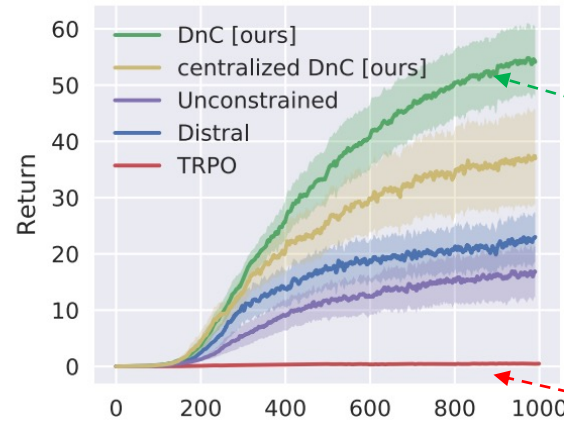
Distillation → supervised learning

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

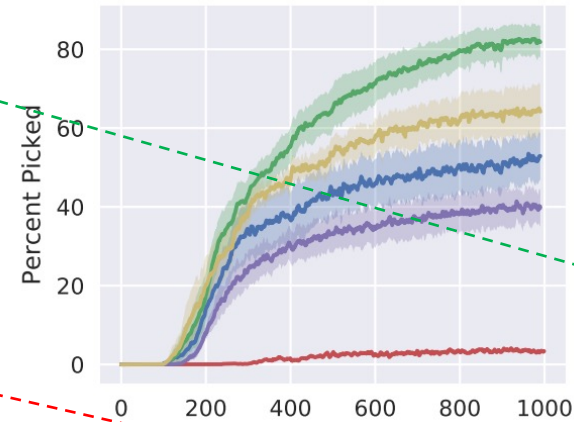
Experimental Validation



(a) *Picking task*

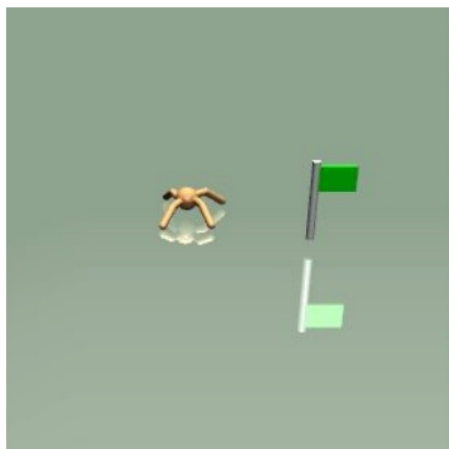


(b) Average Return on *Picking*

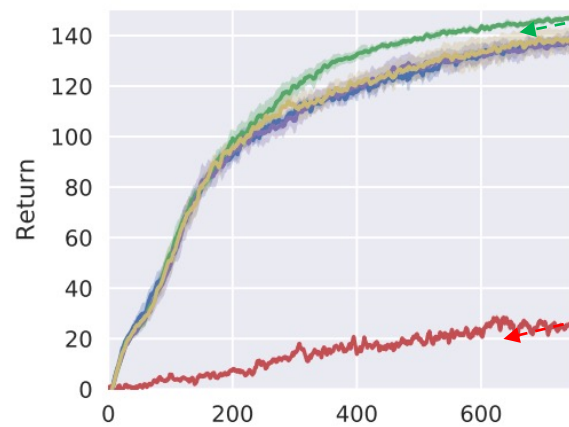


(c) Success rate on *Picking*

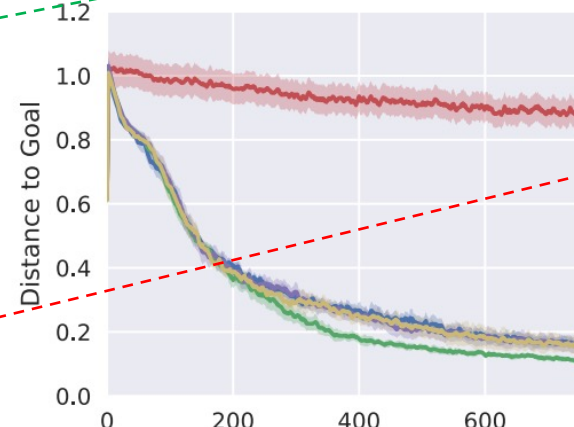
Divide and Conquer



(j) *Ant*



(k) Average Return on *Ant*



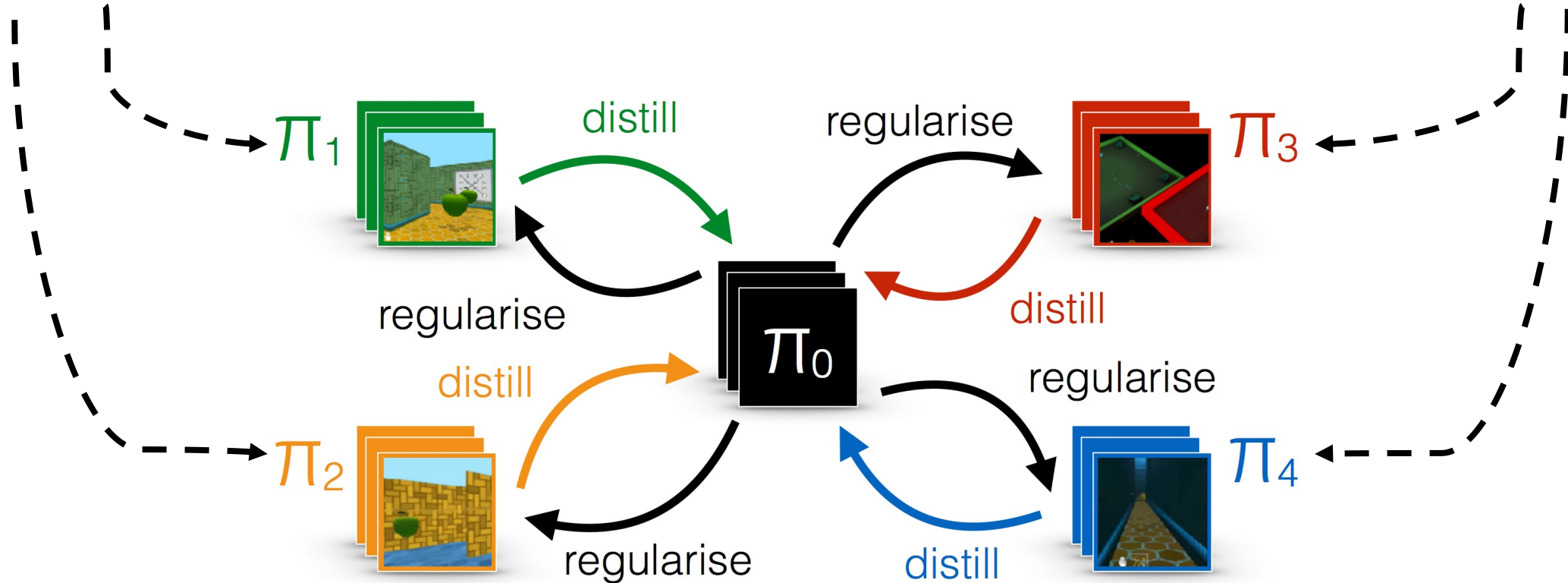
(l) Distance from goal on *Ant*

Monolithic RL

Divide and Conquer Reinforcement Learning

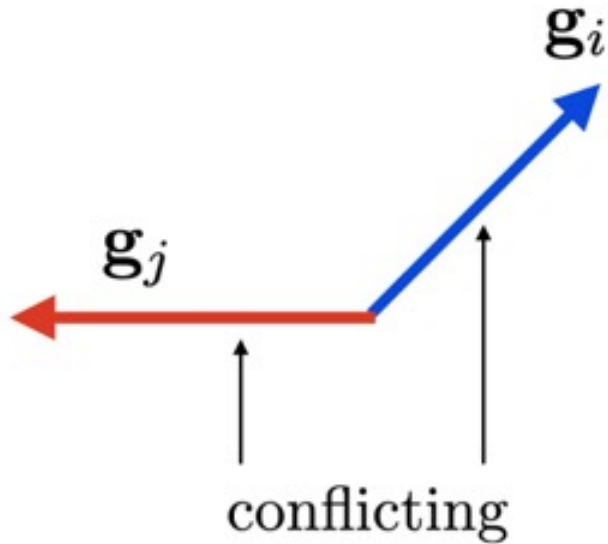
Is this enough?

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'

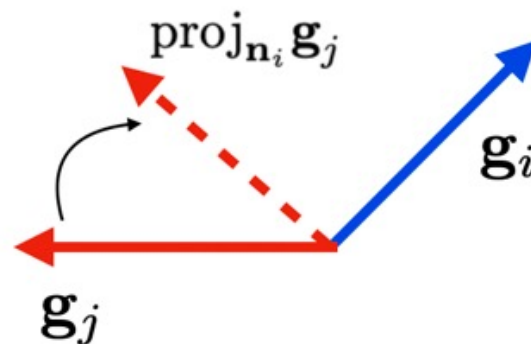
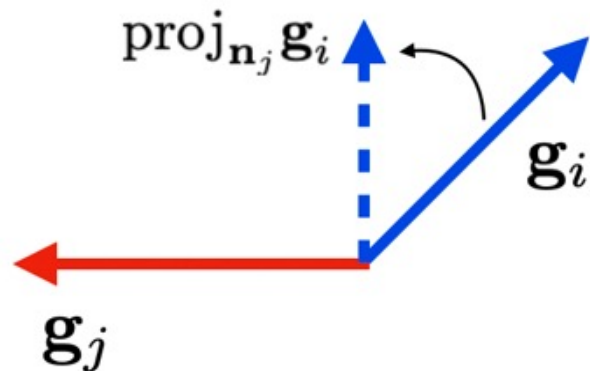
Replace g_j by g_j'

What should g_i' and g_j' be?

Idea: When gradients conflict, project them to deconflict

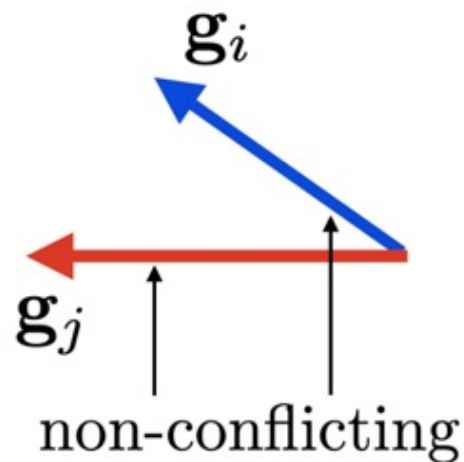
Deconflicting gradients with PCGrad

If gradients conflict: project them onto the normal plane



$$\mathbf{g}_i = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \cdot \mathbf{g}_j$$

Otherwise: leave them alone

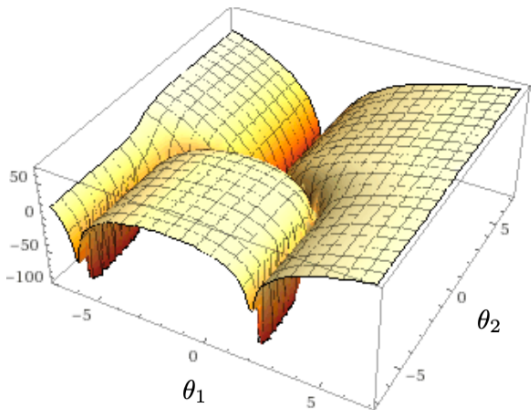


Does this empirically help?

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

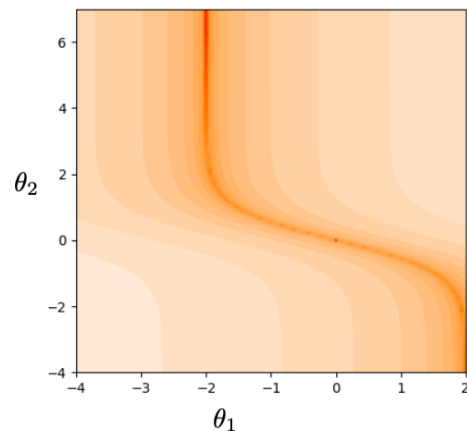
$$\mathcal{L}_2(\theta) = 25 \log(\max(|.5\theta_1 - \tanh(\theta_2) + 2|, 0.000005))$$

Multi-Task Objective



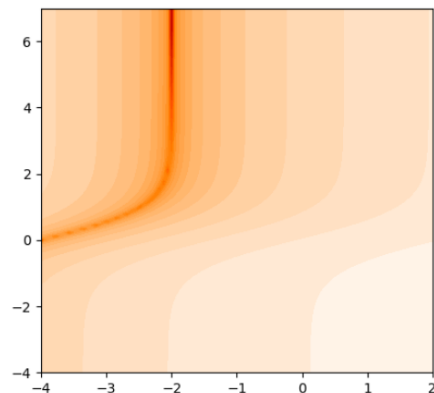
(a)

Task 1 Objective



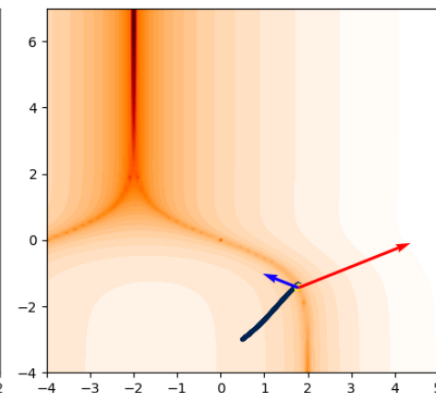
(b)

Task 2 Objective



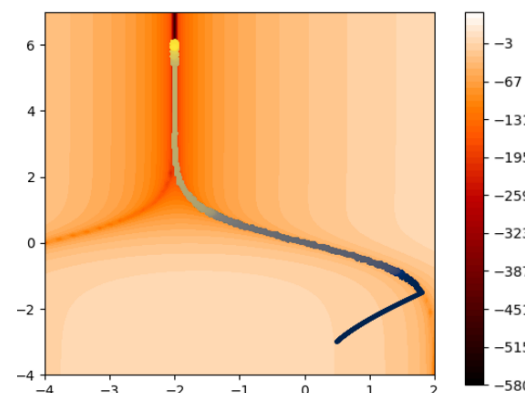
(c)

Adam



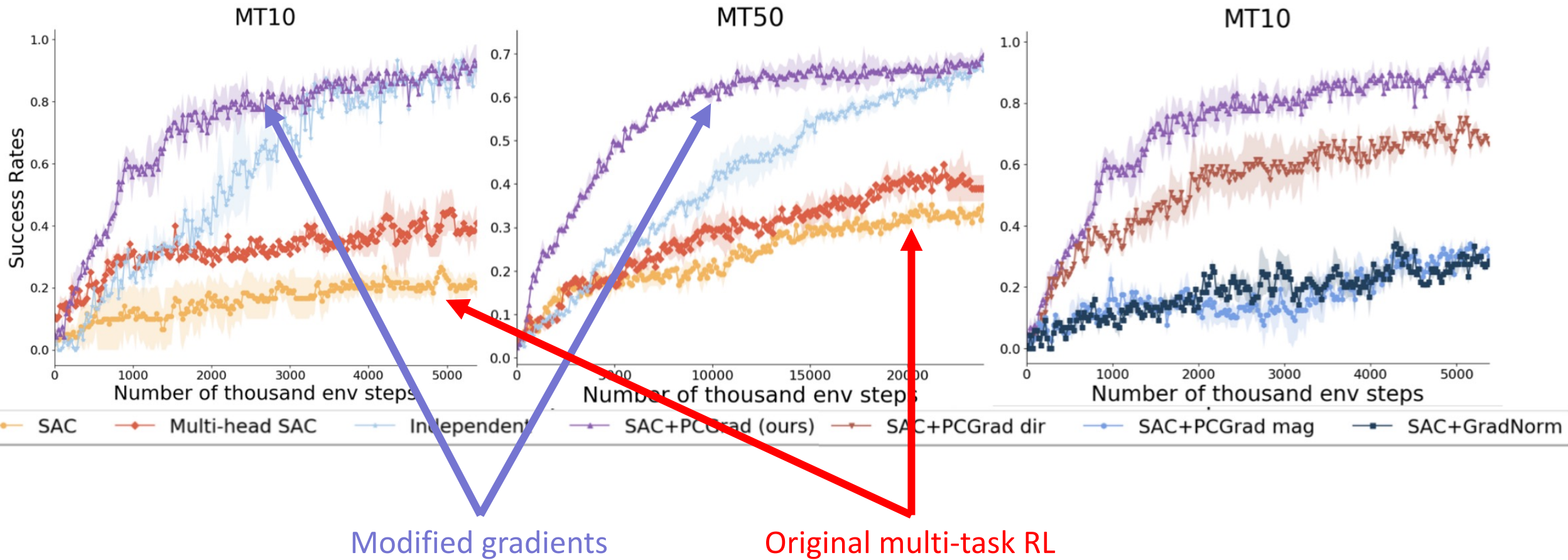
(d)

Adam + PCGrad

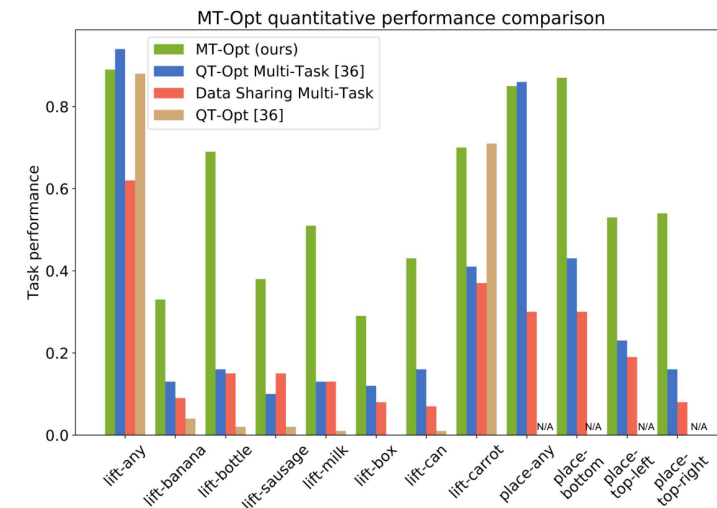
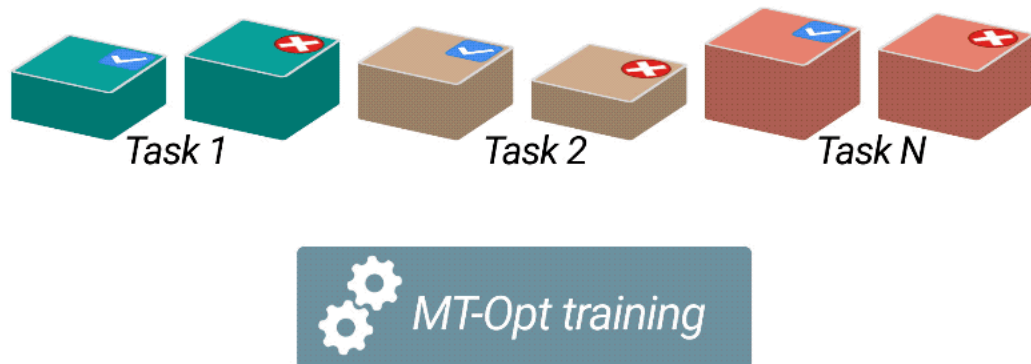
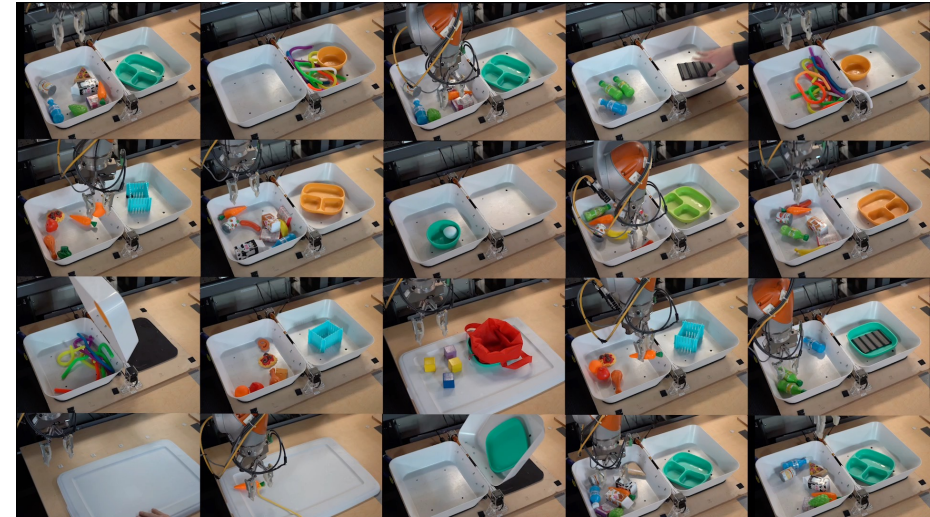
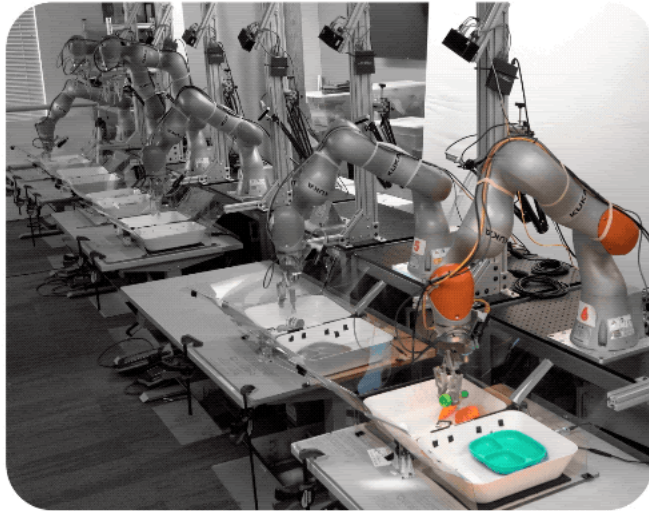


(e)

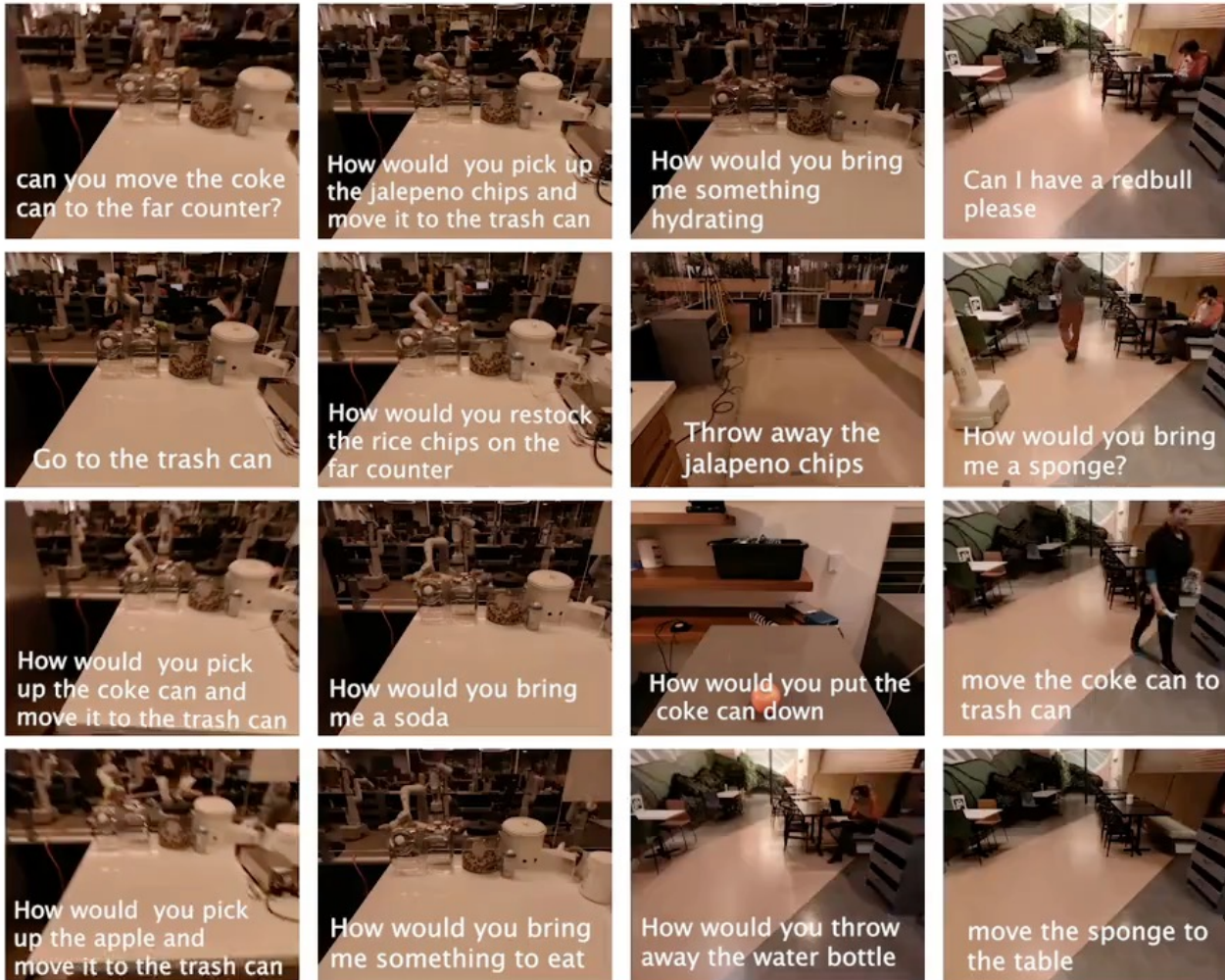
Does this empirically help?



