Meta Reinforcement Learning
Adaptable Models & Policies

CS 330

Reminders

Homework 3 out, due **Monday 10/26.**

Project milestone due **Monday 11/2.**

Mid-Quarter Check In

How are you doing?

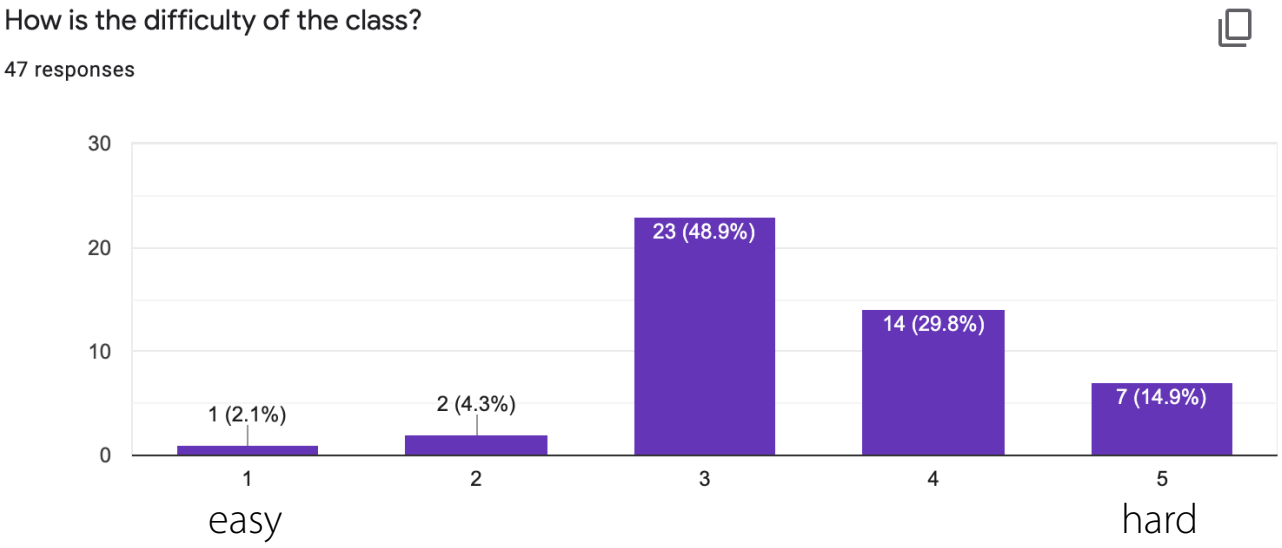
https://pollev.com/chelseafinn494

Mid-Quarter Survey

Thank you!