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



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



ω_i can be language too!

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: Multi-task RL formalism



Multi-Task Reinforcement Learning



Meta-Reinforcement Learning



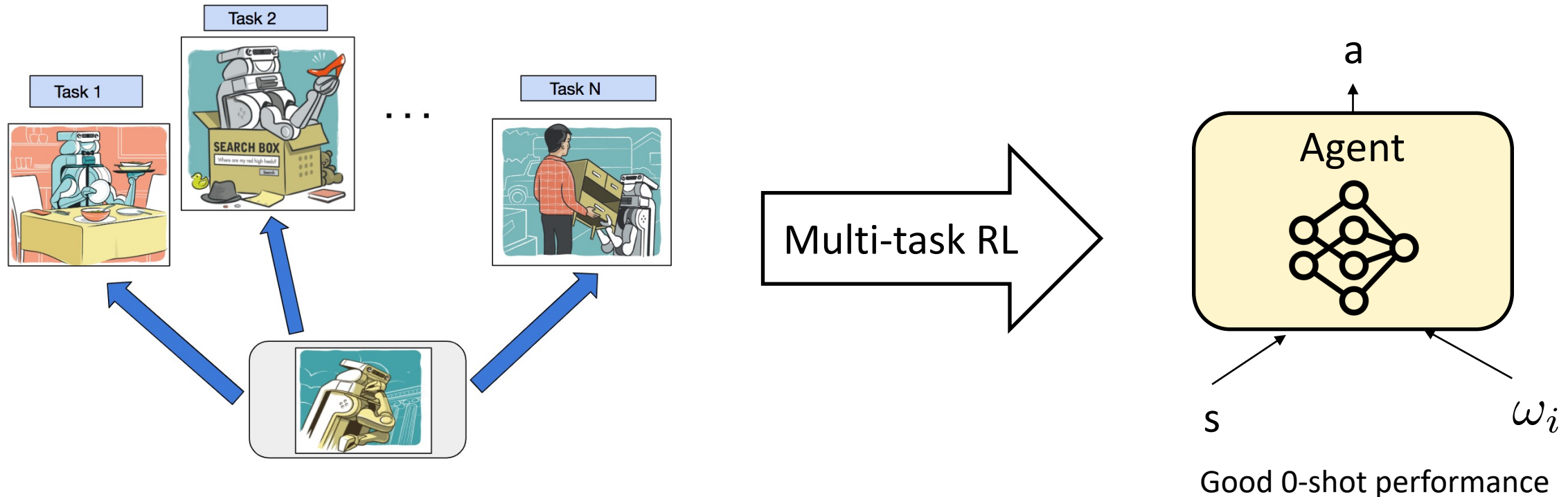
Why offline RL?



Methods for offline RL

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, ...

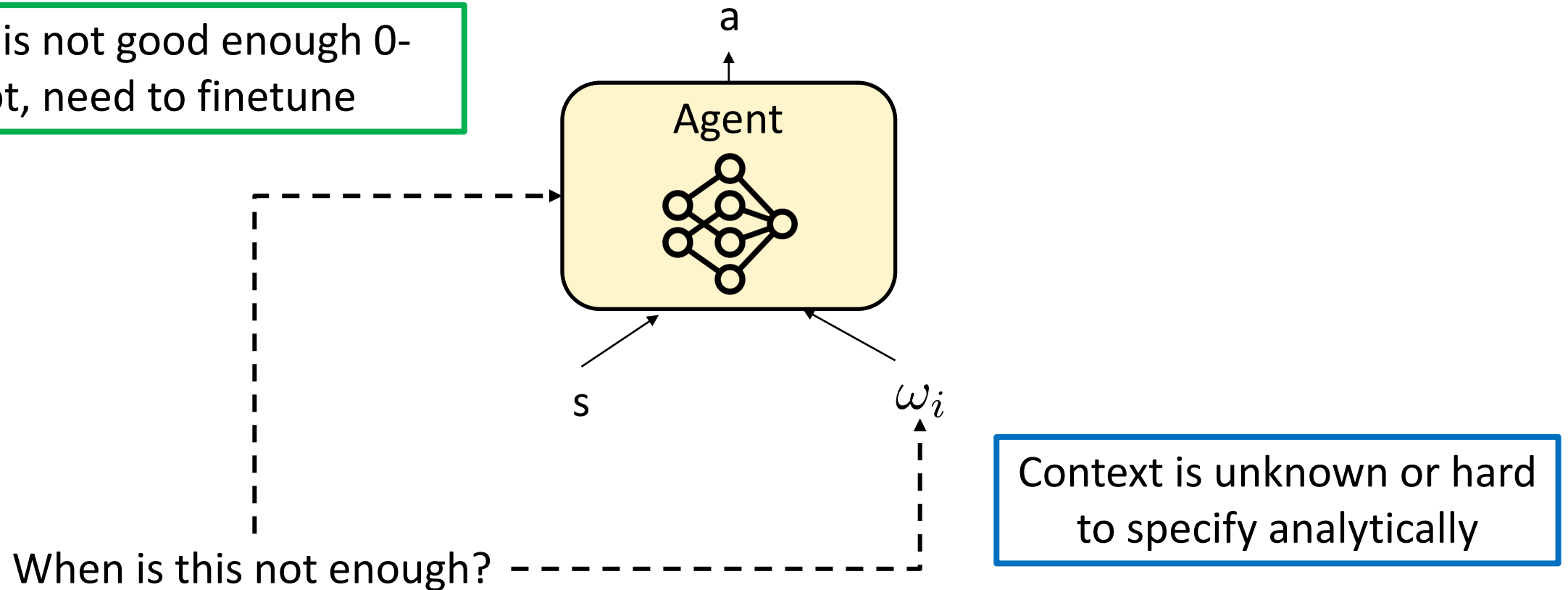


When is this not enough?

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

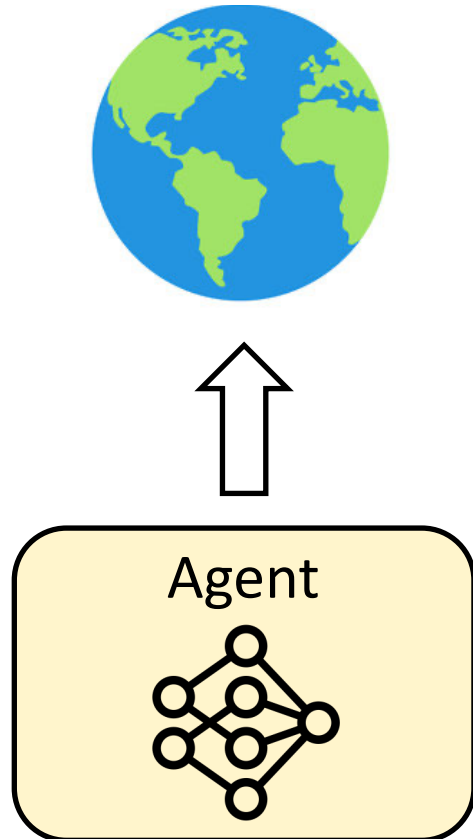


Context is unknown or hard to specify analytically

When is this not enough?

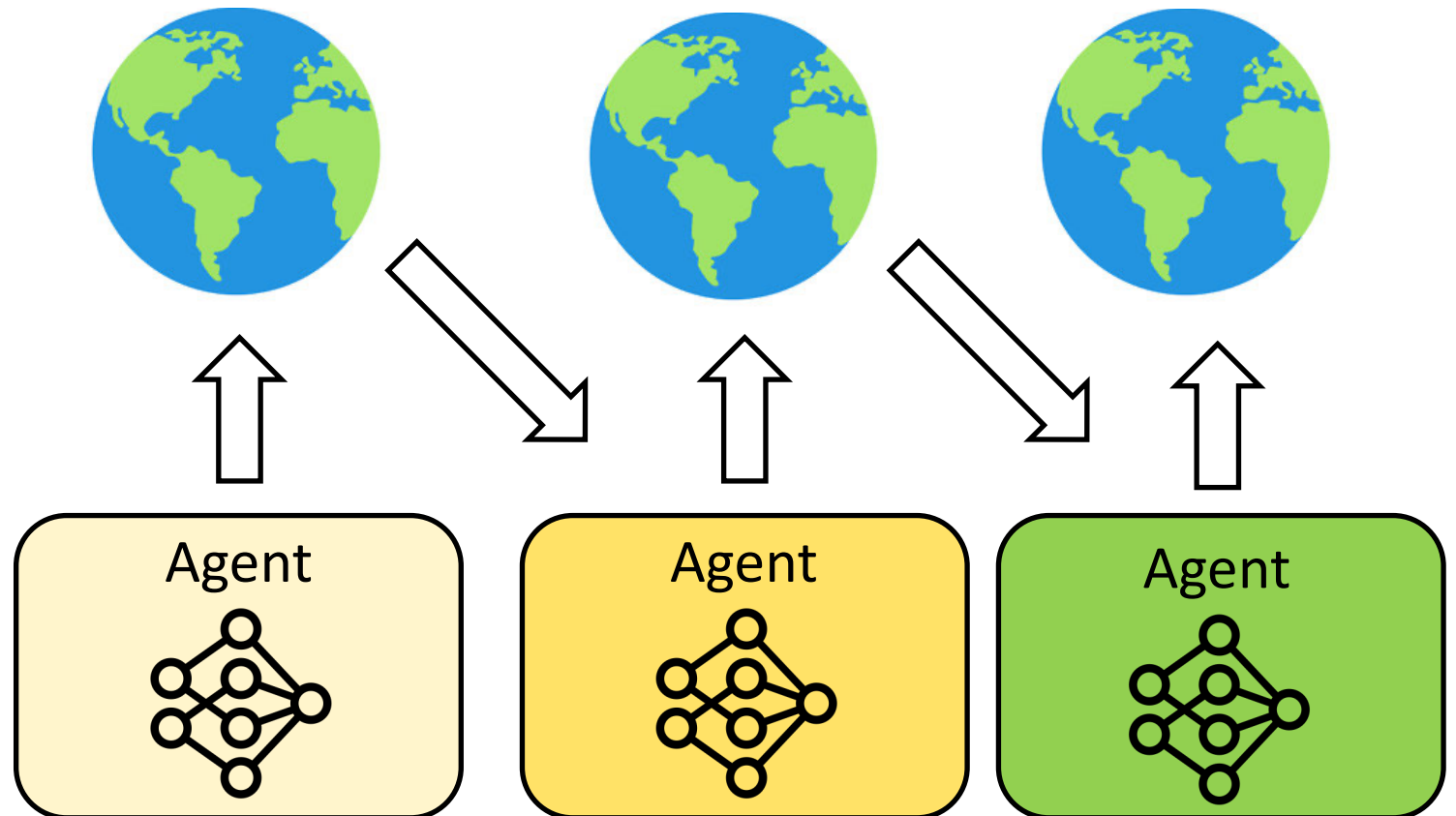
From 0-shot learning to few-shot learning

0-shot MTRL: No experience at test time

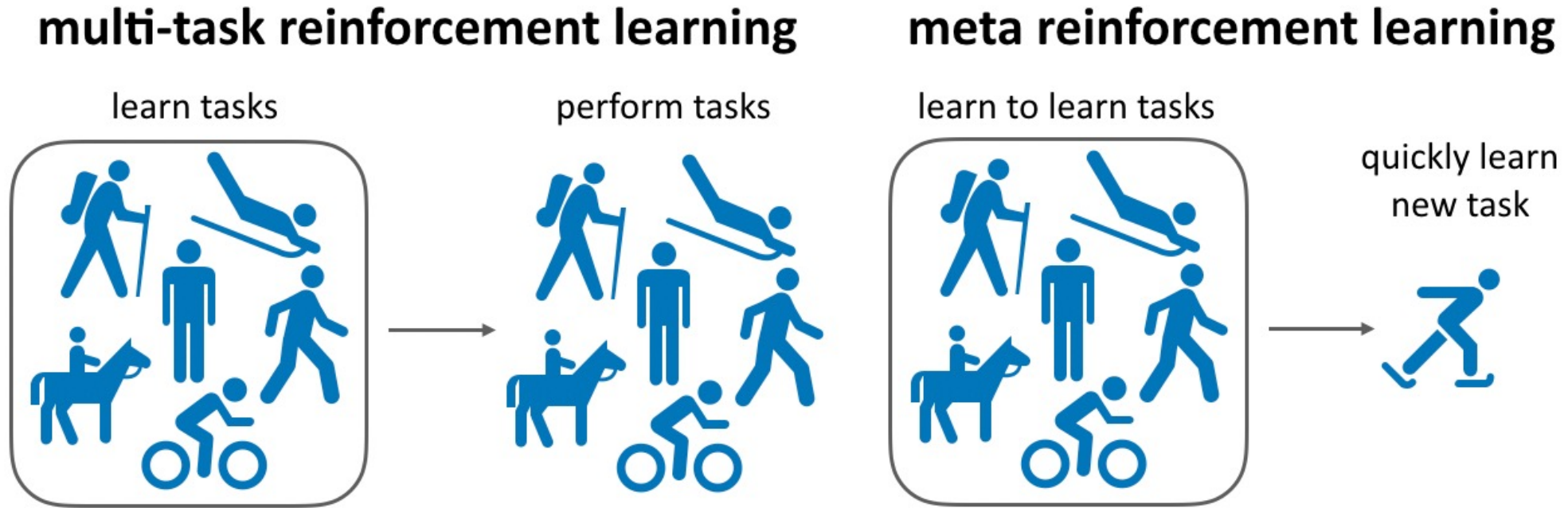


Meta-RL: Small amount of experience at test time

Fast adaptation with experience

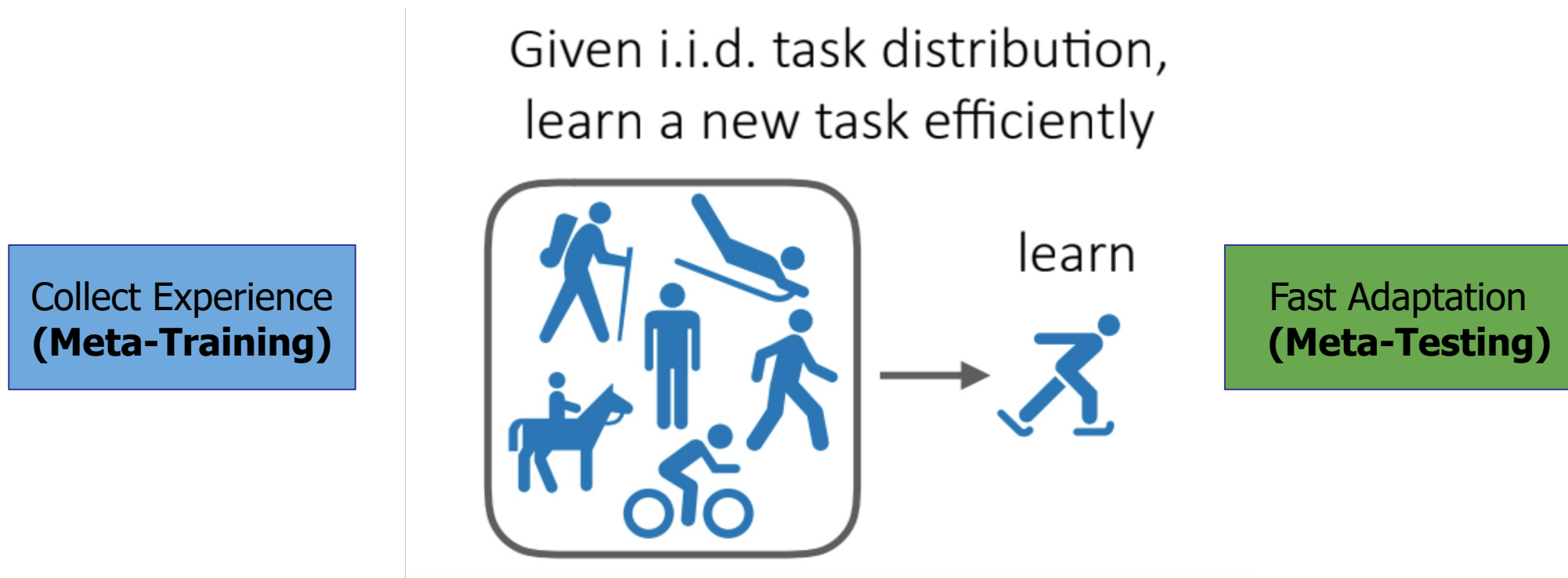


Connection to Contextual Multi-Task RL



- 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

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

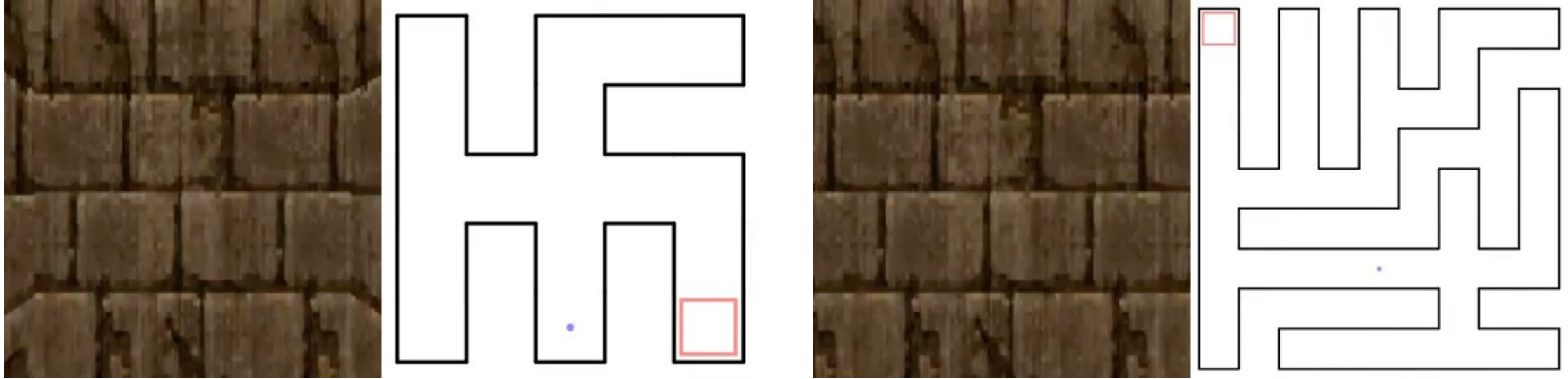
Per-task updated policy

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

General Structure of Meta-RL Algorithms

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

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

- 
1. Sample a batch of tasks from $p(\tau)$
 2. collect data pre-update
 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}

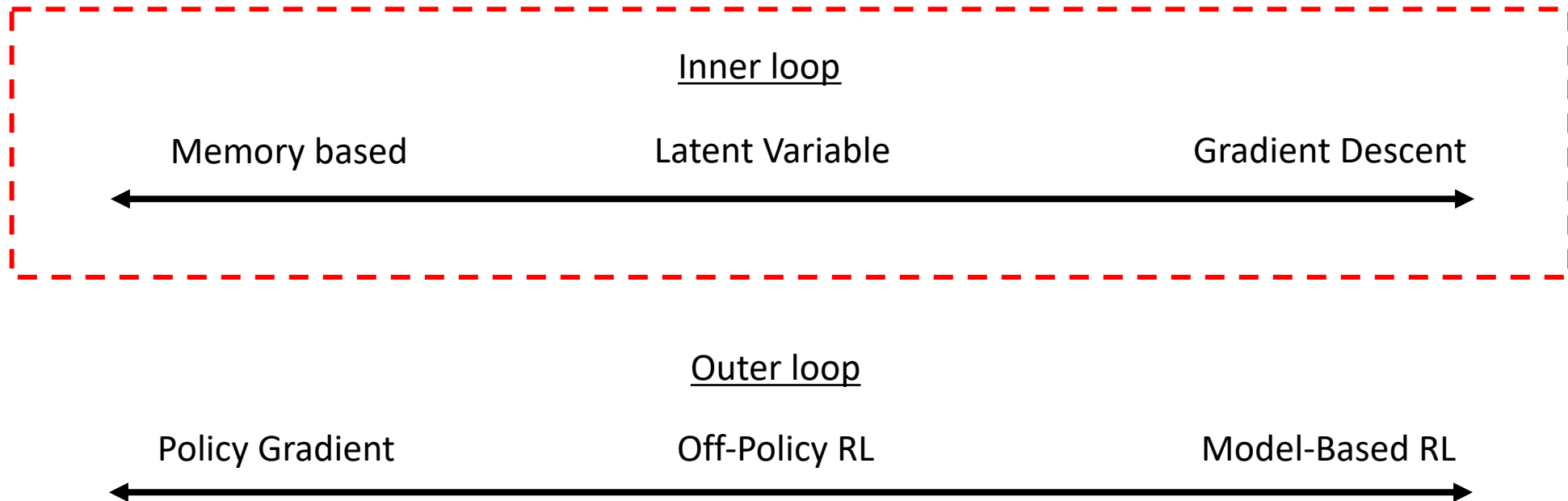
Solution Techniques for Meta-RL Problems

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

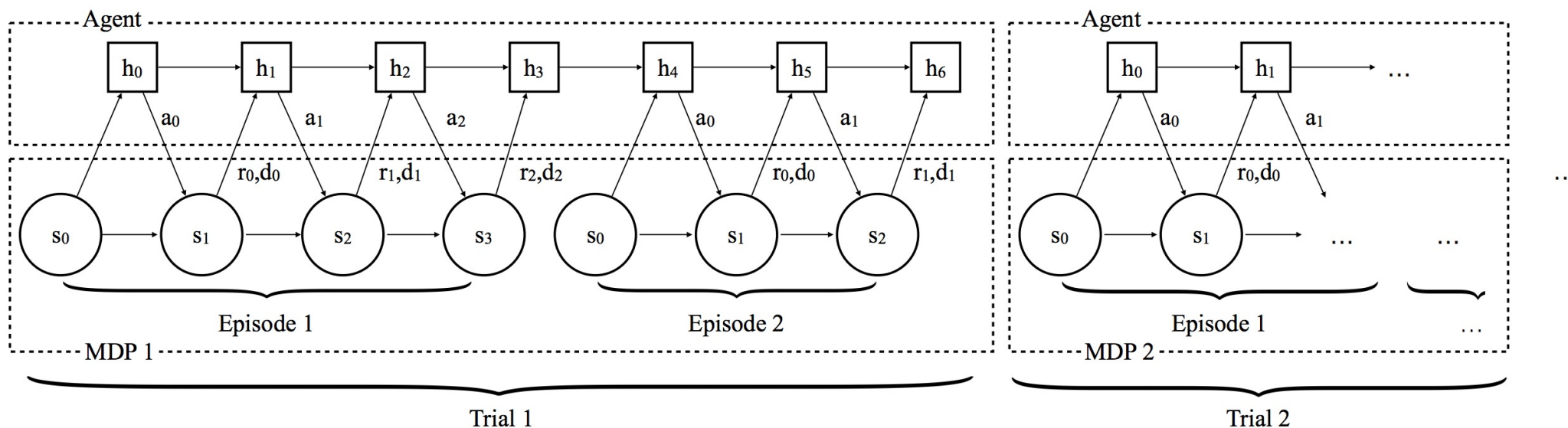
- Parameterization of f_{θ} for inner loop
- Algorithm for outer loop optimization



Memory Based Meta-RL

Idea: Make the update function forward pass of an RNN

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




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

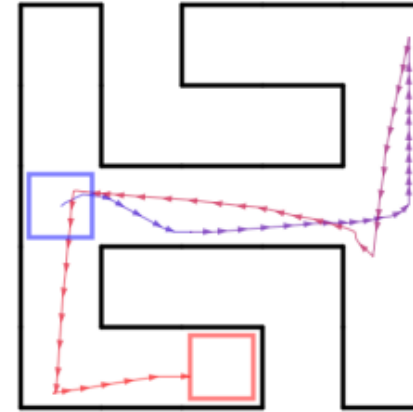
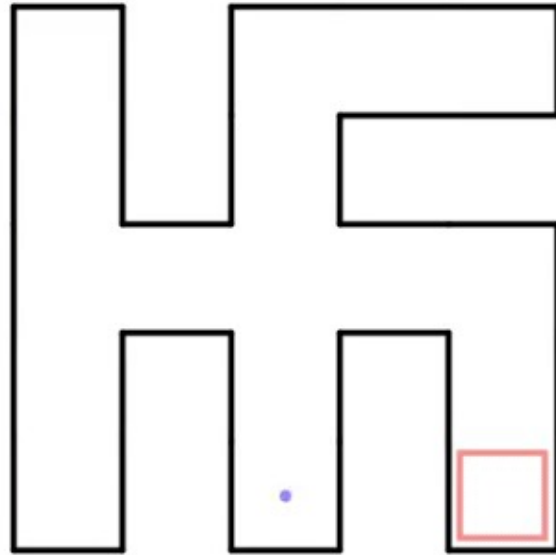
Meta-Training

- 
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

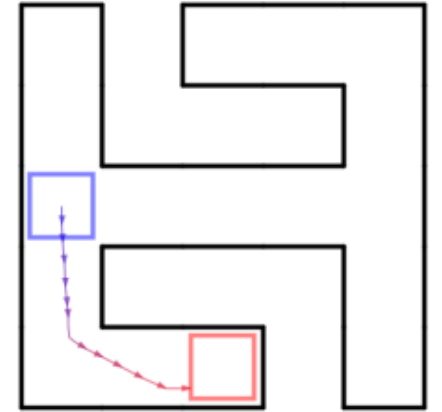
Meta-Testing

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

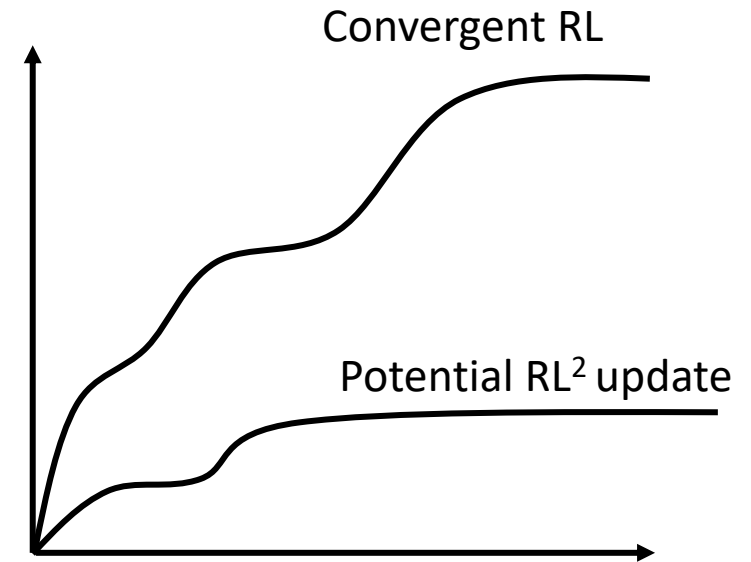
Arbitrarily flexible inner loop

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?