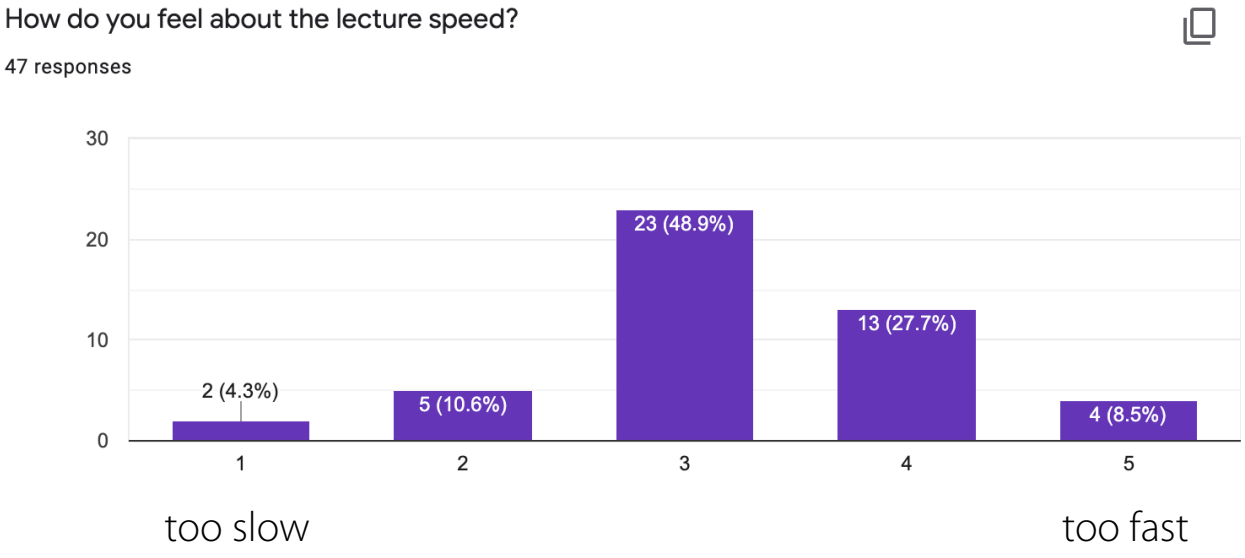
Mid-Quarter Survey

Some of the results



Mid-Quarter Survey

Some of the results



Mid-Quarter Survey

Some of the comments mentioned multiple times.

Lecture feedback:

- Additional sections on background RL, variational inference concepts
- More concrete examples / instantiations of algorithms
- Reduce some gaps between lectures and assignments

Assignment feedback:

- Provide non-colab option
- Cleaner code with more comments
- PyTorch instead of TF
- Reduce # of iterations the algorithms need to be run

Office hours feedback:

- More office hours

Plan for Today

Meta-RL problem statement

Black-box meta-RL methods <- comes up in HW4

Optimization-based meta-RL methods

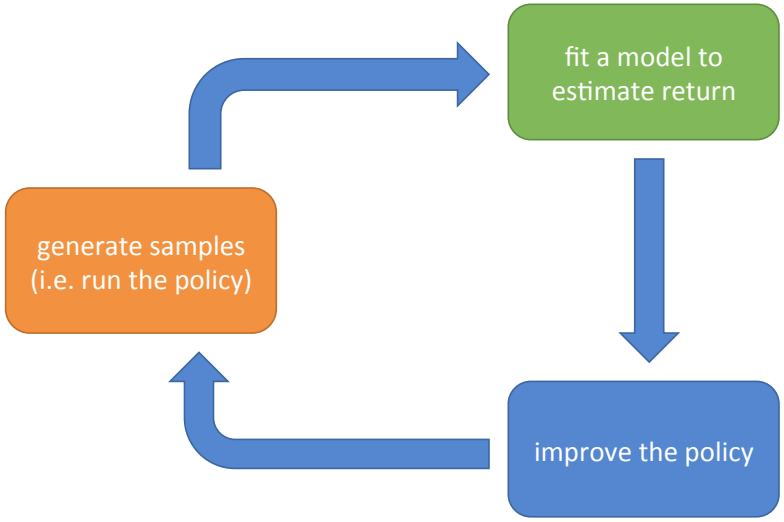
Next time: Learning to explore. <- focus of HW4

Lecture goals:

- Understand the meta-RL problem statement & set-up
- Understand the basics of black-box meta RL algorithms
- Understand the basics & challenges of optimization-based meta RL algorithms

Recap: The anatomy of a reinforcement learning algorithm

Last lectures: introduced model-free, model-based RL methods



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)
estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)
$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)
optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

Recall: Problem Settings

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning

Solve target task $\mathcal{T}_b$ after solving source task $\mathcal{T}_a$ by ***transferring*** knowledge learned from $\mathcal{T}_a$

The Meta-Learning Problem

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, quickly solve new task $\mathcal{T}_{\text{test}}$

In all settings: tasks must share structure.

A reinforcement learning task:

$$\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a}), r_i(\mathbf{s}, \mathbf{a})\}$$

- $\mathcal{S}_i$: state space
- $\mathcal{A}_i$: action space
- $p_i(\mathbf{s}_1)$: initial state distribution
- $p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a})$: dynamics
- $r_i(\mathbf{s}, \mathbf{a})$: reward

Meta-reinforcement learning = meta-learning with RL tasks

Recall: The Meta-Learning Problem

Supervised Learning:

Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$ Data: $\{(\mathbf{x}, \mathbf{y})_i\}$

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Meta Supervised Learning:

Inputs: $\mathcal{D}_{\text{train}}$ $\mathbf{x}_{\text{test}}$ Outputs: $\mathbf{y}_{\text{test}}$ Data: $\{\mathcal{D}_i\}$

$$\{(\mathbf{x}, \mathbf{y})_{1:K}\}$$

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

Why is this view useful?