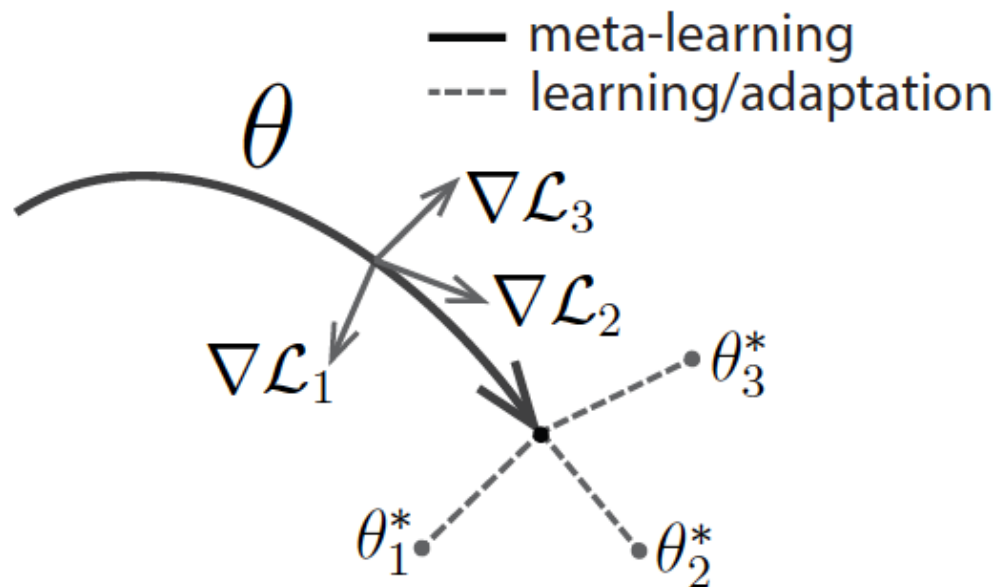
Force $f(\theta)$ to be a convergent optimization algorithm like SGD

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

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

↑ 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 pre-update 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

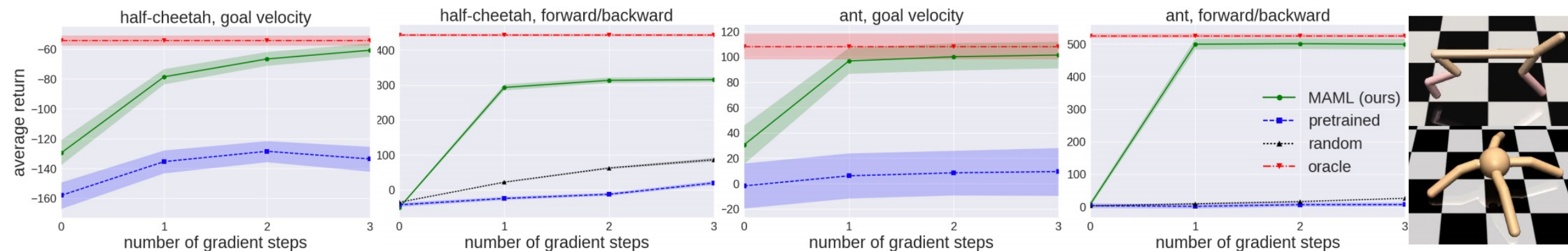
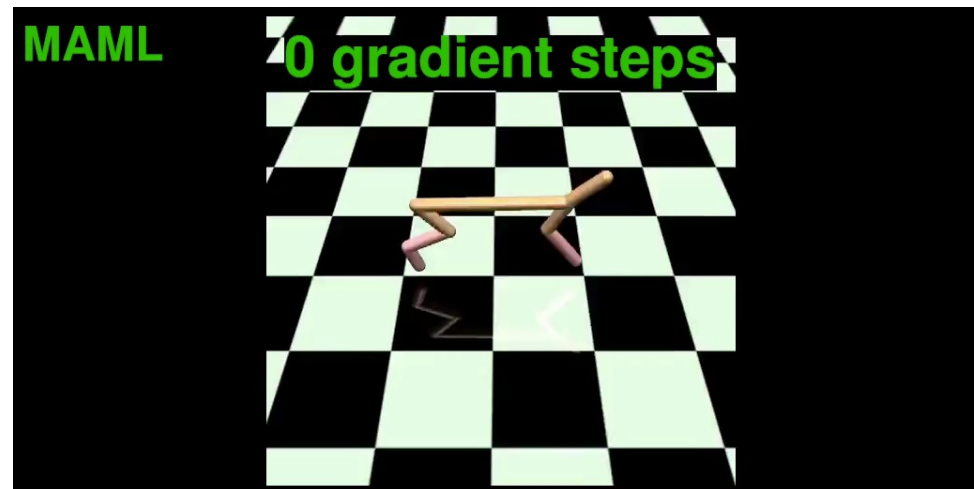
How well does it work?

Tasks:

Half cheetah: goal velocity,

Half cheetah: forward/backward

Ant: forward/backward



How well does it work?

Pros:

Consistent, worst case performance is PG

Only need to learn initialization

Cons:

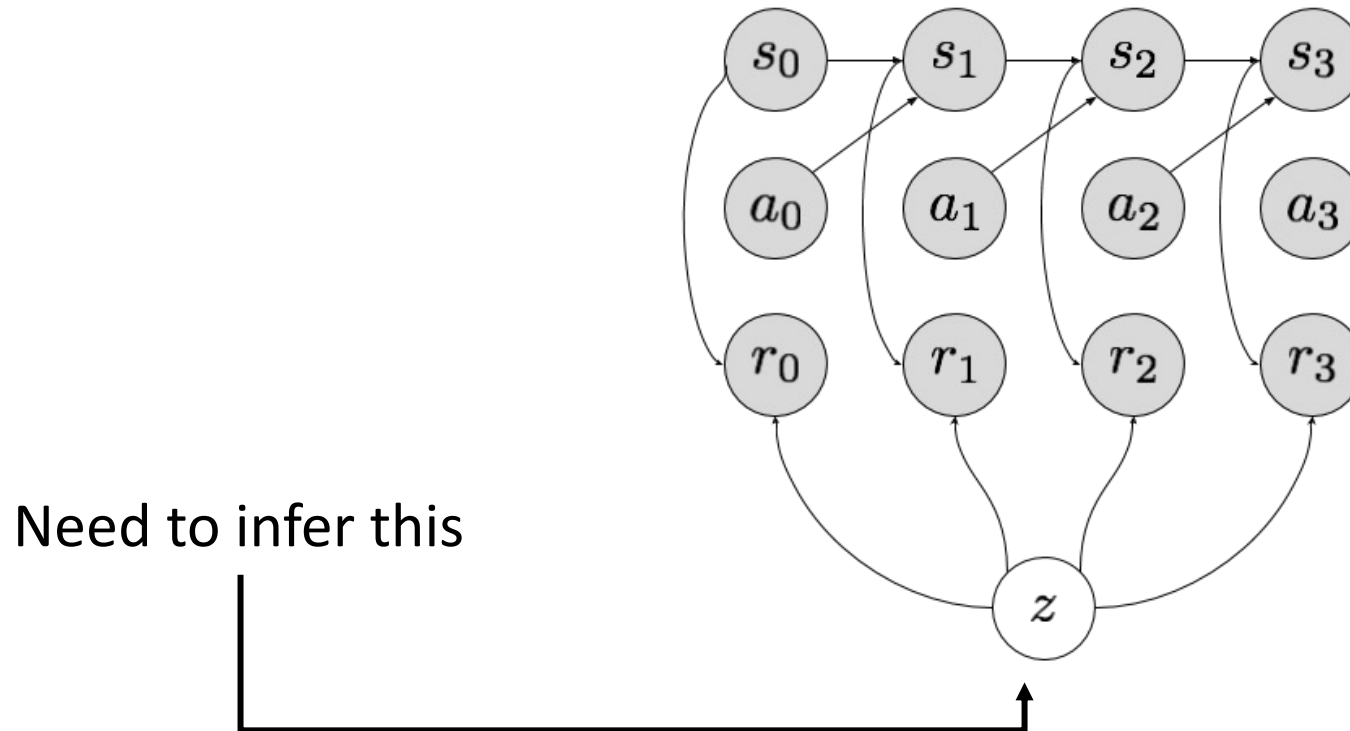
Second order gradients needed

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



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


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

Deploy latent conditioned
policy


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

Recasting meta-RL as context inference

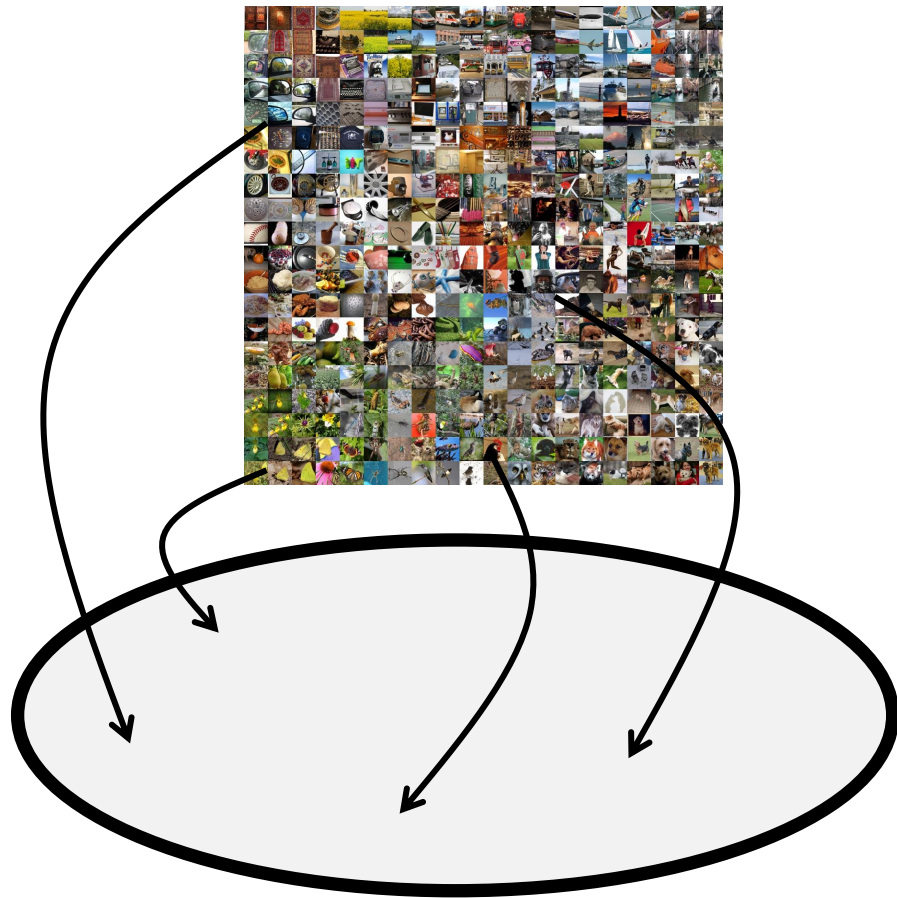
Meta-Training

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

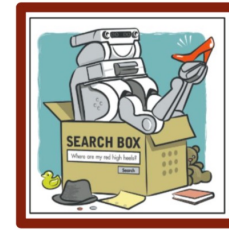
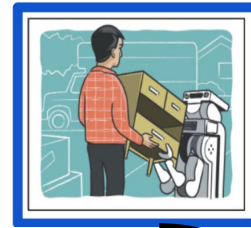
Meta-Testing

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

Latent Variable Model Intuition



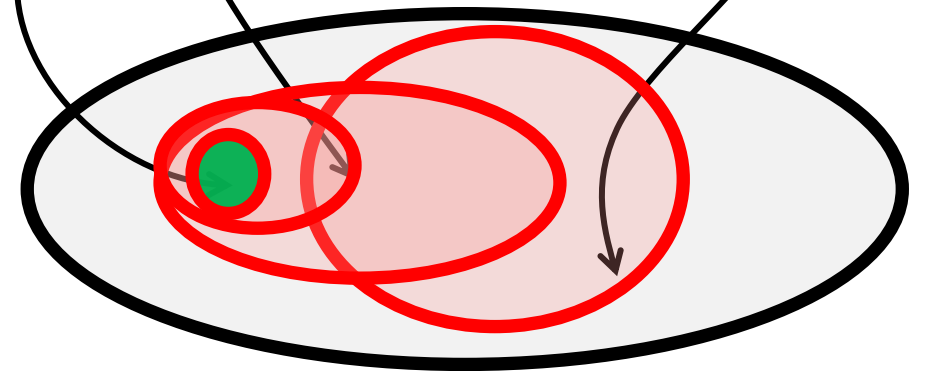
Different images correspond to different z



...



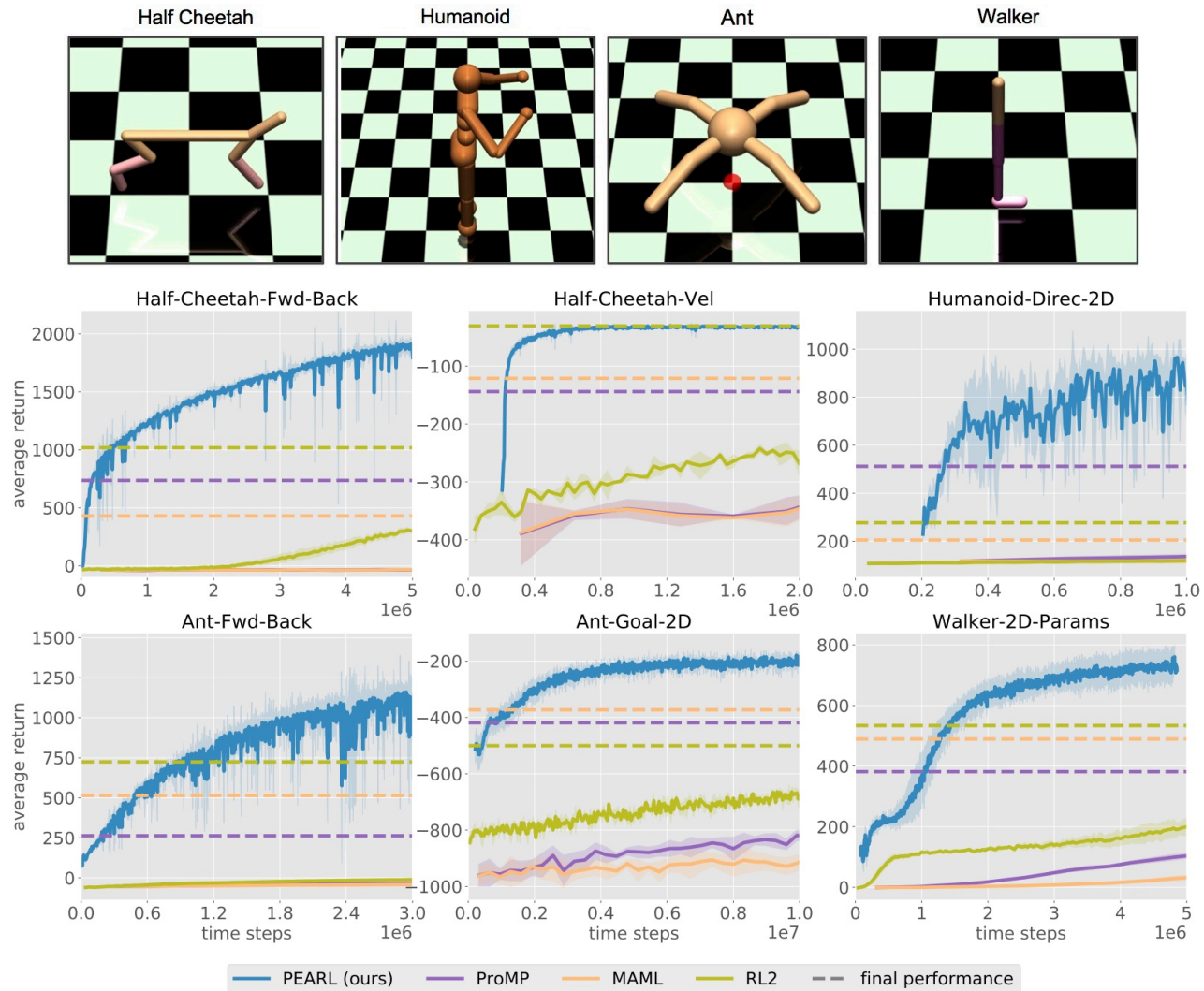
Latent Space



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

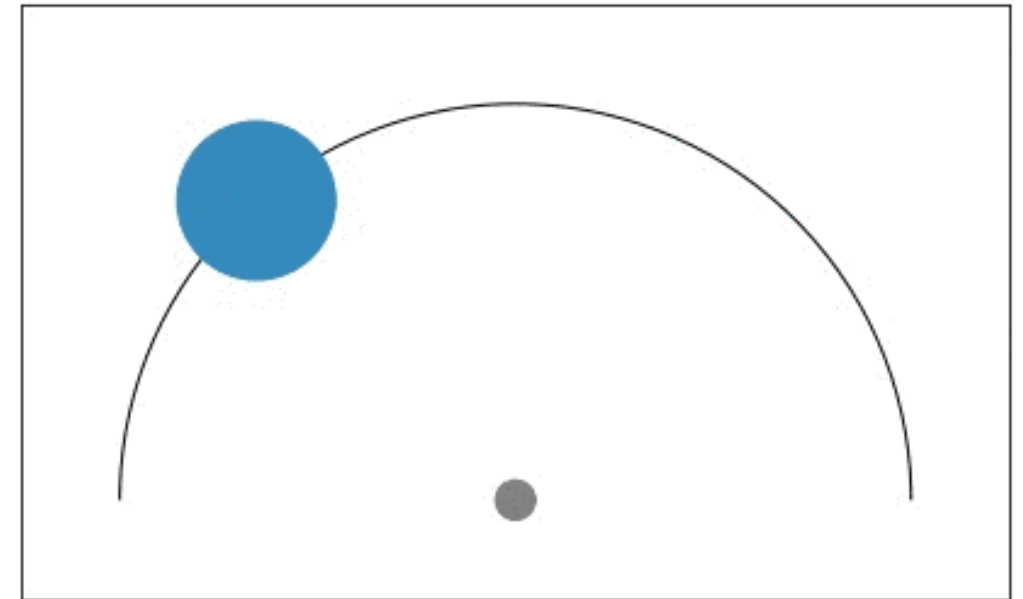
How well does it work?

Quantitative:



Gains mainly from off-policy RL

Exploration:



How well does it work?



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

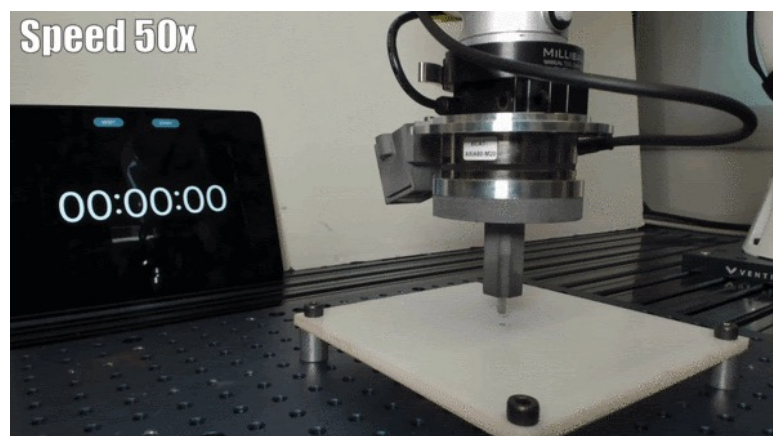
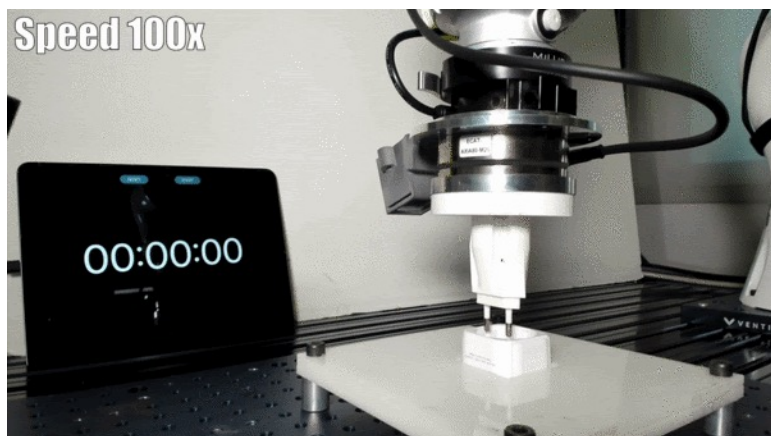
Car-plug-4p

Metal-peg-sq

Car-plug-3p

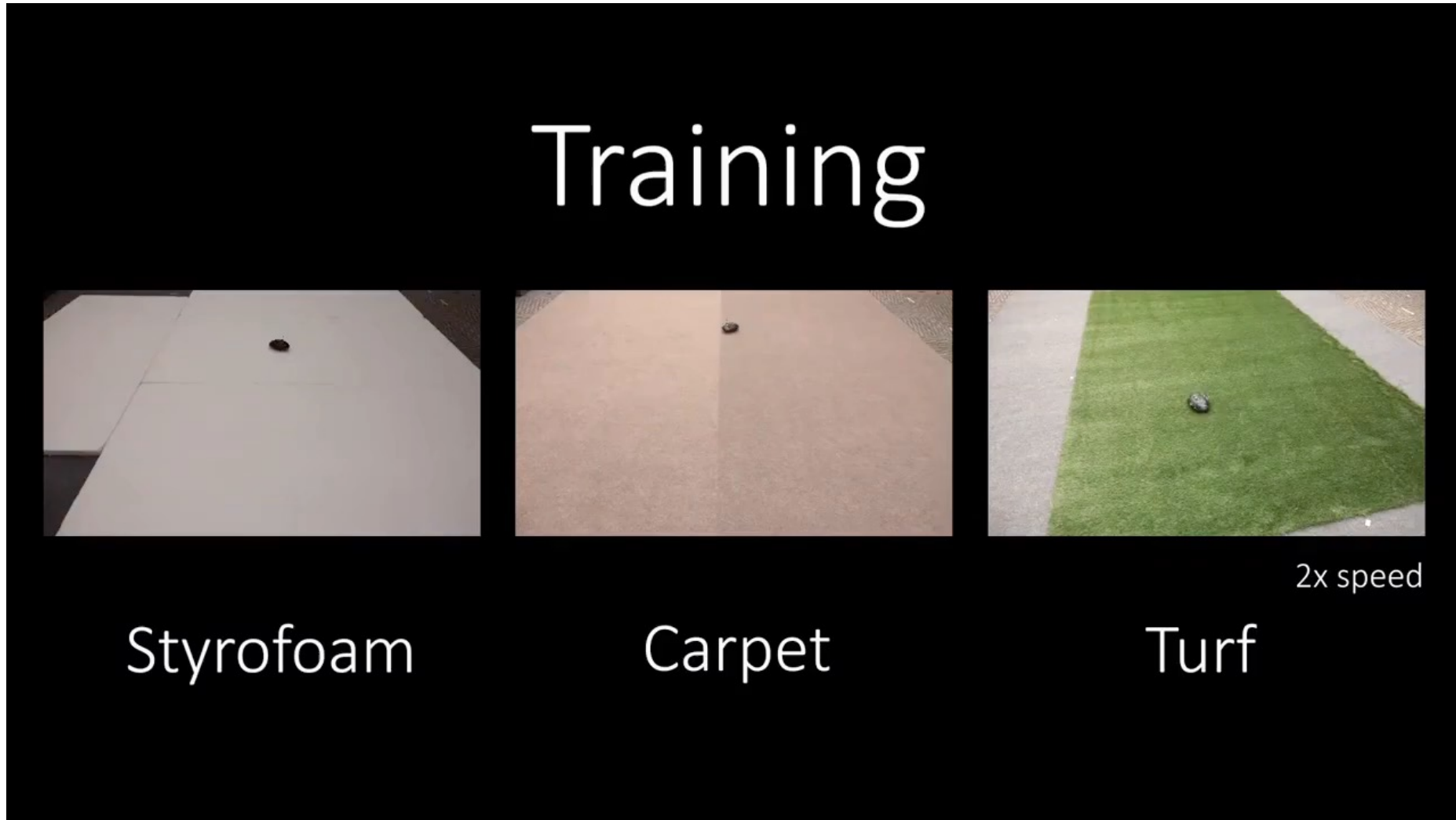
EU-AC-plug

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



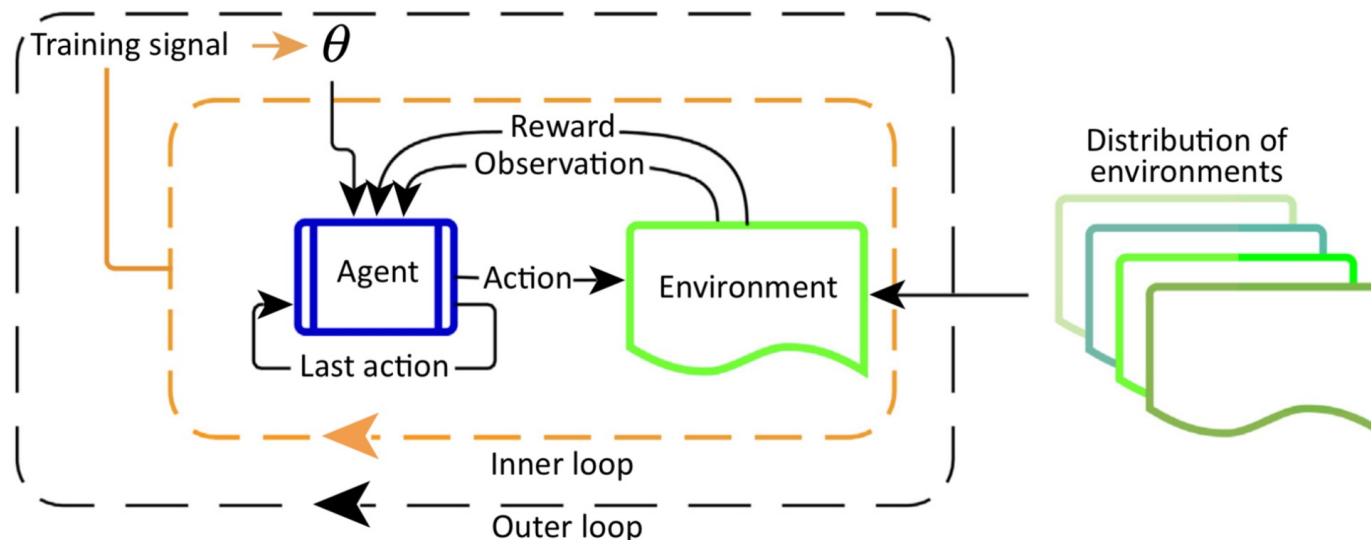
So meta-RL is cool, does it actually work?

Adapting to different terrains/robot conditions



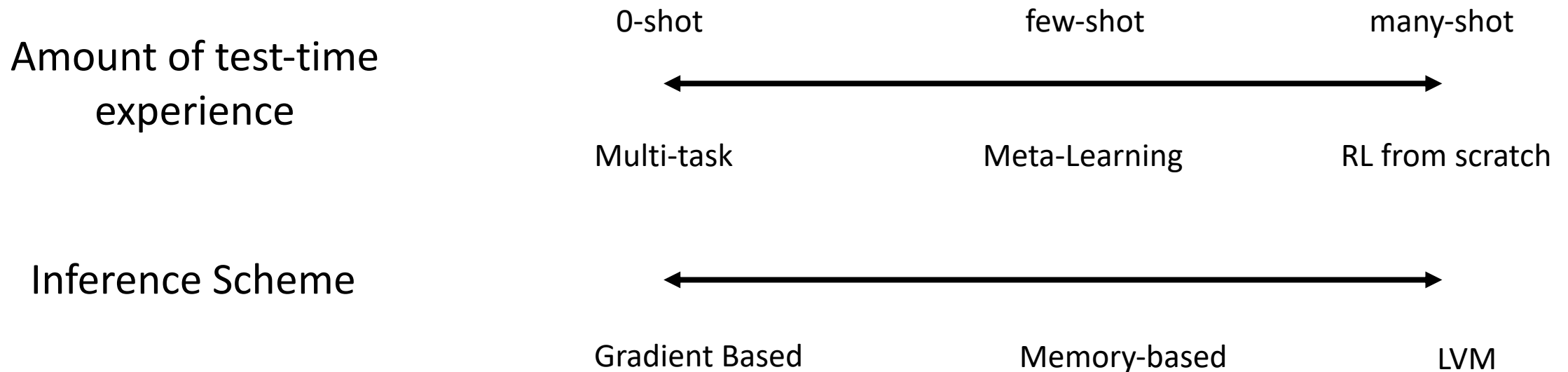
Takeaways from meta-RL

- 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



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: Distal: 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: RL² (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: Multi-task RL formalism



Multi-Task Reinforcement Learning



Meta-Reinforcement Learning



Why offline RL?

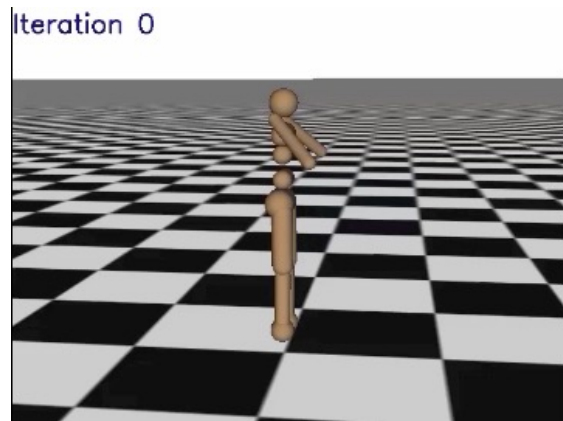


Methods for offline RL

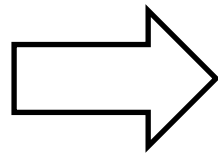
Cost of Real World Data Collection



40 days



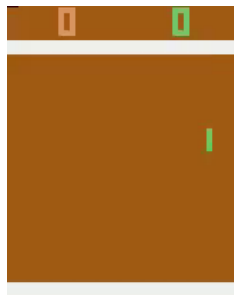
10 days



Real world data is not free!

Where does vanilla RL fall short?

Sample Efficiency

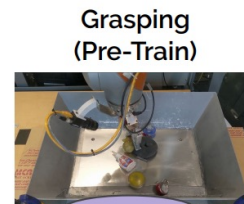


40 days



8 years

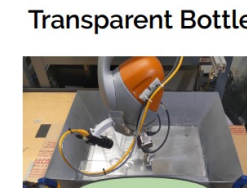
Generalization



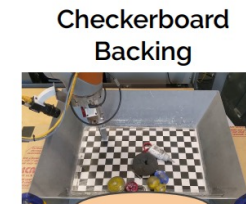
Grasping
≈608,000
 $\mathcal{D}_{\text{grasp}}$ | 86%



Harsh Light
800
 $\mathcal{D}_{\text{harsh}}$ | 32%



Bottles
800
 $\mathcal{D}_{\text{bottles}}$ | 49%



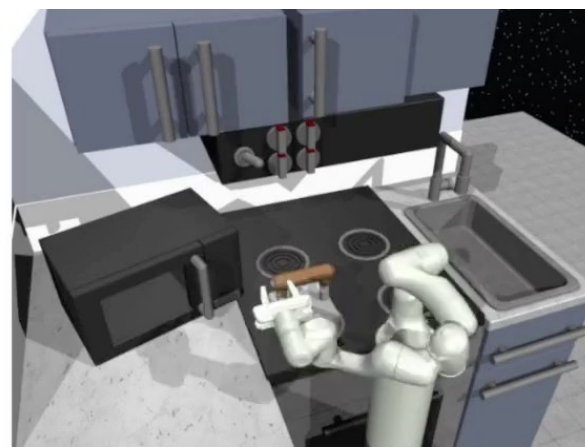
Checker
800
 $\mathcal{D}_{\text{checker}}$ | 50%

Julian et al

Exploration



Long Horizon Goal

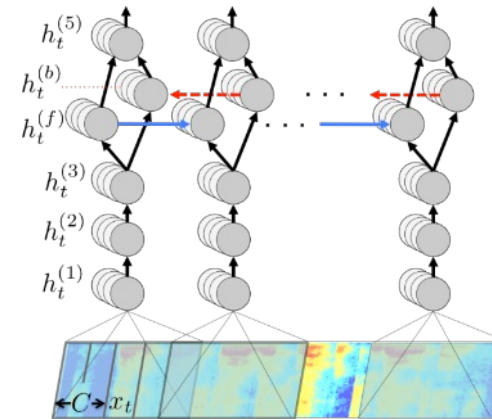
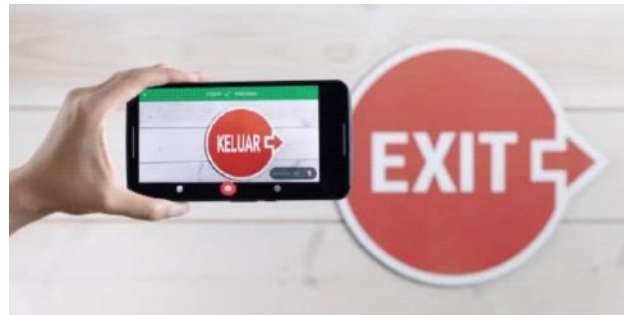
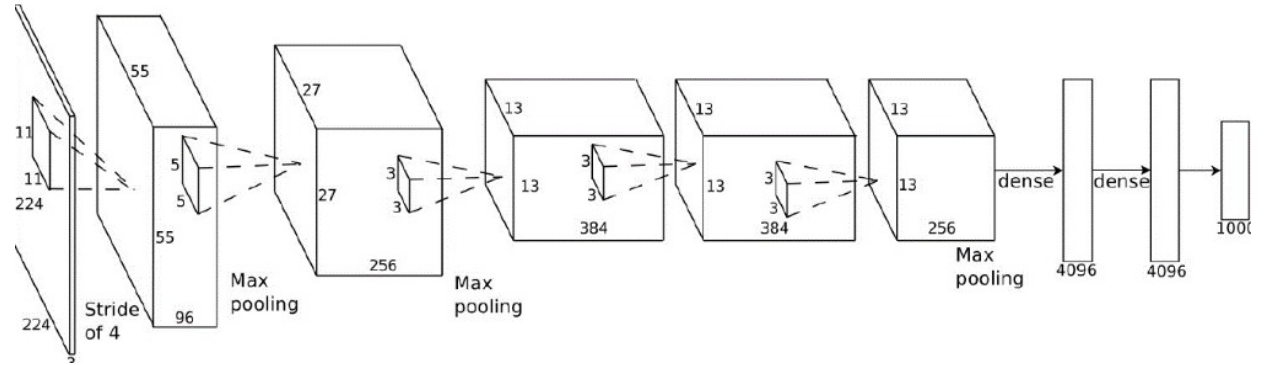
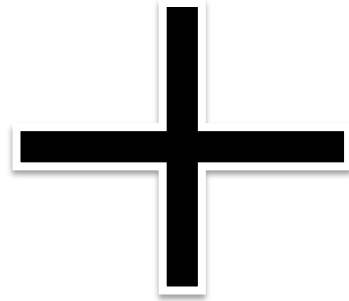


RPL Policy

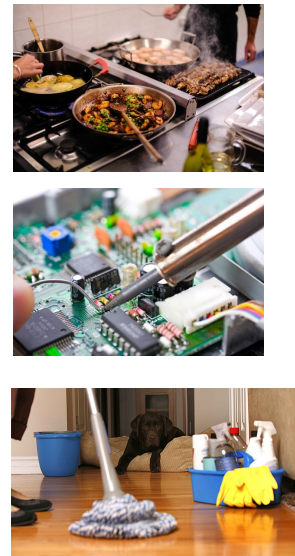
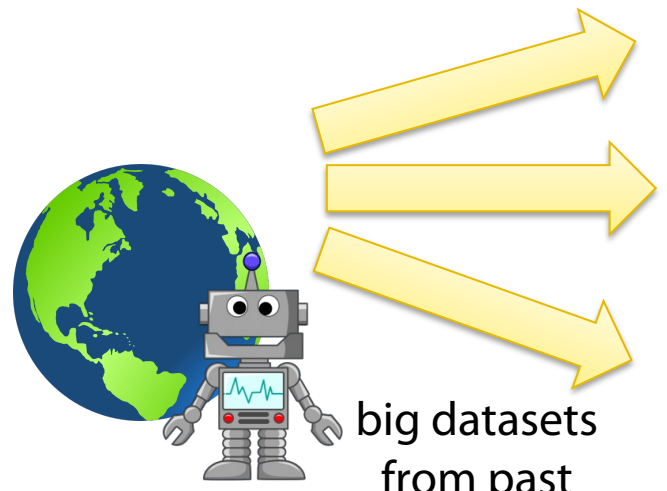
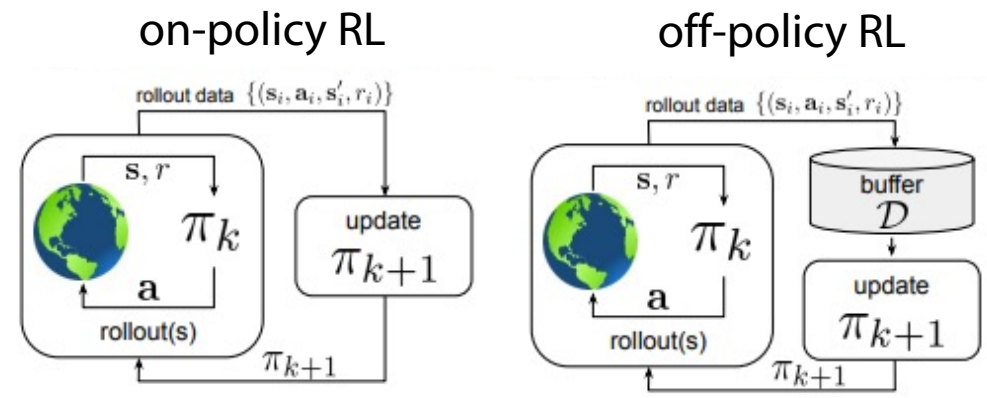
Gupta et al



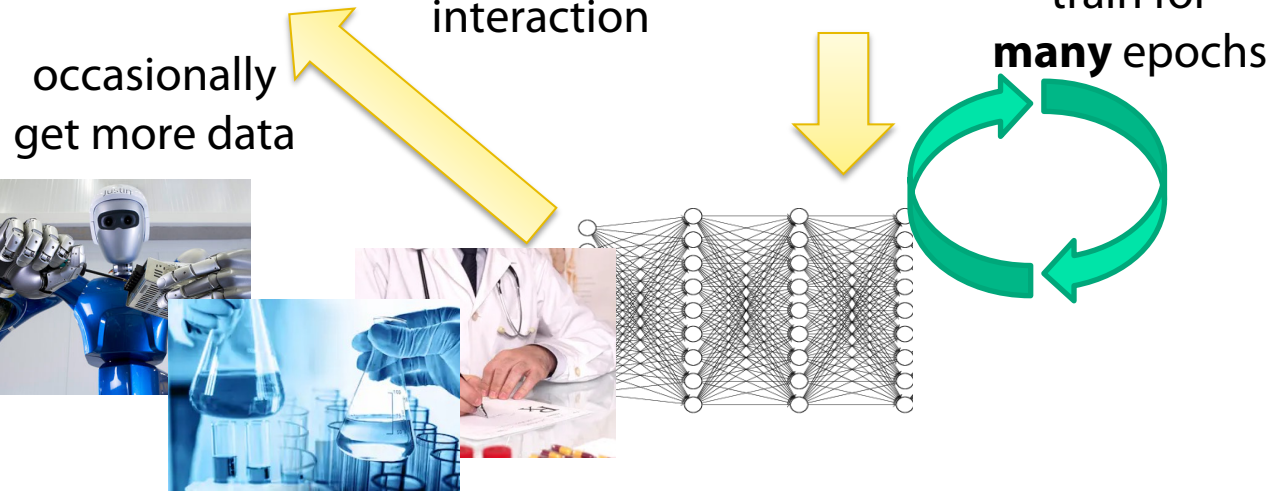
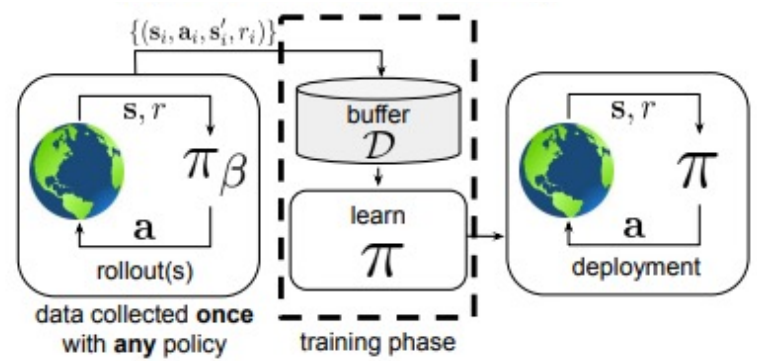
What makes modern machine learning work?



Can we develop data-driven RL methods?

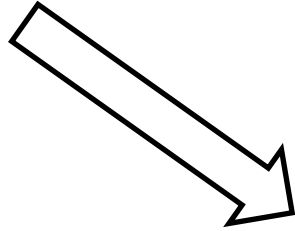


Data-driven reinforcement learning

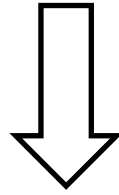


Transfer in Reinforcement Learning

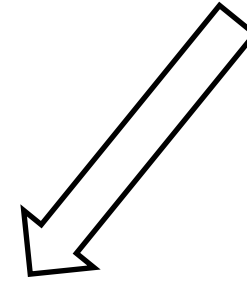
Other tasks



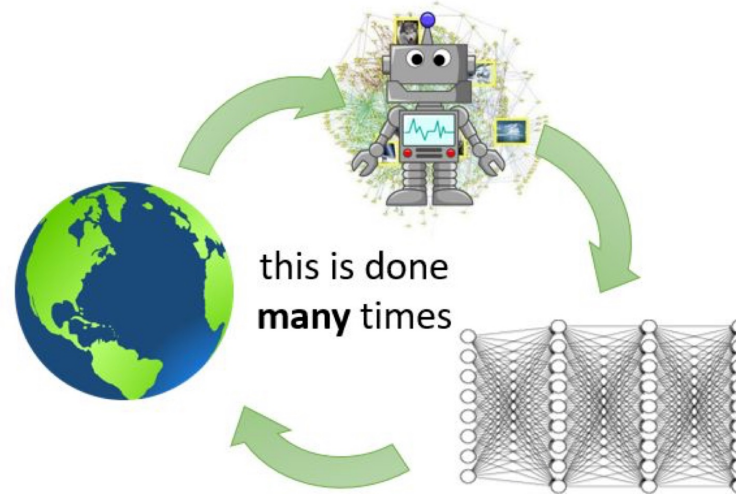
Prior Experience on
the same task



Human supervision

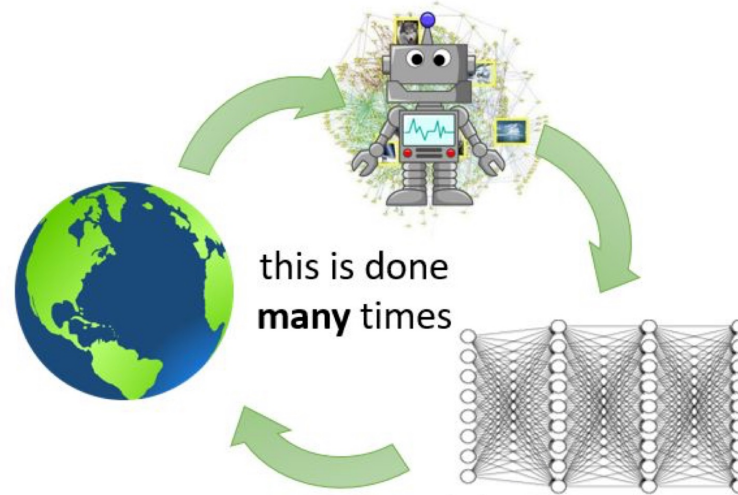
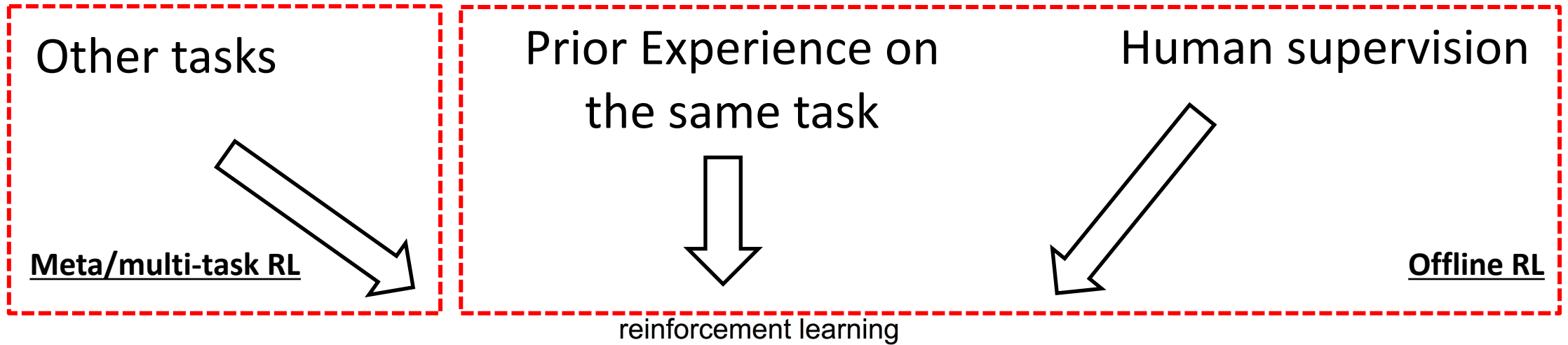


reinforcement learning



Learning from scratch is impractical, leverage different sources of prior information!