Reduces the problem to the design & optimization of f.

The Meta Reinforcement Learning Problem

Reinforcement Learning:

Inputs: ~~$\mathbf{x}$~~ $\mathbf{s}_t$ Outputs: ~~$\mathbf{y}$~~ $\mathbf{a}_t$ Data: ~~$\{(\mathbf{x}, \mathbf{y})_i\}$~~

~~$\mathbf{y} = f(\mathbf{x}; \theta)$~~

$\mathbf{a}_t = \pi(\mathbf{s}_t; \theta)$

$\{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})\}$

Meta Reinforcement Learning:

Inputs: $\mathcal{D}_{\text{train}}$ $\mathbf{s}_t$ Outputs: $\mathbf{a}_t$ Data: $\{\mathcal{D}_i\}$

k rollouts from π

$\mathbf{a}_t = f(\mathcal{D}_{\text{train}}, \mathbf{s}_t; \theta)$

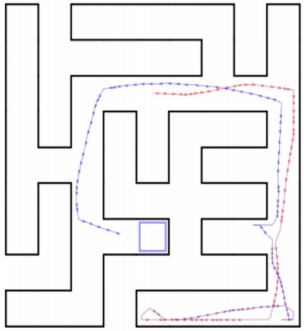
dataset of datasets collected for each task

Design & optimization of *f* *and* collecting appropriate data (learning to explore)

Meta-RL Example: Maze Navigation

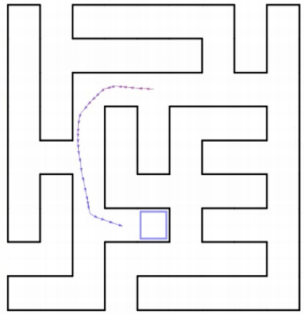
[meta] test time

Given a small amount of experience



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

Learn to solve the task



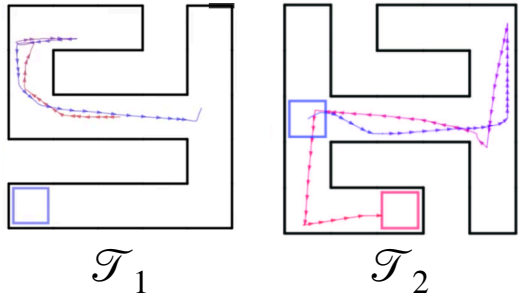
$\mathbf{s}_t \rightarrow \mathbf{a}_t$

diagram adapted from Duan et al. '17

Meta-RL Example: Maze Navigation

[meta] train time

By learning how to learn many other tasks:

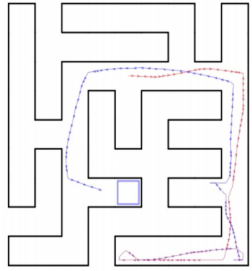


...

meta-training tasks

[meta] test time

Given a small amount of experience



Learn to solve the task

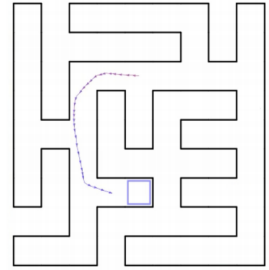
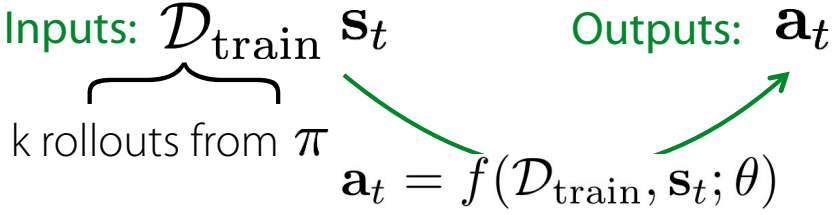


diagram adapted from Duan et al. '17

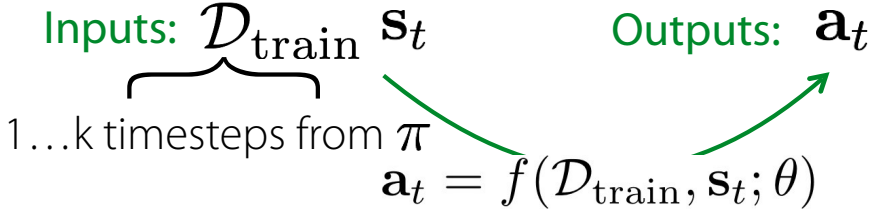
The Meta Reinforcement Learning Problem

Meta Reinforcement Learning:

Episodic Variant



Online Variant



Plan for Today

Meta-RL problem statement

Black-box meta-RL methods

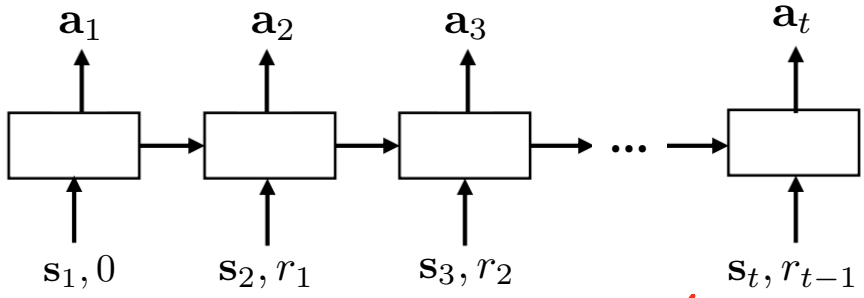
Optimization-based meta-RL methods

Black-box meta-RL

Black-box network
(LSTM, NTM, Conv, …)

$$\mathbf{a}_t = f(\mathcal{D}_{\text{train}}, \mathbf{s}_t; \theta)$$

data has temporal information

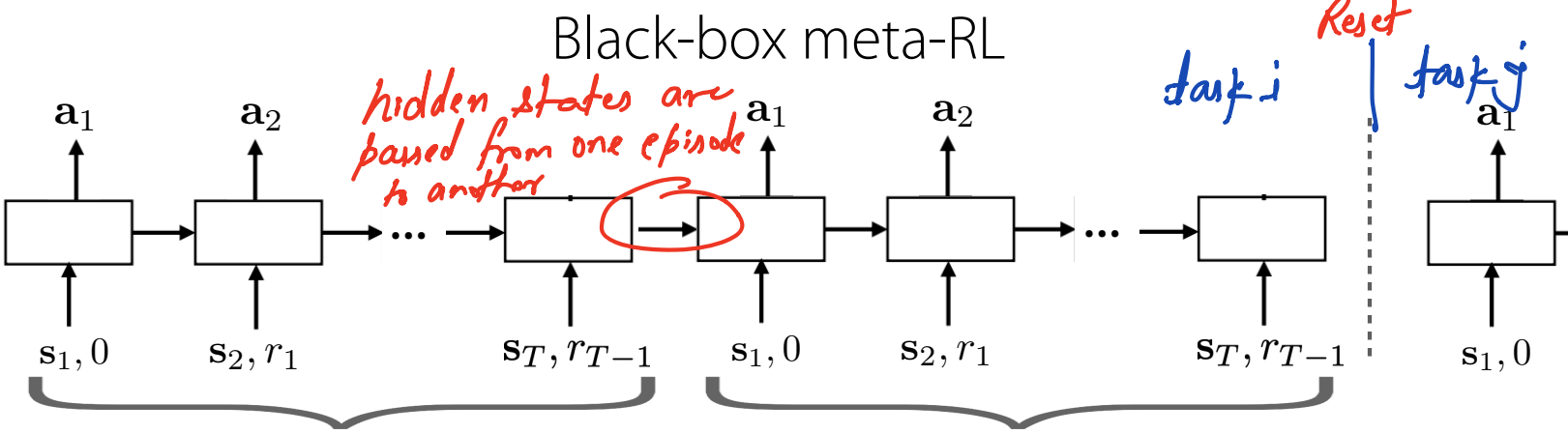


Question: How is this different from simply using a recurrent policy?
(answer in chat or by raising your hand)

Reward is passed as input
(& trained across multiple MDPs)

Hidden state maintained
across episodes within a task!