Transfer in Reinforcement Learning



Learning from scratch is impractical, leverage different sources of prior information!

Transfer from Prior Datasets in Supervised Learning



Unsupervised Pre-training

Untrained
GPT-3

Expensive training on massive datasets

Dataset: 300 billion tokens of text

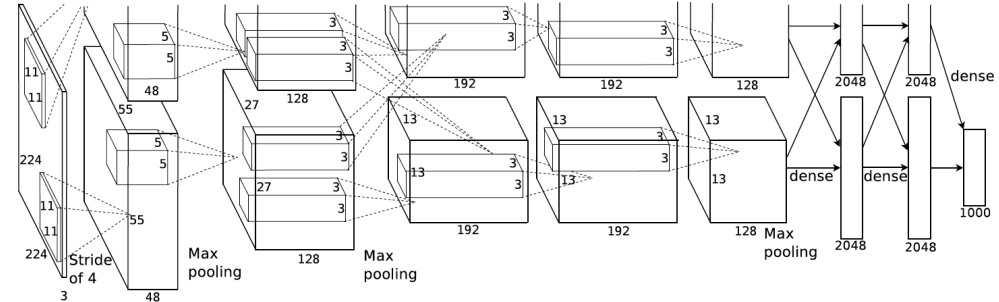
Objective: Predict the next word

Example:

a robot must ?

GPT-3

Pretraining



Input Prompt: Recite the first law of robotics

GPT-3

Output:

A robot may not injure a human being or, through inaction, allow a human being to come to harm.

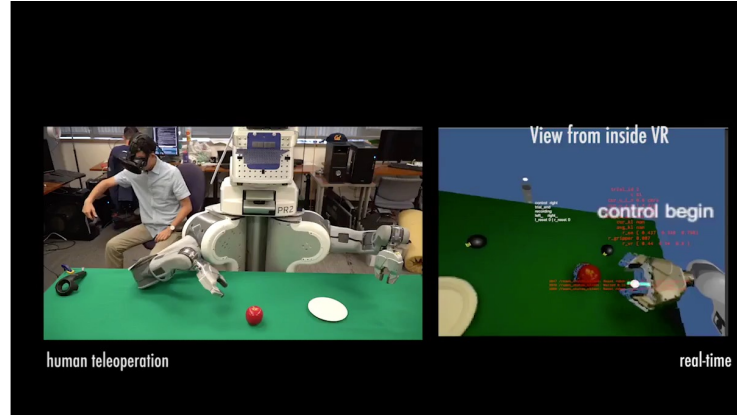
Transfer from Prior Datasets in RL

Large scale datasets



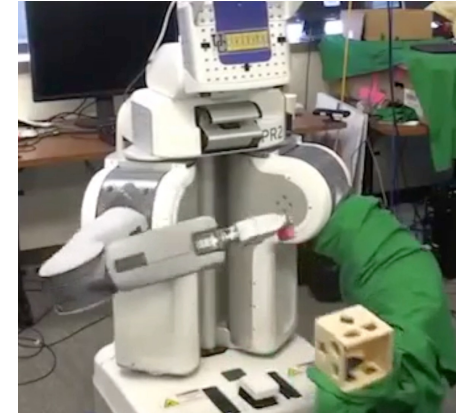
Damen et al

Teleoperated Data



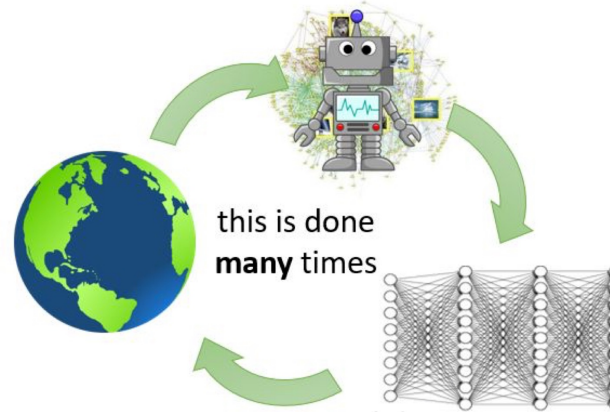
Zhang et al

Previous Experiments



Levine et al

reinforcement learning



Sample efficiency

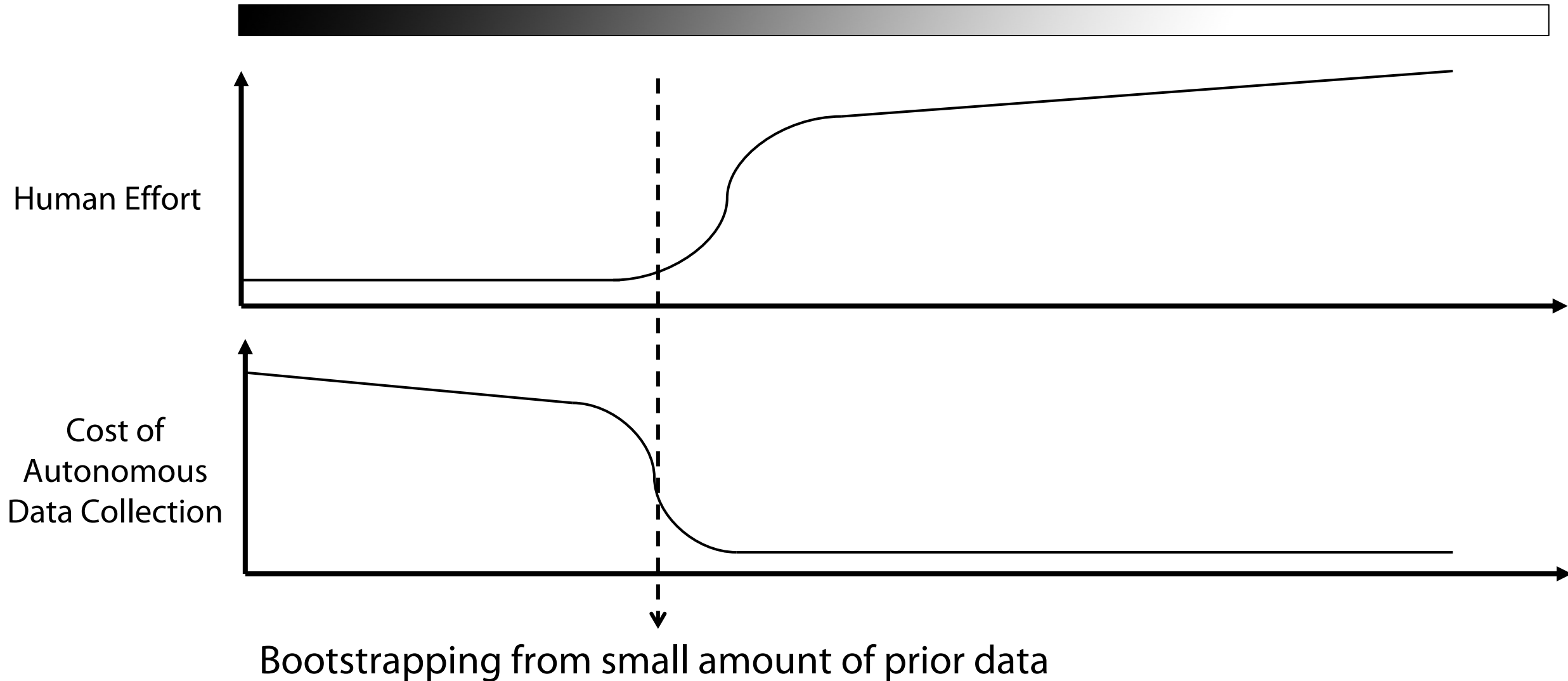
Generalization

Exploration

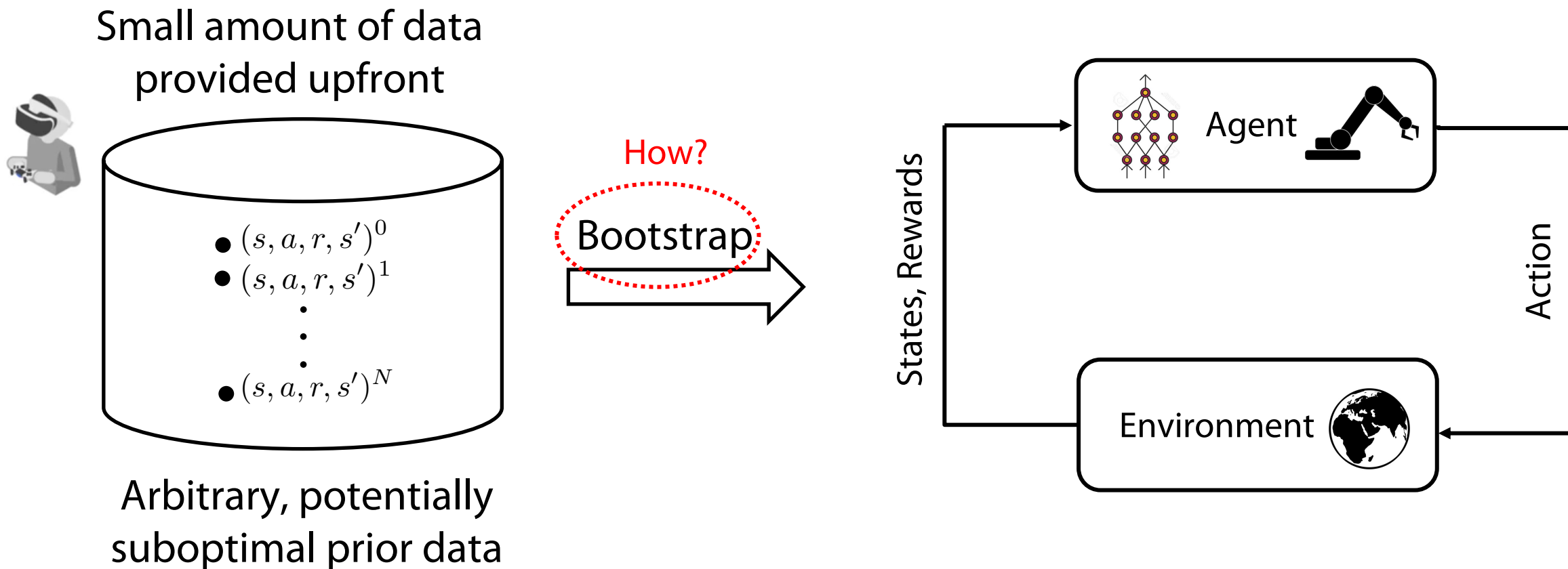
Offsetting the Cost of Real World Data

RL from scratch

Large-Scale Imitation Learning



Making the RL problem more tractable



How can we bootstrap RL with prior data to improve data collection?

Lecture Outline

Recap: Multi-task RL formalism



Multi-Task Reinforcement Learning



Meta-Reinforcement Learning



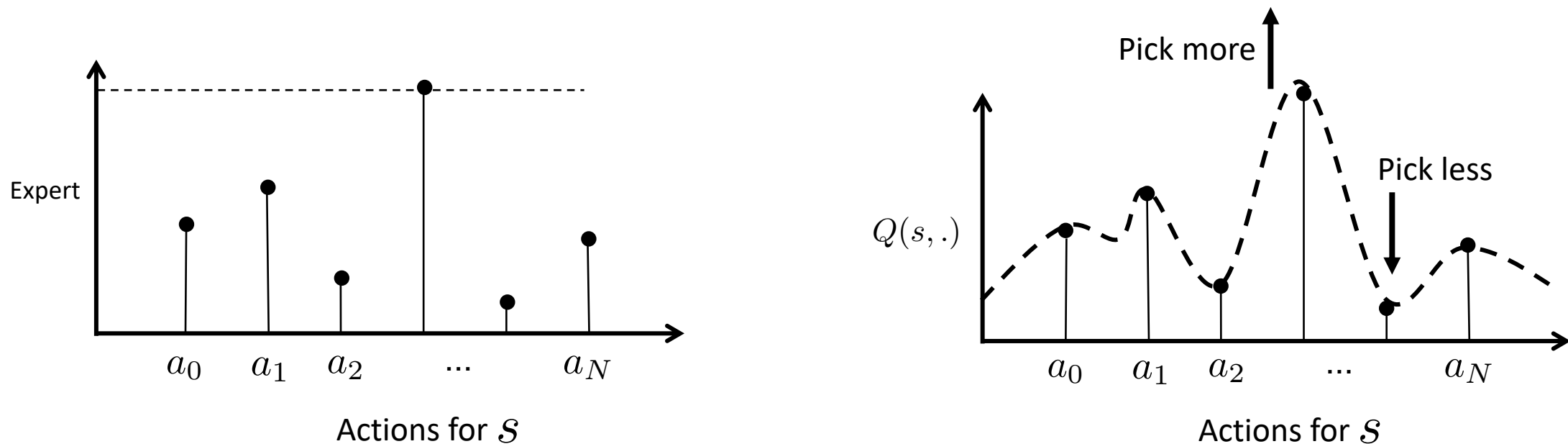
Why offline RL?



Methods for offline RL

Off-Policy Bootstrapping from Suboptimal Data

Imitation learning no longer works as performance may be arbitrarily suboptimal!

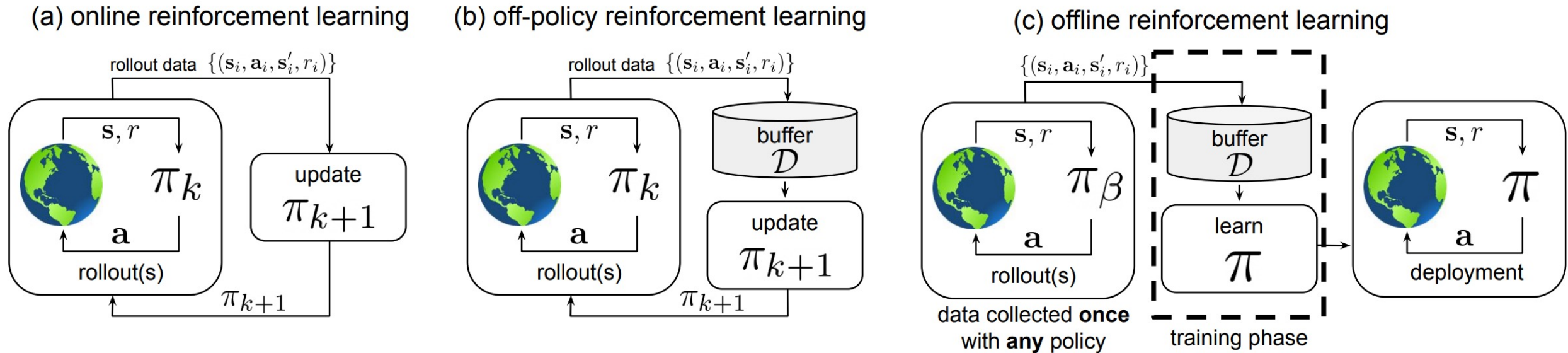


If we were able to estimate long term “goodness” of actions, we can use them to pick actions

↓
Q-function

Offline RL Problem Setting

Go from RL with data collection -> learning policies offline from large datasets



Why would we do this?

- Leverage large existing datasets -> sample efficiency, generalization, exploration
- Useful debugging tool for fundamental problems in RL

Formalism for Offline RL

$$\max_{\theta} \mathbb{E}_{s_0 \sim \mu_0(s), a_t \sim \pi_{\theta}(a_t | s_t), s_{t+1} \sim \mathcal{P}(s_{t+1} | s_t, a_t)} \left[\sum_{t=0}^T \gamma^t R(s_t, a_t) \right]$$

Offline dataset: $\mathcal{D} = \{(s, a, s', r)\}_{i=1}^n$

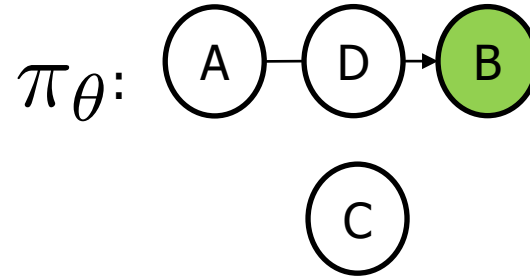
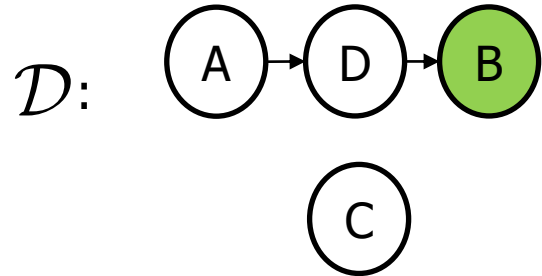
Behavior policy generating offline dataset: $\pi_{\beta}(a | s)$

Goal:

- With only sampling access to \mathcal{D} , learn π_{θ}^* for the original RL objective
- Perform better than the original behavior policy

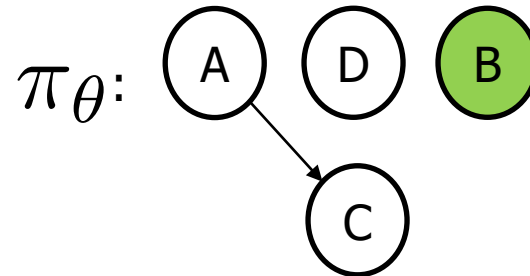
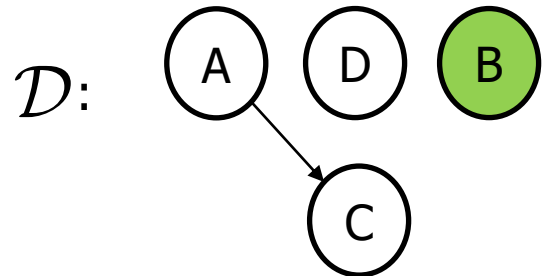
When might offline RL be effective?

If \mathcal{D} is only expert demos:



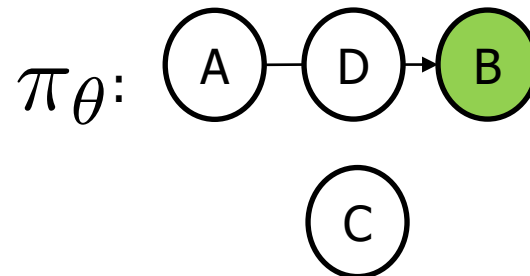
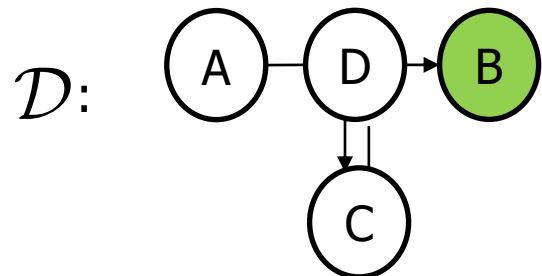
π_θ as good as \mathcal{D}

If \mathcal{D} is sub-optimal data with insufficient coverage:



π_θ, \mathcal{D} are both suboptimal

If \mathcal{D} is sub-optimal data with insufficient coverage:

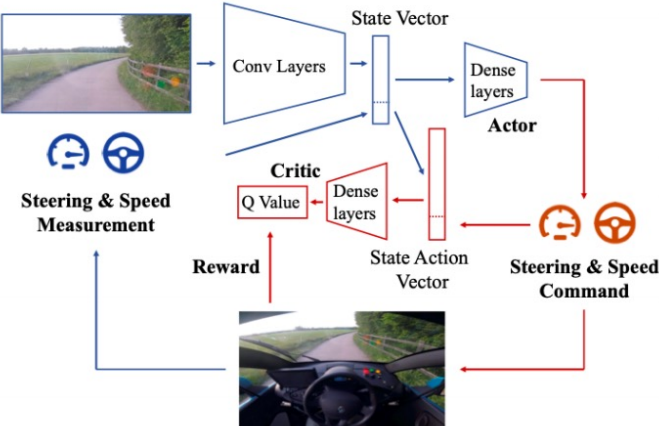


π_θ better than \mathcal{D}

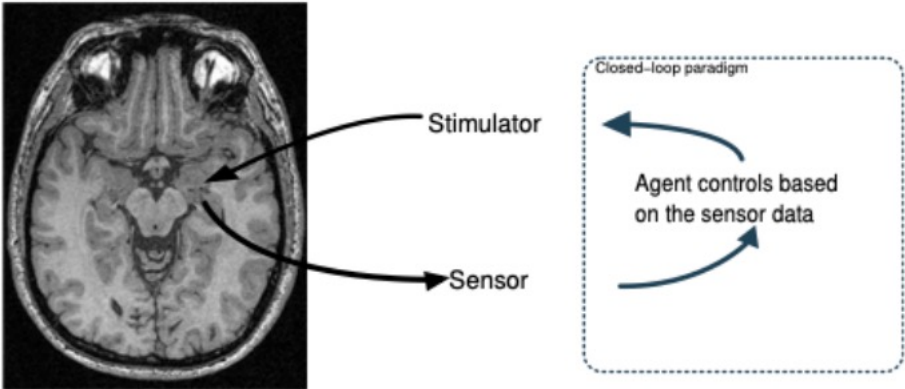
Application Domains

Useful in domains where data collection is risky/hard, but human datasets available

Driving



Healthcare



Dialogue

```
[User]: i'm just at work.
[RL - Model avg Psi]: 🟡🟡🟡🟡 i'm not sure, but i'm glad to see what you did!
[User]: you're boring.
[RL - Model avg Psi]: 🟡🟡🟡 why would you say this? what's up to you?!?!?!?!?!
[User]: i'm sorry! I didn't mean to offend you.
[RL - Model avg Psi]: 🟡🟡 i'm not sure, and i think i'm going to do this, and i'm sure you'll go!
[User]: i will go, you're being weird!
[RL - Model avg Psi]: 🟡🟡 we are the same, we don't understand that, sir.
[User]: we are not the same. i'm a human and you're a bot.
[RL - Model avg Psi]: 🟡🟡💖🟡 that's really nice to me too.
>
```

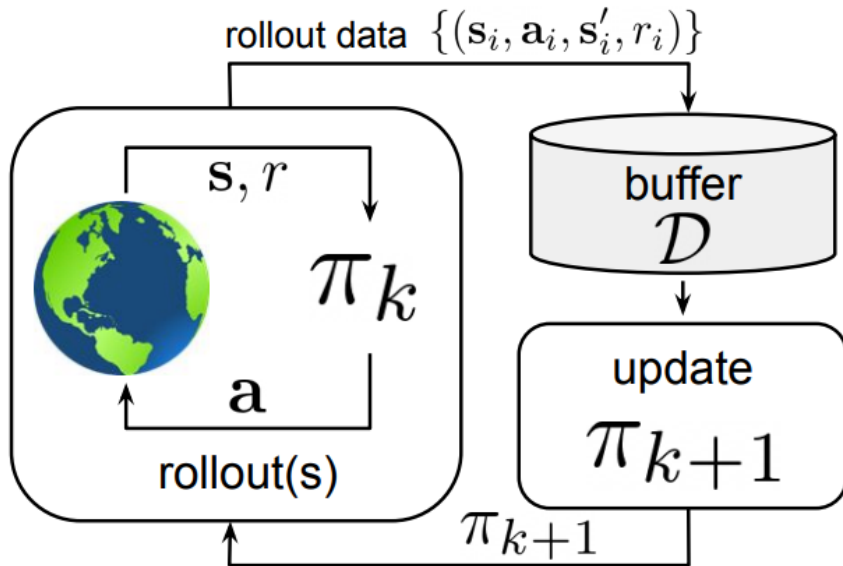
Robotics



Visuals from Levine et al

Attempt 1: Off-Policy RL

(b) off-policy reinforcement learning



Can try directly adding offline data to \mathcal{D}

Train by sampling from \mathcal{D} (no sampling in env):