Black-box meta-RL



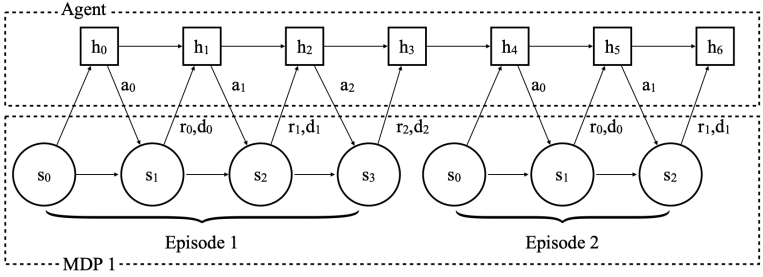
Learn across multiple episodes

1. Sample task $\mathcal{T}_i$
2. Roll-out policy $\pi(a \mid s, \mathcal{D}_{\text{tr}})$ for N episodes (under dynamics $p_i(s' \mid s, a)$ and reward $r_i(s, a)$)
3. Store sequence in replay buffer for task $\mathcal{T}_i$.
4. Update policy to maximize discounted return for all tasks.

keep adding episodes to D_{tr}

RNN architecture

TRPO/A3C (on-policy)



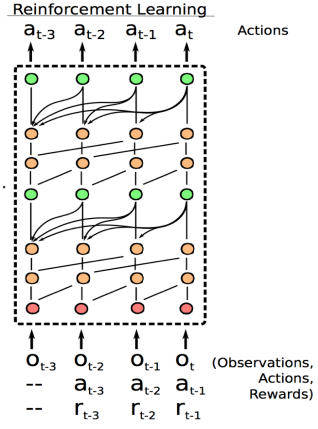
Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. ***RL²: Fast Reinforcement Learning via Slow Reinforcement Learning***. 2017

Wang, Kurth-Nelson, Tirumala, Soyer, Leibo, Munos, Blundell, Kumaran, Botvinick. ***Learning to Reinforcement Learn***. CogSci 2017

Attention + 1D conv

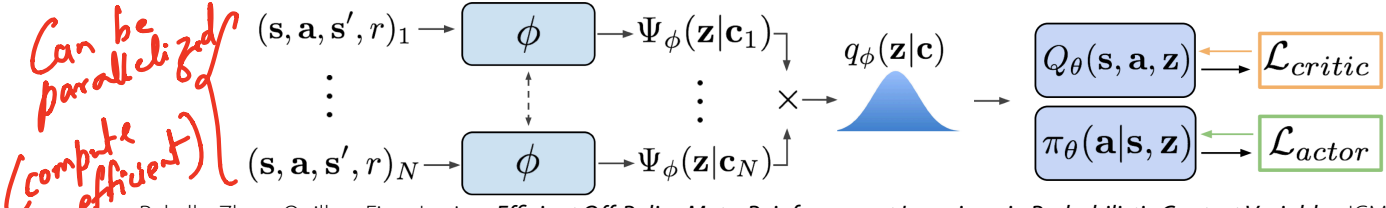
TRPO (on-policy)

Mishra, Rohaninejad, Chen, Abbeel. ***A Simple Neural Attentive Meta-Learner***. ICLR 2018



Feedforward + average

SAC (off-policy)



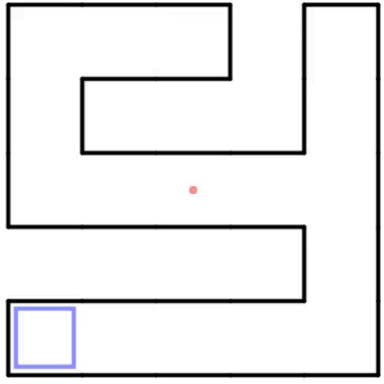
Rakelly, Zhou, Quillen, Finn, Levine. ***Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables***. ICML 2019.