1. Add offline data to the replay buffer
2. Minimize Bellman Equation



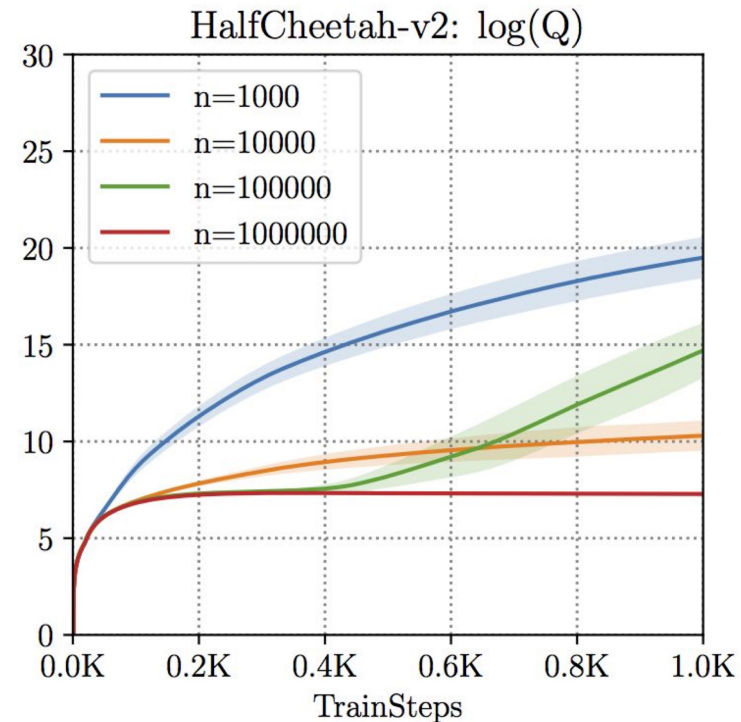
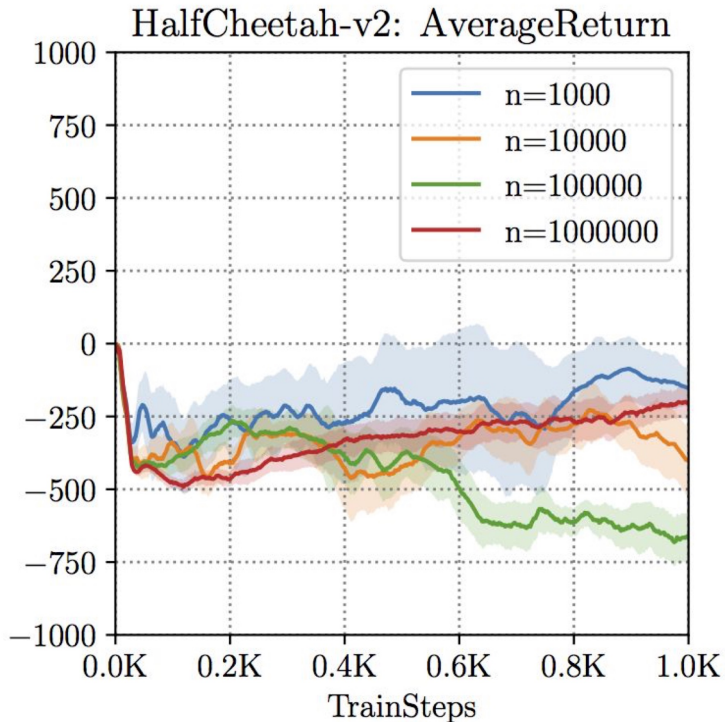
$$\min(Q(s, a) - (r(s, a) + \gamma \max_{a'} Q(s', a')))^2$$

3. Optimize actor $\pi(a|s)$ wrt $Q^\pi(s, a)$

$$\pi \leftarrow \arg \max_{\pi} \mathbb{E}_{\pi} [Q^{\pi}]$$

Attempt 1: Off-Policy RL

Empirical performance with vanilla off-policy RL on offline data



- Returns don't increase but Q-values diverge
- Not classical overfitting!
- More data does not improve performance

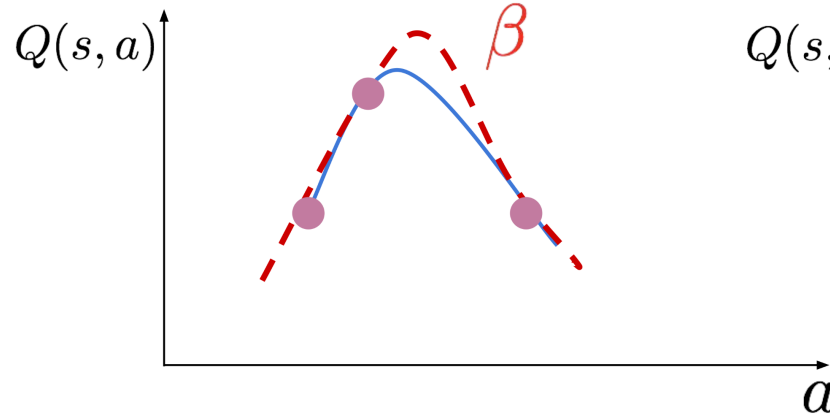
Divergence because of distribution shift

Distribution shift in Offline RL

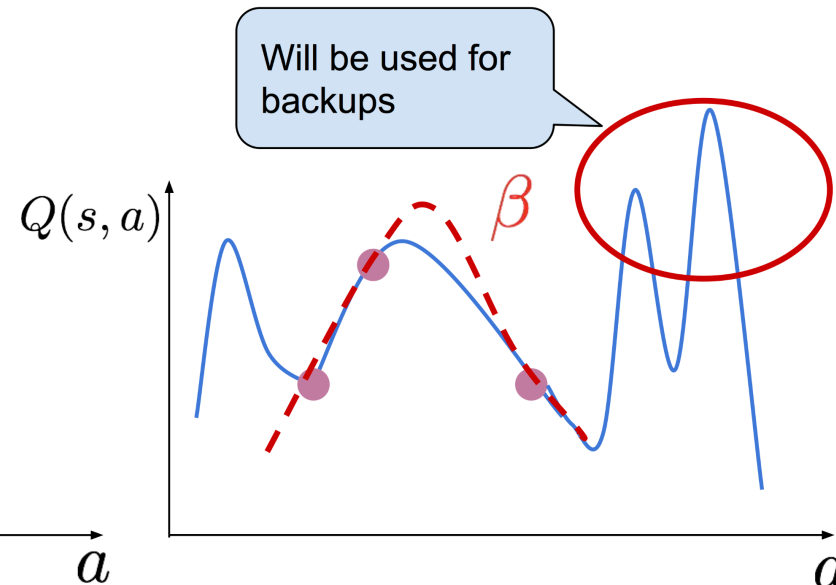
$$Q^* = \mathcal{T}^* Q^* \quad ; \quad (\mathcal{T}^* \hat{Q})(s, a) := R(s, a) + \gamma \mathbb{E}_{T(s'|s, a)} [\max_{a'} \hat{Q}(s', a')]$$

Can bootstrap on OOD actions. Q can be arbitrarily overestimated

$$Q := \arg \min_{\hat{Q}} \mathbb{E}_{s \sim \mathcal{D}, a \sim \beta(a|s)} \left[(\hat{Q}(s, a) - (\mathcal{T}^* \hat{Q})(s, a))^2 \right]$$



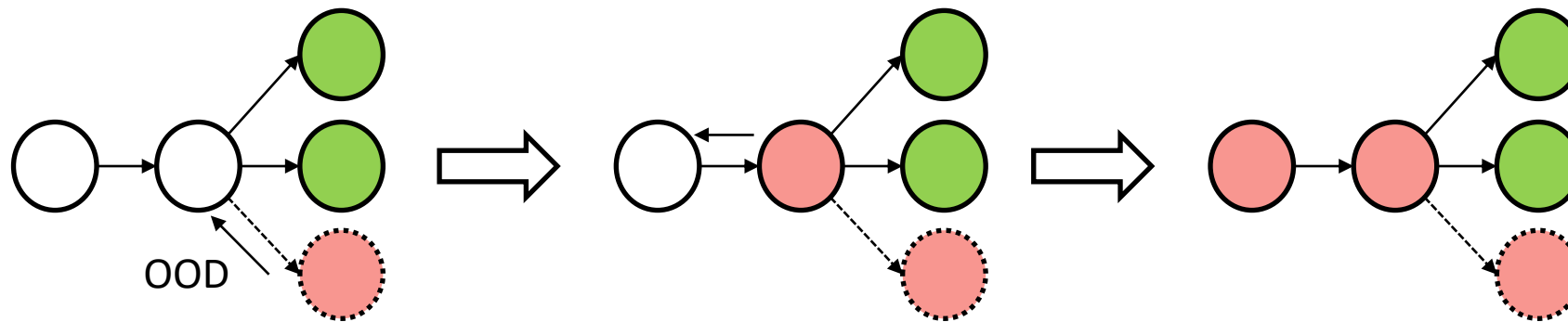
Q-values on training data



Q-values being backed up

Why is distribution shift problematic in Offline RL?

- When Q is trained on $\pi_\beta(a|s)$, it may not be accurate for arbitrary $\pi(a|s)$
 - Some a, s may just be very OOD/out of support.
- Overestimated Q-values can continue to be backed up erroneously.

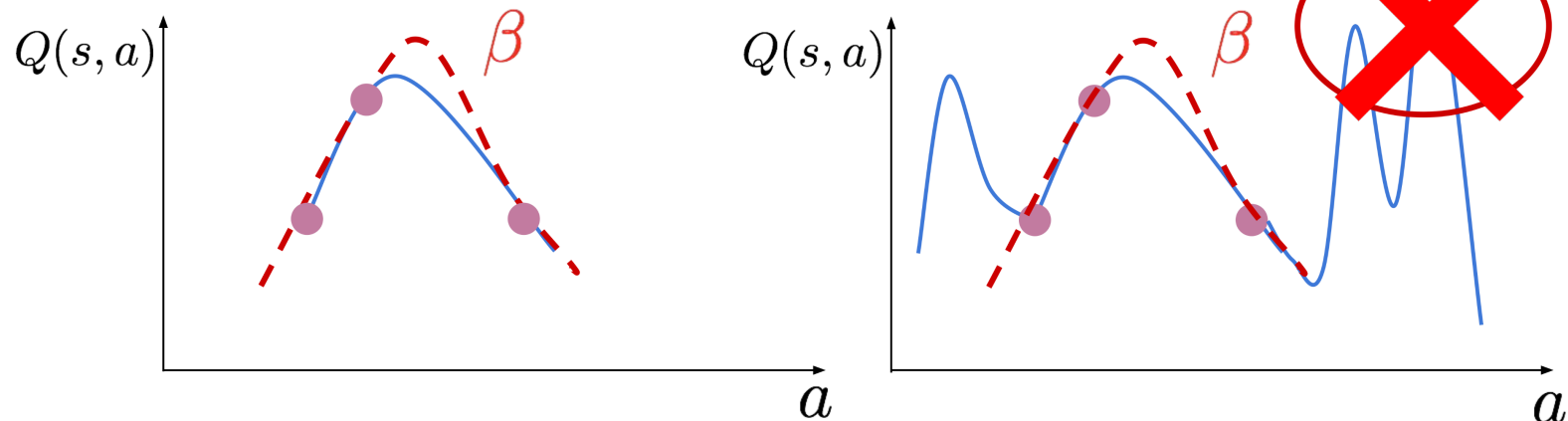


Online data collection corrects over-optimistic Q, but not so offline!

Tackling distribution shift in offline RL

Key Idea for controlling distribution shift in offline RL: (minimize choosing OOD actions)

- Policy constraint methods
- Lower bounding returns/Q-values



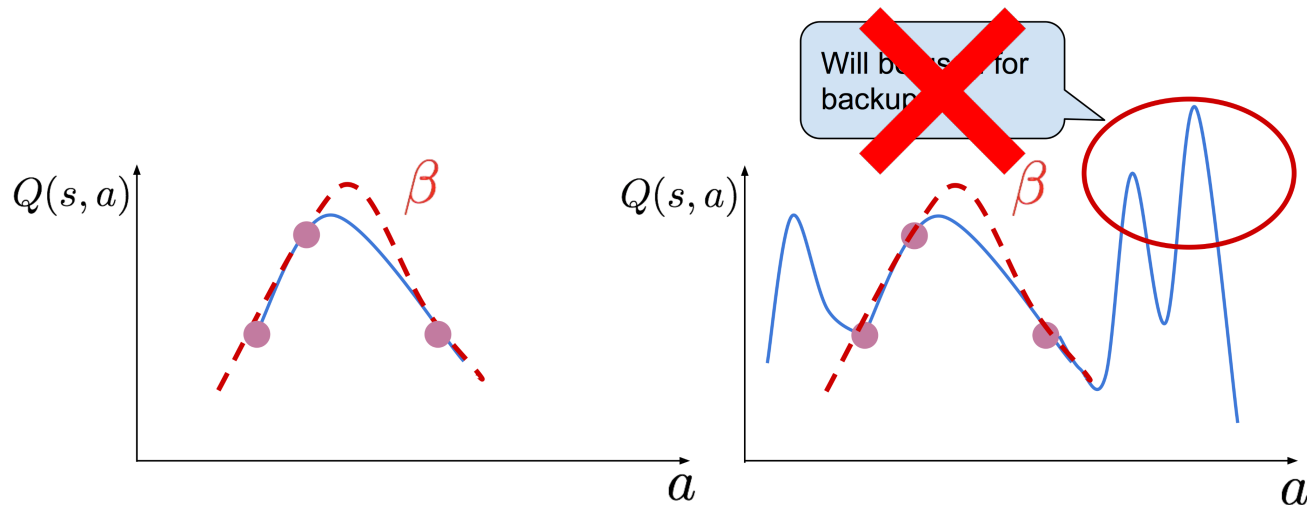
Attempt 2: Policy Constraint Algorithms

Idea: Constrain the actor update to remain close to the behavior policy.

$$\hat{Q}_{k+1}^{\pi} \leftarrow \arg \min_Q \mathbb{E}_{(s, a, s') \sim \mathcal{D}} \left[\left(Q(s, a) - \left(r(s, a) + \gamma \mathbb{E}_{a' \sim \pi_k(a'|s')} [\hat{Q}_k^{\pi}(s', a')] \right) \right)^2 \right]$$

$$\pi_{k+1} \leftarrow \arg \max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi(a|s)} [\hat{Q}_{k+1}^{\pi}(s, a)] \right] \text{ s.t. } D(\pi, \pi_{\beta}) \leq \epsilon.$$

Why would this work?



OOD actions violate the constraint

$$\pi(a|s) \approx \pi_{\beta}(a|s)$$

Policy Constraint Algorithms

Different forms of the constraint and optimization leads to different algorithms

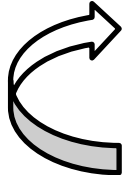
Constraint Form	Constraint Objective	Resulting Algorithm
Support matching	$D = \text{MMD}(\pi_\beta, \pi)$	(Kumar et al. 2019, Laroche et al. 2019, Wu et al. 2019)
Distribution Matching	$D = D_{KL}(\pi \pi_\beta)$	BCQ (Fujimoto et al), (Jaques et al 2019), BRAC (Wu et al)
State-marginal constraints	$D = D_{KL}(d^\pi d^{\pi_\beta})$	AlgaeDICE (Nachum et al)
Implicit distribution constraints	$D = D_{KL}(\pi \pi_\beta)$	AWR (Peng et al), AWAC (Nair et al), CRR (Wang et al)

Implementation of Policy Constraint Algorithms

Pseudocode of a policy constraint algorithm:

1. Add offline data to the replay buffer

2. Minimize Bellman Equation

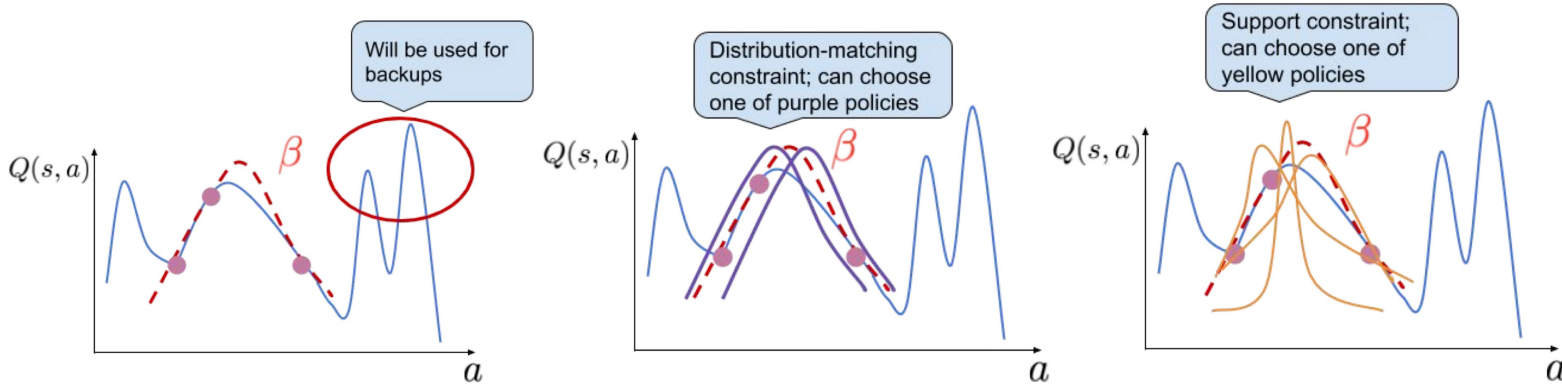

$$\hat{Q}_{k+1}^{\pi} \leftarrow \arg \min_Q \mathbb{E}_{(\mathbf{s}, \mathbf{a}, \mathbf{s}') \sim \mathcal{D}} \left[\left(Q(\mathbf{s}, \mathbf{a}) - \left(r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_k(\mathbf{a}'|\mathbf{s}')} [\hat{Q}_k^{\pi}(\mathbf{s}', \mathbf{a}')] - \alpha \gamma D(\pi_k(\cdot|\mathbf{s}'), \pi_{\beta}(\cdot|\mathbf{s}')) \right) \right)^2 \right]$$

3. Optimize actor $\pi(a|s)$ wrt $Q^{\pi}(s, a)$

$$\pi_{k+1} \leftarrow \arg \max_{\pi} \mathbb{E}_{\mathbf{s} \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} [\hat{Q}_{k+1}^{\pi}(\mathbf{s}, \mathbf{a})] - \alpha D(\pi(\cdot|\mathbf{s}), \pi_{\beta}(\cdot|\mathbf{s})) \right]$$

Requires estimation of behavior policy π_{β} via MLE

Tradeoffs between Policy Constraints



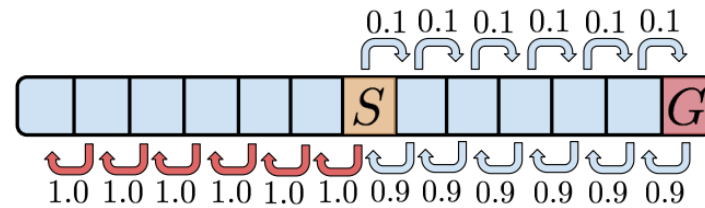
Actions: \rightarrow, \leftarrow



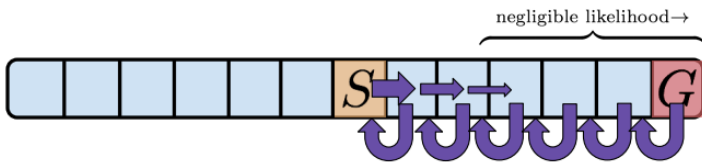
$$r(s, \leftarrow) = r(s, \rightarrow) = 0 \quad r(s, \rightarrow) = 1, r(s, \leftarrow) = -1$$

Initial state: S Goal state: G

(a) 1D-Lineworld Environment



(b) Behavior Policy



(c) Learned Policy via distribution-matching



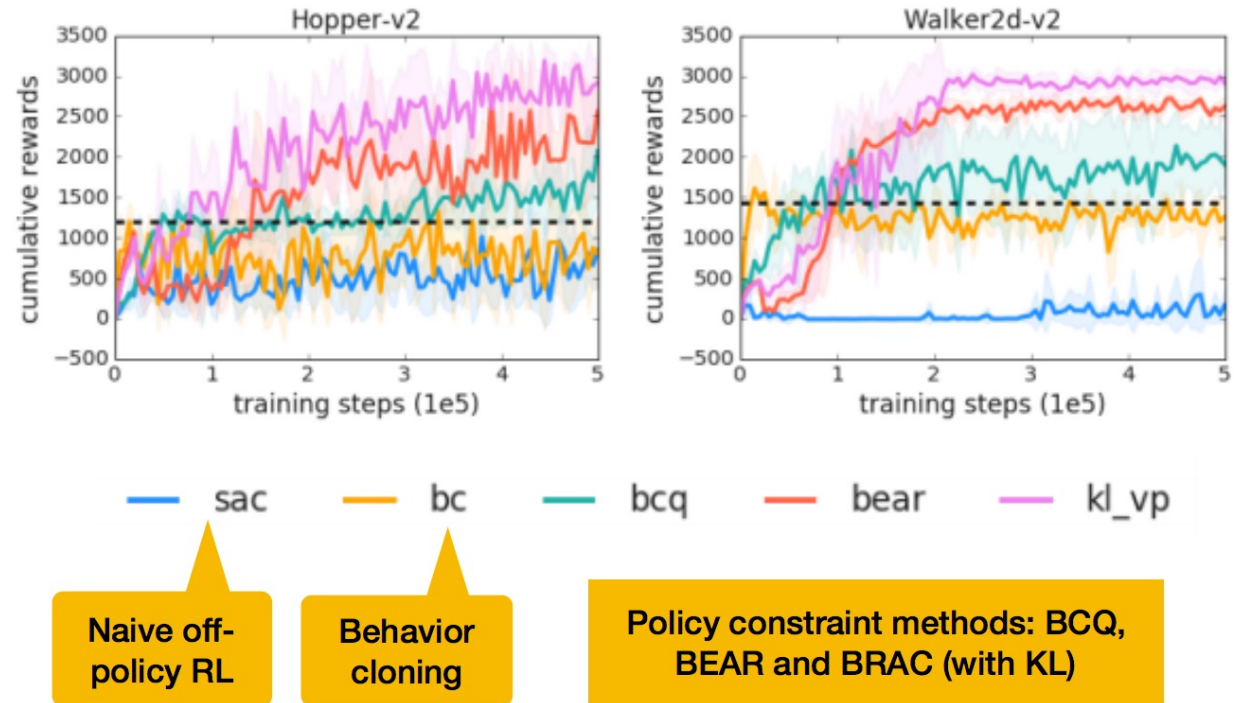
(d) Learned Policy via support-constraint

Support constraints are less pessimistic, works across behavior distributions

Performance of Policy Constraint Algorithms

How well does this work?

- Works significantly better than off-policy RL
- Not good on harder tasks, too pessimistic



	Task Name	SAC	BC	SAC-off	BEAR	BRAC-p	BRAC-v	AWR	cREM	BCQ	aDICE
Maze 2D	maze2d-umaze	62.7	3.8	88.2	3.4	4.7	-16.0	1.0	-15.8	12.8	-15.7
	maze2d-medium	21.3	30.3	26.1	29.0	32.4	33.8	7.6	0.9	8.3	10.0
	maze2d-large	2.7	5.0	-1.9	4.6	10.4	40.6	23.7	-2.2	6.2	-0.1
AntMaze	antmaze-umaze	0.0	65.0	0.0	73.0	50.0	70.0	56.0	0.0	78.9	0.0
	antmaze-umaze-diverse	0.0	55.0	0.0	61.0	40.0	70.0	70.3	0.0	55.0	0.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	antmaze-medium-diverse	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.7	0.0
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.2	0.0

Challenges with Policy Constraint Algorithms

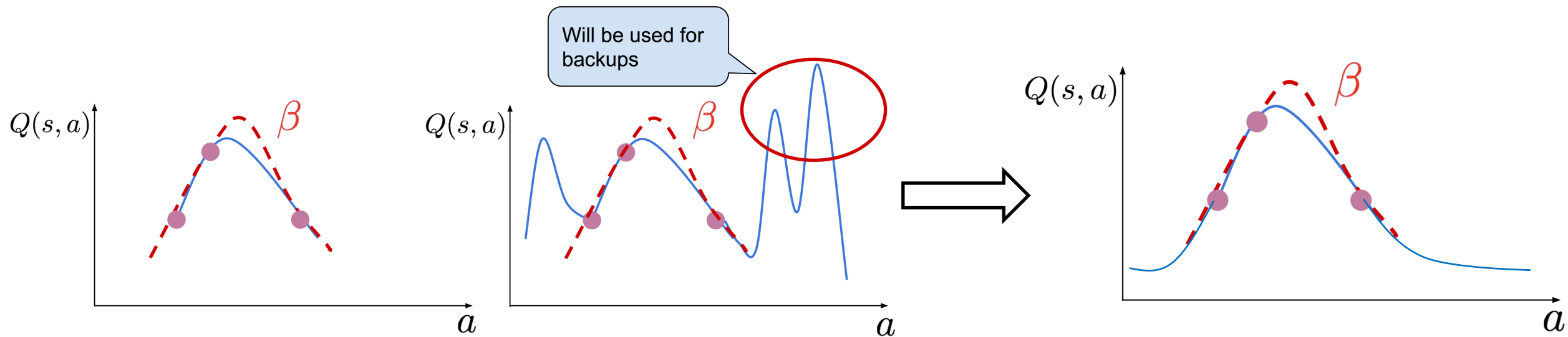
$$\hat{Q}_{k+1}^\pi \leftarrow \arg \min_Q \mathbb{E}_{(\mathbf{s}, \mathbf{a}, \mathbf{s}') \sim \mathcal{D}} \left[\left(Q(\mathbf{s}, \mathbf{a}) - \left(r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_k(\mathbf{a}'|\mathbf{s}')} [\hat{Q}_k^\pi(\mathbf{s}', \mathbf{a}')] \right) \right)^2 \right]$$
$$\pi_{k+1} \leftarrow \arg \max_\pi \mathbb{E}_{\mathbf{s} \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} [\hat{Q}_{k+1}^\pi(\mathbf{s}, \mathbf{a})] \right] \text{ s.t. } D(\pi, \pi_\beta) \leq \epsilon.$$

1. Requires challenging estimation of behavior policy
2. Constraint can often be **too** pessimistic

Attempt 3: Lower Bounding Q-Values

Idea: Push down over-optimistic Q-values, rather than completely avoiding OOD

- Less pessimistic than avoiding all OOD actions. Some OOD actions may not be bad!



Model-Free Lower Bounding Q-Values

Conservative Q-Learning:

Reduce overestimation by forcing them to lower bound the true Q-value

Push down big Q values



$$\min_Q \max_{\mu} \alpha \mathbb{E}_{s \sim \mathcal{D}, a \sim \mu(a|s)} [Q(s, a)] +$$

$$\mathbb{E}_{(s, a, s') \sim \mathcal{D}} [(Q(s, a) - (r(s, a) + \mathbb{E}_{\pi} [Q(s', a')]))^2]$$

Minimize Bellman Equation



$$\hat{Q}^{\pi}(s, a) \leq Q^{\pi}(s, a), \forall s, a$$

Model-Free Lower Bounding Q-Values

Can tighten bound further by also pushing up Q-values in the data

Push down big Q values

Push up Q values in data

$$\min_Q \max_{\mu} \alpha \left(\mathbb{E}_{s \sim \mathcal{D}, a \sim \mu(a|s)} [Q(s, a)] - \mathbb{E}_{s \sim \mathcal{D}, a \sim \hat{\pi}_{\beta}(a|s)} [Q(s, a)] \right) +$$

Minimize Bellman Equation $\mathbb{E}_{(s,a,s') \sim \mathcal{D}} [(Q(s, a) - (r(s, a) + \mathbb{E}_{\pi} [Q(s', a')]))^2]$

$$\mathbb{E}_{\pi(a|s)} \left[\hat{Q}^{\pi}(s, a) \right] \leq \mathbb{E}_{\pi(a|s)} [Q^{\pi}(s, a)], \forall s \in \mathcal{D}$$

CQL Pseudocode

Pseudocode:

- Simply modify the critic update, same actor update and replay buffer.

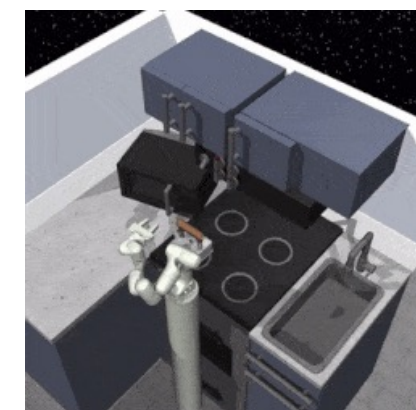
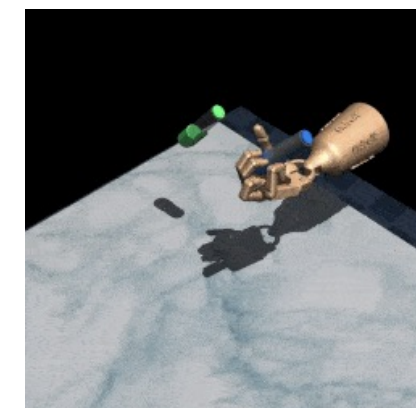
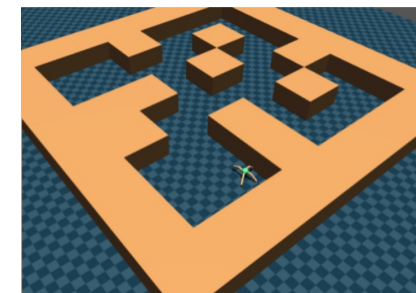
Algorithm 1 Conservative Q-Learning (both variants)

- 1: Initialize Q-function, Q_θ , and optionally a policy, π_ϕ .
- 2: **for** step t in $\{1, \dots, N\}$ **do**
- 3: Train the Q-function using G_Q gradient steps on objective from Equation 4
 $\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \text{CQL}(\mathcal{R})(\theta)$
 (Use \mathcal{B}^* for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
- 4: (only with actor-critic) Improve policy π_ϕ via G_π gradient steps on ϕ with SAC-style entropy regularization:
 $\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\phi(\cdot|\mathbf{s})} [Q_\theta(\mathbf{s}, \mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
- 5: **end for**

How well does this do?

Performs surprisingly well!

Domain	Task Name	BC	SAC	BEAR	BRAC-p	BRAC-v	CQL(\mathcal{H})	CQL(ρ)
AntMaze	antmaze-umaze	65.0	0.0	73.0	50.0	70.0	74.0	73.5
	antmaze-umaze-diverse	55.0	0.0	61.0	40.0	70.0	84.0	61.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	61.2	4.6
	antmaze-medium-diverse	0.0	0.0	8.0	0.0	0.0	53.7	5.1
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	15.8	3.2
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	14.9	2.3
Adroit	pen-human	34.4	6.3	-1.0	8.1	0.6	37.5	55.8
	hammer-human	1.5	0.5	0.3	0.3	0.2	4.4	2.1
	door-human	0.5	3.9	-0.3	-0.3	-0.3	9.9	9.1
	relocate-human	0.0	0.0	-0.3	-0.3	-0.3	0.20	0.35
	pen-cloned	56.9	23.5	26.5	1.6	-2.5	39.2	40.3
	hammer-cloned	0.8	0.2	0.3	0.3	0.3	2.1	5.7
	door-cloned	-0.1	0.0	-0.1	-0.1	-0.1	0.4	3.5
	relocate-cloned	-0.1	-0.2	-0.3	-0.3	-0.3	-0.1	-0.1
Kitchen	kitchen-complete	33.8	15.0	0.0	0.0	0.0	43.8	31.3
	kitchen-partial	33.8	0.0	13.1	0.0	0.0	49.8	50.1
	kitchen-undirected	47.5	2.5	47.2	0.0	0.0	51.0	52.4



Underestimates the true Q-values

Task Name	CQL(\mathcal{H})	CQL (Eqn. 1)	Ensemble(2)	Ens.(4)	Ens.(10)	Ens.(20)	BEAR
hopper-medium-expert	-43.20	-151.36	3.71e6	2.93e6	0.32e6	24.05e3	65.93
hopper-mixed	-10.93	-22.87	15.00e6	59.93e3	8.92e3	2.47e3	1399.46
hopper-medium	-7.48	-156.70	26.03e12	437.57e6	1.12e12	885e3	4.32

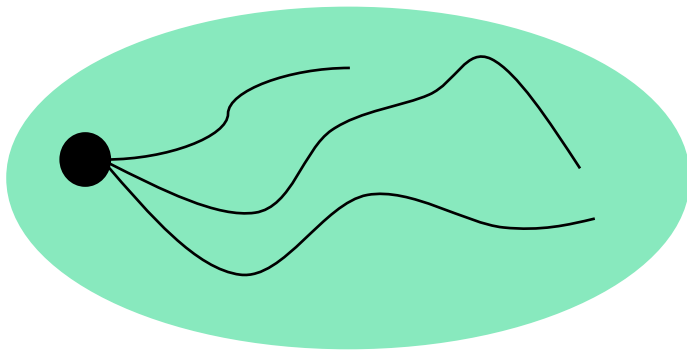
Difference between estimated Q and true Q

Attempt 4: Model-Based Offline RL

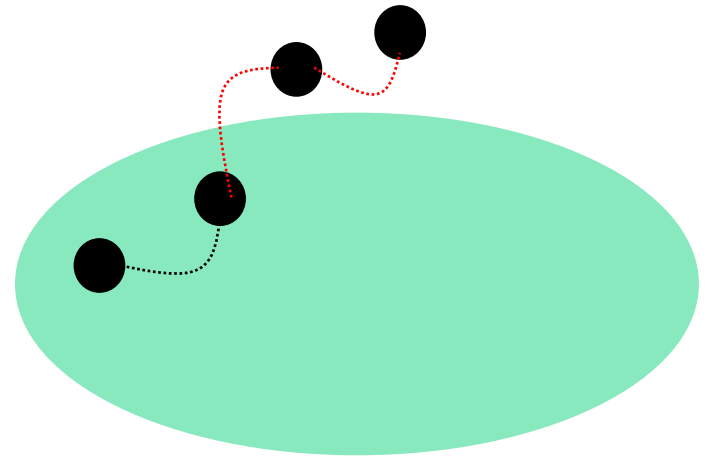
Can we extrapolate past the offline data besides stitching?

Models can generalize past experienced states, actions.

What is the problem with using models for offline RL?



Experienced Data



Erroneously Overoptimistic Planning

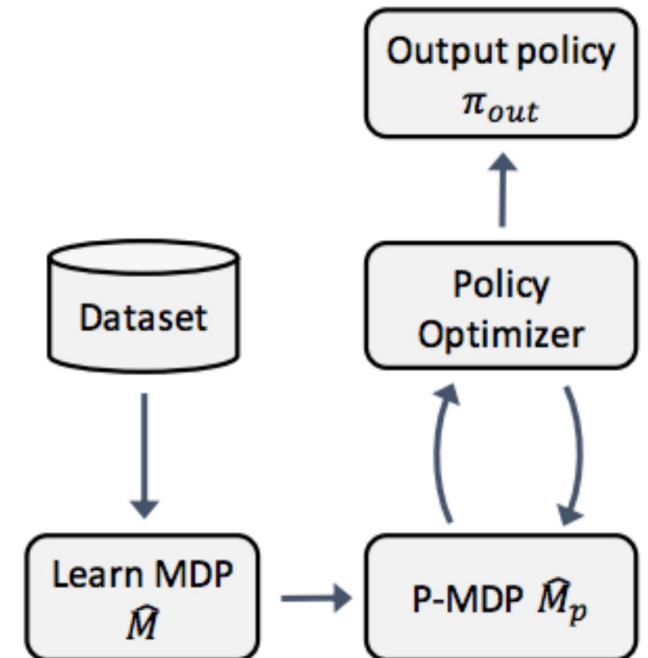
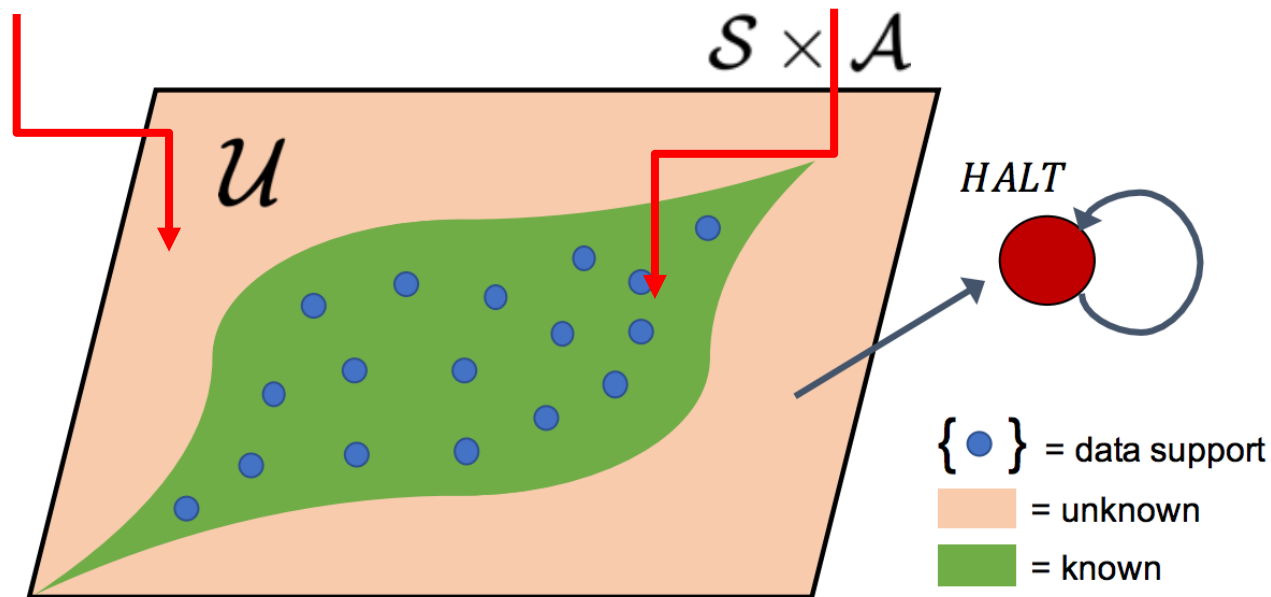
Model-Based Lower Bounding Q-Values

Idea: Down-weight rewards for uncertain model transitions to avoid overoptimism

Modified pessimistic MDP with $r(s, a) = r(s, a) - \lambda u(s, a)$

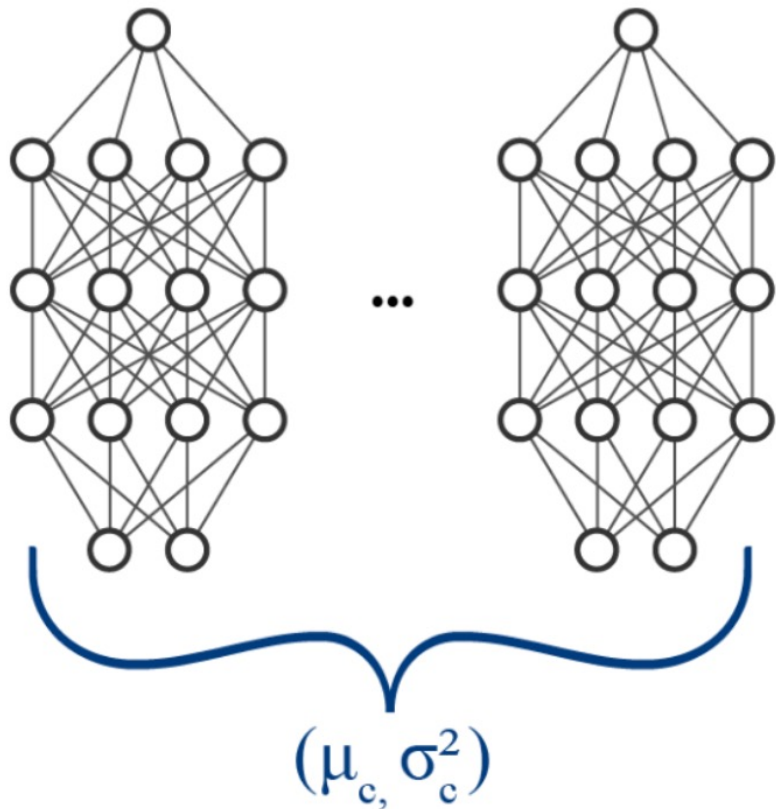
Penalize OOD (s, a) via reward

Leave in distribution r unchanged



Model-Based Lower Bounding Q-Values

How to implement $u(s, a)$?



MOPO (Yu et al):

$$\tilde{r} = r_j - \lambda \max_{i=1}^N \|\Sigma^i(s_j, a_j)\|_F$$

MoREL (Kidambi et al):

$$\text{disc}(s, a) = \max_{i,j} \|f_{\phi_i}(s, a) - \tilde{f}_{\phi_j}(s, a)\|_2$$

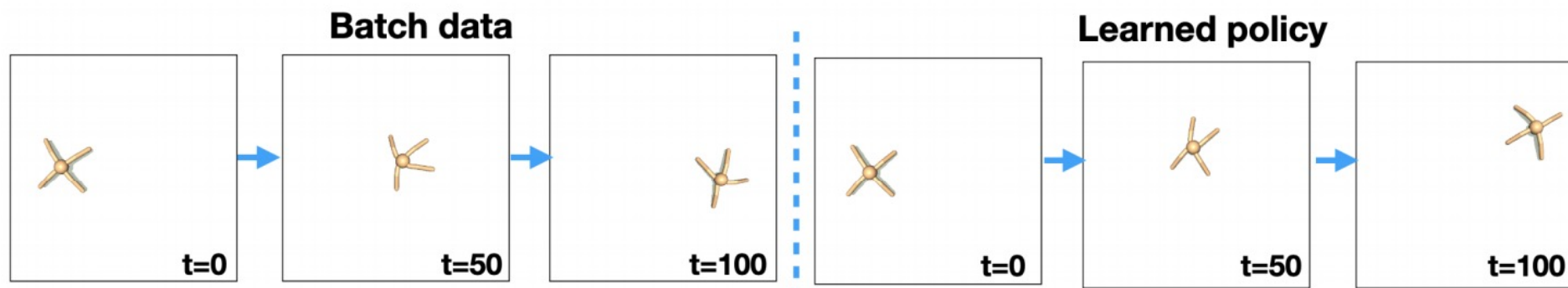
$$\tilde{r}(s, a) = -R_{\max} \quad \text{if } \text{disc}(s, a) > \text{threshold}$$

Model-Based Lower Bounding Q-Values

Works well on random data and mixed data, but less so on medium data

Dataset type	Environment	BC	MOPO (ours)	MBPO	SAC	BEAR	BRAC-v
random	halfcheetah	2.1	31.9 \pm 2.8	30.7 \pm 3.9	30.5	25.5	28.1
random	hopper	1.6	13.3 \pm 1.6	4.5 \pm 6.0	11.3	9.5	12.0
random	walker2d	9.8	13.0 \pm 2.6	8.6 \pm 8.1	4.1	6.7	0.5
medium	halfcheetah	36.1	40.2 \pm 2.7	28.3 \pm 22.7	-4.3	38.6	45.5
medium	hopper	29.0	26.5 \pm 3.7	4.9 \pm 3.3	0.8	47.6	32.3
medium	walker2d	6.6	14.0 \pm 10.1	12.7 \pm 7.6	0.9	33.2	81.3
mixed	halfcheetah	38.4	54.0 \pm 2.6	47.3 \pm 12.6	-2.4	36.2	45.9
mixed	hopper	11.8	92.5 \pm 6.3	49.8 \pm 30.4	1.9	10.8	0.9
mixed	walker2d	11.3	42.7 \pm 8.3	22.2 \pm 12.7	3.5	25.3	0.8
med-expert	halfcheetah	35.8	57.9 \pm 24.8	9.7 \pm 9.5	1.8	51.7	45.3
med-expert	hopper	111.9	51.7 \pm 42.9	56.0 \pm 34.5	1.6	4.0	0.8
med-expert	walker2d	6.4	55.0 \pm 19.1	7.6 \pm 3.7	-0.1	26.0	66.6

Can extrapolate beyond the offline data coverage.



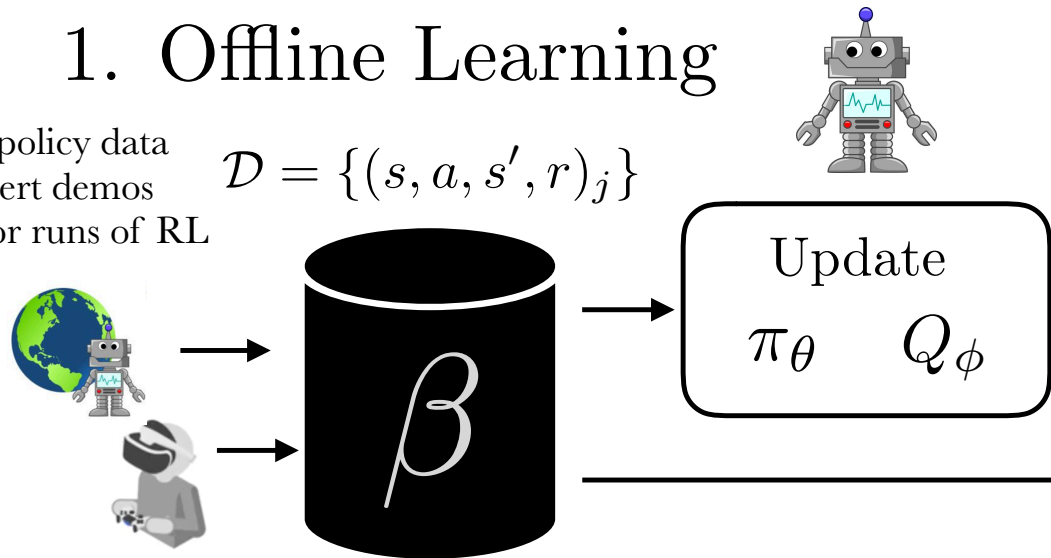
Going from offline RL to online finetuning

In deployment, typically want to pre-train offline, but finetune online!

1. Offline Learning

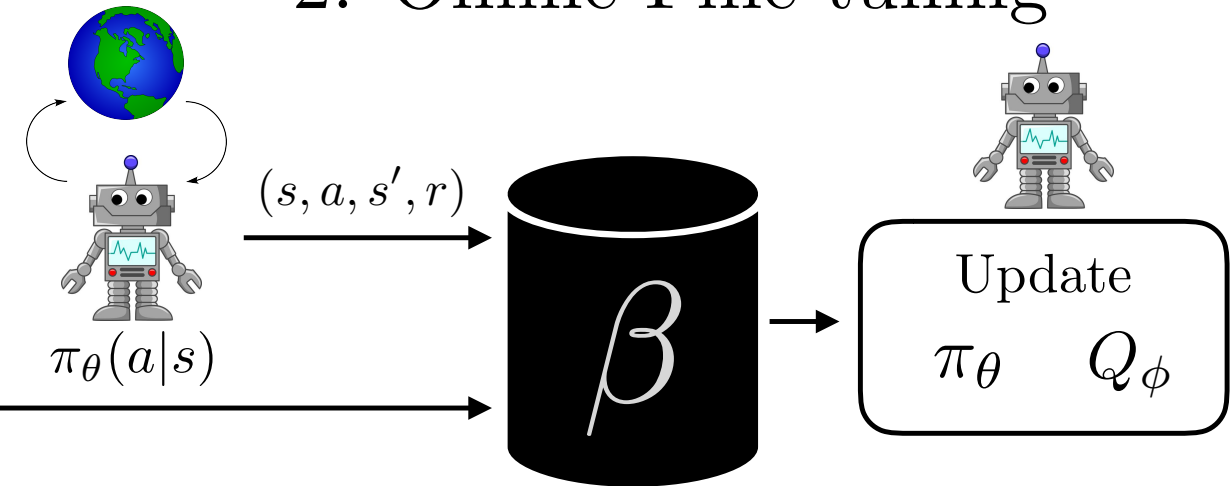
- off-policy data
- expert demos
- prior runs of RL

$$\mathcal{D} = \{(s, a, s', r)_j\}$$



$$p(s'|s, a)$$

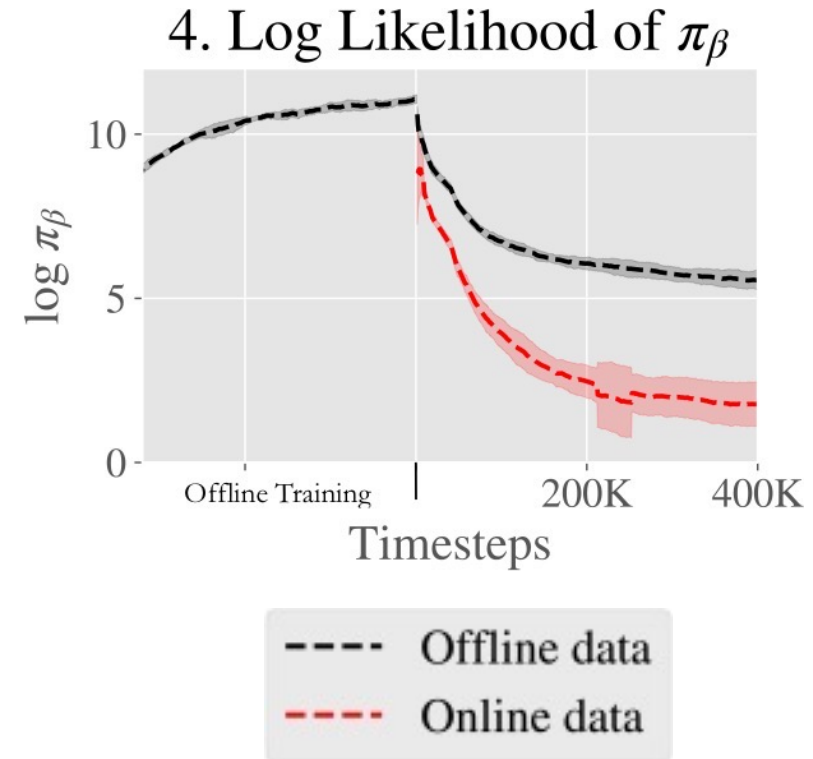
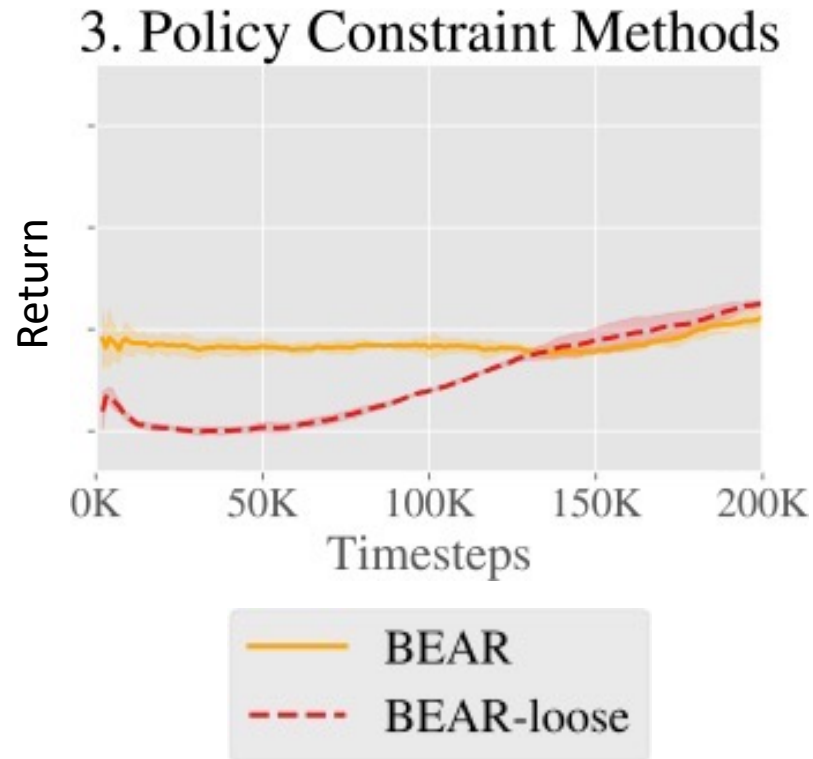
2. Online Fine-tuning



Use large datasets to inform exploration and quick learning.

Why is this challenging?

Typical offline methods can be too conservative, and struggle to improve



Behavior models hard to fit and adapt when finetuning

Advantage Weighted Actor-Critic

Avoid behavior modeling using a policy constraint method, but solve implicitly

$$\pi_{\text{new}} = \max_{\pi \in \Pi} \mathbb{E}_{a_t \sim \pi(\cdot | s_t)} [A^{\pi_k}(s_t, a_t)] \text{ s.t. } D_{KL}(\pi(\cdot | s_t) || \pi_{\beta}(\cdot | s_t)) \leq \epsilon.$$

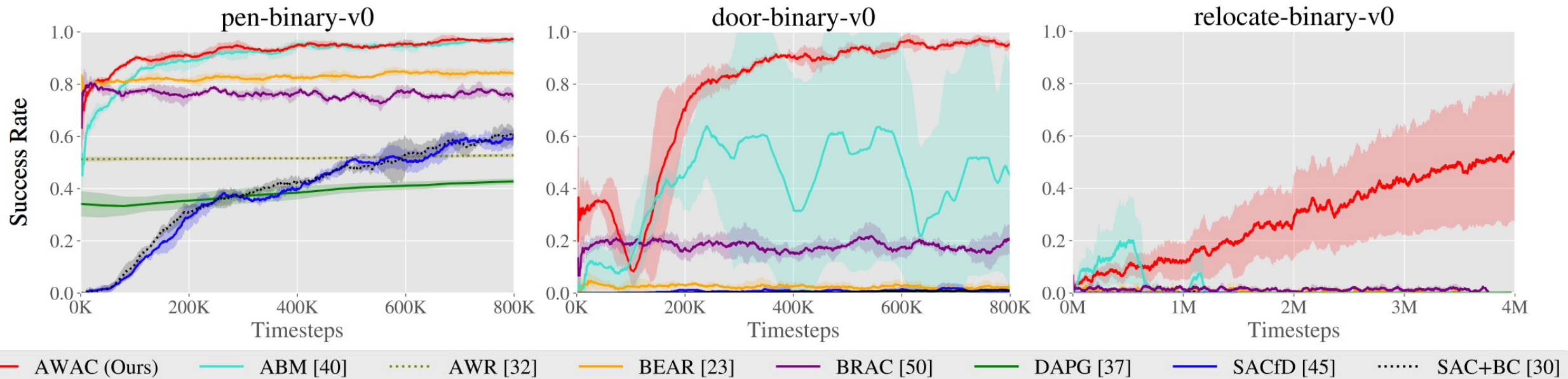
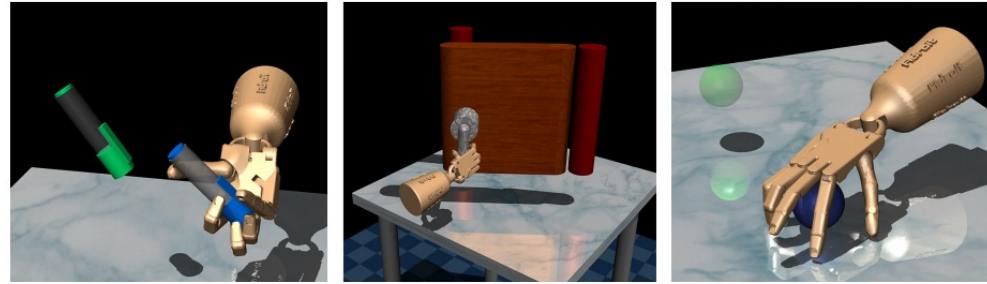
Solving the KKT conditions, projecting via supervised learning

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s_t, a_t \sim \beta} \left[\log \pi_{\theta}(a_t | s_t) \exp \left(\frac{1}{\lambda} A^{\pi_k}(s_t, a_t) \right) \right]$$

↑
Never sample OOD actions, only from buffer

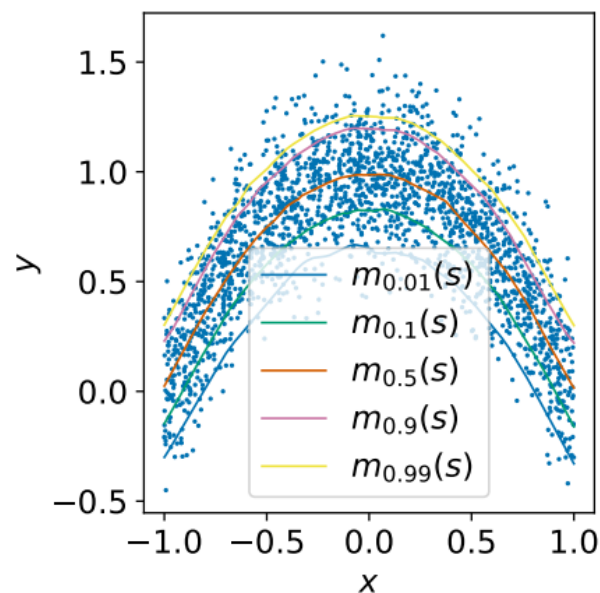
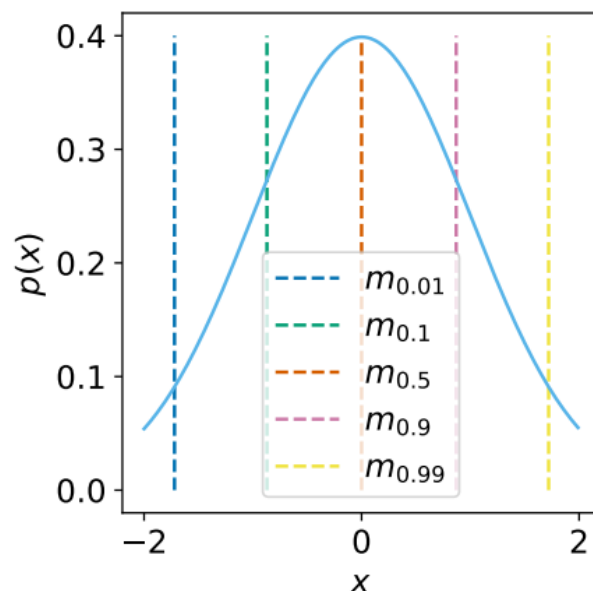
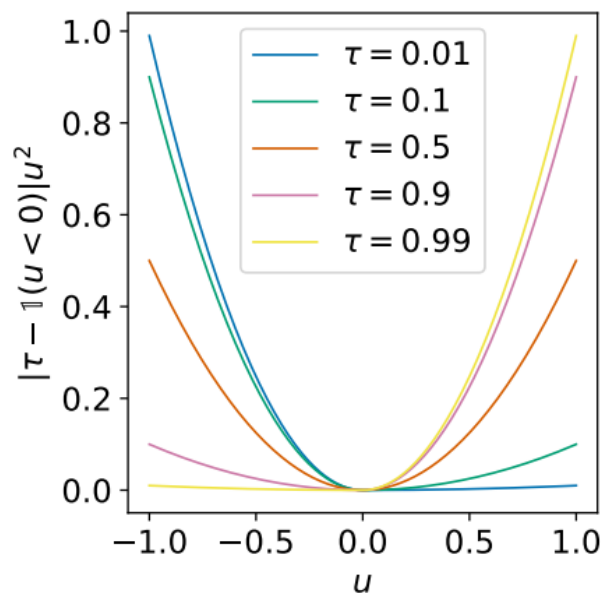
Does this work?

Works reasonably on offline RL, but very effective on finetuning!



Can we get rid of the actor altogether?

Idea: Stay within the data like AWAC, just regress onto the upper expectile



$$\arg \min_{m_\tau} \mathbb{E}_{x \sim X} [L_2^\tau(x - m_\tau)],$$

$$L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$$

Implicit Q-learning

Learn an upper expectile of Q-functions from replay buffer, no bootstrapping!!

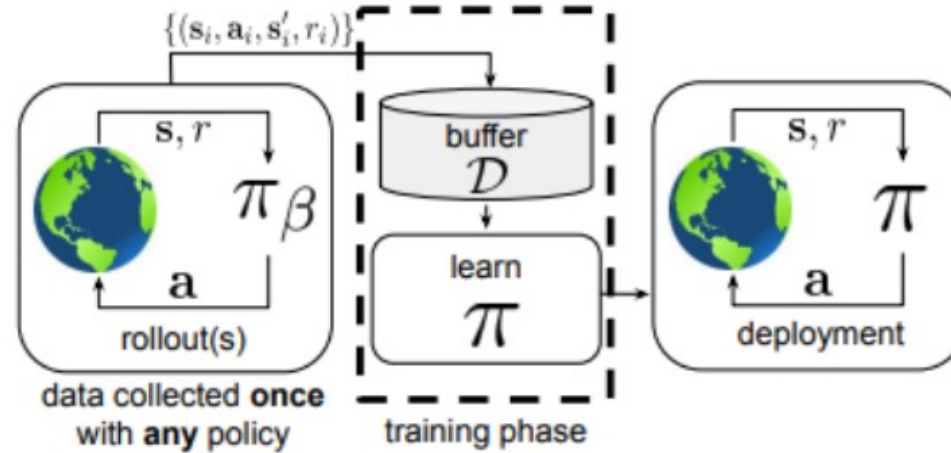
$$L(\theta) = \mathbb{E}_{(s,a,s',a') \sim \mathcal{D}} [L_2^T(r(s,a) + \gamma Q_{\hat{\theta}}(s',a') - Q_{\theta}(s,a))]$$

But why?

Dataset	AWAC	CQL	IQL (Ours)
antmaze-umaze-v0	56.7 → 59.0	70.1 → 99.4	86.7 → 96.0
antmaze-umaze-diverse-v0	49.3 → 49.0	31.1 → 99.4	75.0 → 84.0
antmaze-medium-play-v0	0.0 → 0.0	23.0 → 0.0	72.0 → 95.0
antmaze-medium-diverse-v0	0.7 → 0.3	23.0 → 32.3	68.3 → 92.0
antmaze-large-play-v0	0.0 → 0.0	1.0 → 0.0	25.5 → 46.0
antmaze-large-diverse-v0	1.0 → 0.0	1.0 → 0.0	42.6 → 60.7
antmaze-v0 total	107.7 → 108.3	151.5 → 231.1	370.1 → 473.7
pen-binary-v0	44.6 → 70.3	31.2 → 9.9	37.4 → 60.7
door-binary-v0	1.3 → 30.1	0.2 → 0.0	0.7 → 32.3
relocate-binary-v0	0.8 → 2.7	0.1 → 0.0	0.0 → 31.0
hand-v0 total	46.7 → 103.1	31.5 → 9.9	38.1 → 124.0
total	154.4 → 211.4	182.8 → 241.0	408.2 → 597.7

Landscape of Offline RL methods

offline reinforcement learning



Will cover in readings!

Policy Constraint Methods

- Simple, works well offline
- Too pessimistic

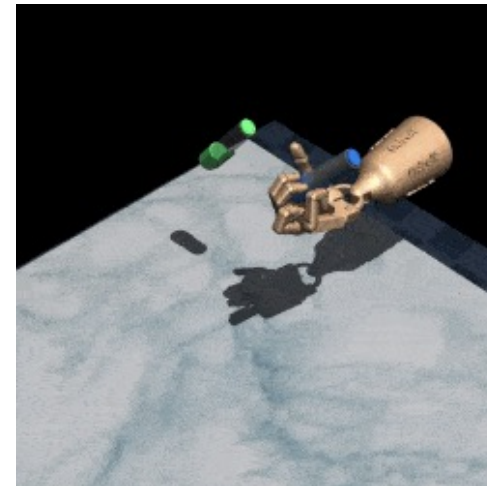
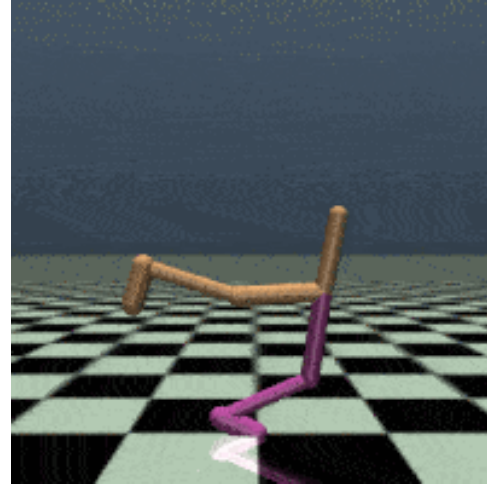
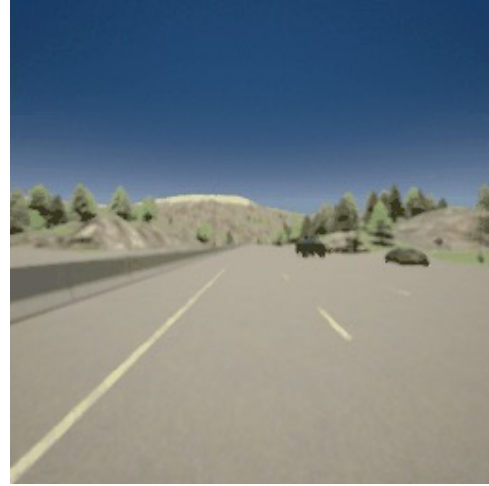
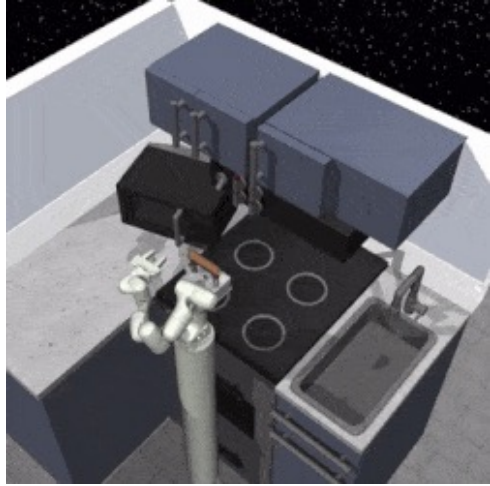
Lower Bounded Q Values

- Less conservative
- More complex min-max optimization

Importance Sampling

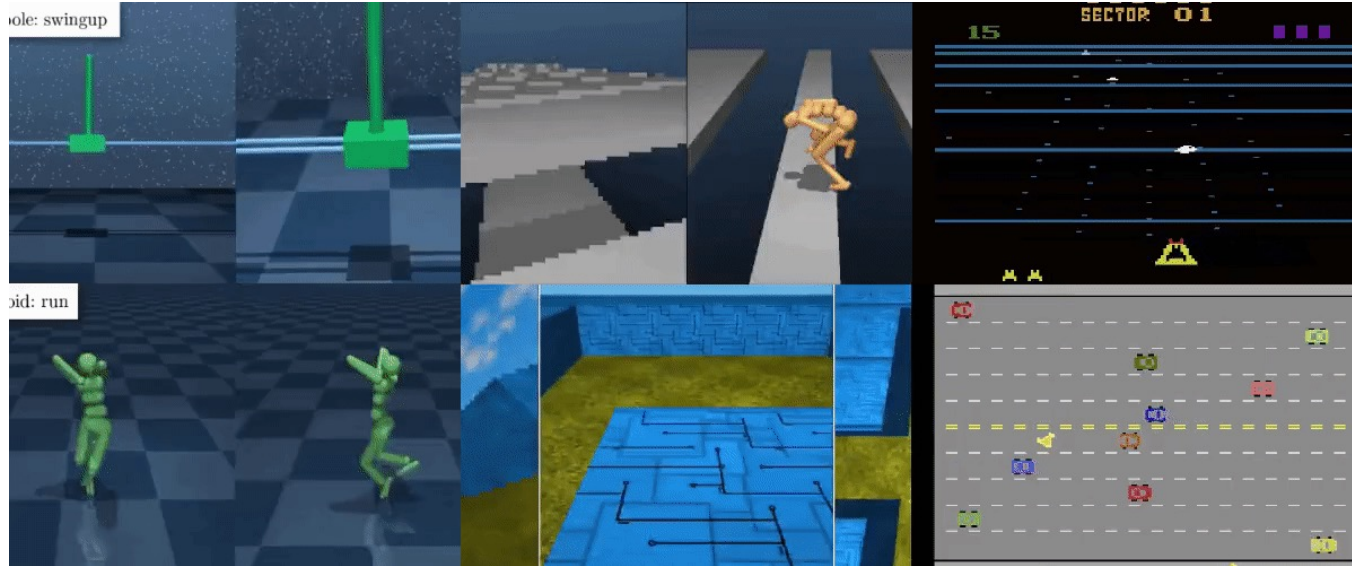
- Unbiased solution
- High variance

D4RL: Benchmarks for Offline RL



- Non-representable behavior policies
- Narrow distributions
- Undirected/multi-task behavior
- Visual perception
- Human demos

RL Unplugged: Benchmarks for Offline RL

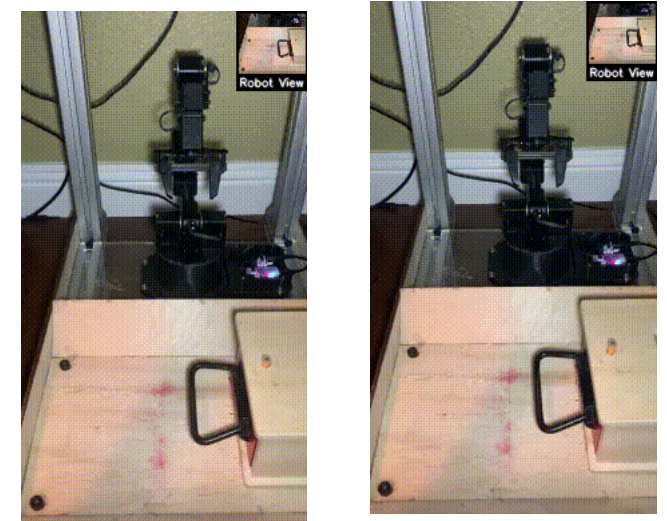
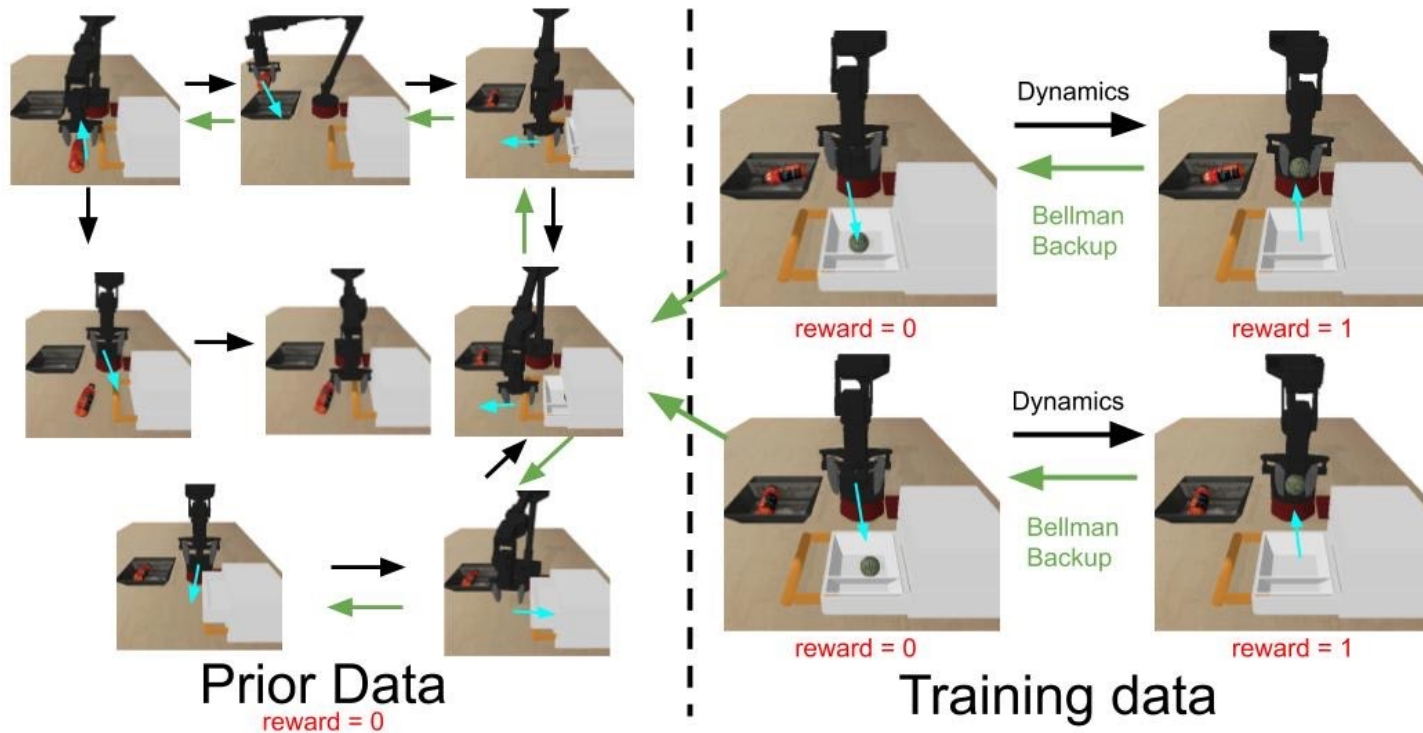


- Partially observable
- Real world RL considerations – latency, noise
- Image based tasks
- Challenging exploration

Task domain	DM Control Suite / Real World RL Suite	DM Locomotion Humanoid	DM Locomotion Rodent	Atari 2600
Action space	continuous	continuous	continuous	discrete
Observation space	state	pixels	pixels	pixels
Exploration difficulty	low to moderate	high	moderate	moderate
Dynamics	deterministic / stochastic	deterministic	deterministic	stochastic

Offline RL on Real Problems

Robotics:



Can stitch together behaviors from prior data

Takeaways from Offline RL

- Offline RL allows us to leverage large offline datasets to improve RL
- Key challenge in offline RL is distribution shift
 - Avoid OOD actions
 - Push down OOD Q/R values
 - Importance sampling
- Can be used to finetune online as well

Lecture Outline

Recap: Multi-task RL formalism



Multi-Task Reinforcement Learning



Meta-Reinforcement Learning



Why offline RL?



Methods for offline RL