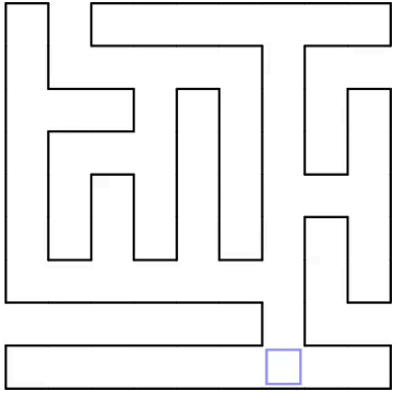
Meta-RL Example #1

From: Mishra, Rohaninejad, Chen, Abbeel. *A Simple Neural Attentive Meta-Learner*. ICLR 2018

Experiment: Learning to visually navigate a maze

- train on 1000 small mazes
- test on held-out small mazes and large mazes



Meta-RL Example #1

From: Mishra, Rohaninejad, Chen, Abbeel. *A Simple Neural Attentive Meta-Learner*. ICLR 2018

Experiment: Learning to visually navigate a maze
- train on 1000 small mazes
- test on held-out small mazes and large mazes

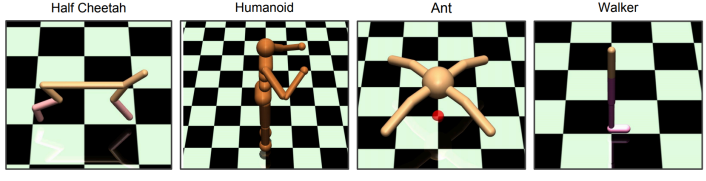
Method	Small Maze		Large Maze	
	Episode 1	Episode 2	Episode 1	Episode 2
Random	188.6 ± 3.5	187.7 ± 3.5	420.2 ± 1.2	420.8 ± 1.2
LSTM	52.4 ± 1.3	39.1 ± 0.9	180.1 ± 6.0	150.6 ± 5.9
SNAIL (ours)	$\mathbf{50.3 \pm 0.3}$	$\mathbf{34.8 \pm 0.2}$	$\mathbf{140.5 \pm 4.2}$	$\mathbf{105.9 \pm 2.4}$

Table 5: Average time to find the goal on each episode

Meta-RL Example #2

Rakelly, Zhou, Quillen, Finn, Levine. *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*. ICML 2019.

Experiment: Continuous control problems

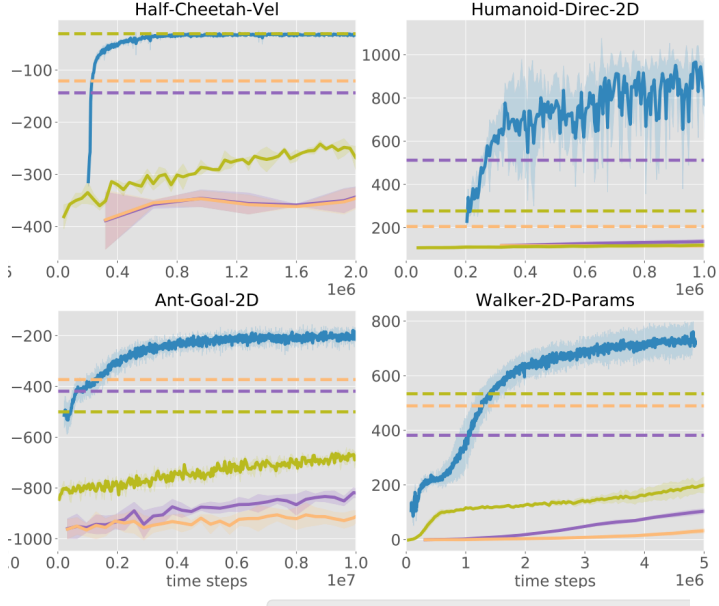


- different directions, velocities
- different physical dynamics

Meta-RL algos are very efficient at new tasks.

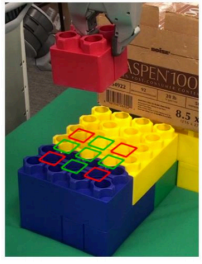
What about **meta-training efficiency**?

Question: Do you expect off-policy meta-RL to be more or less efficient than on-policy meta-RL?

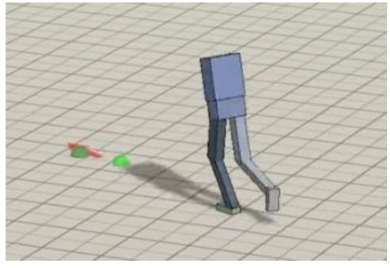


Digression: Connection to Multi-Task Policies

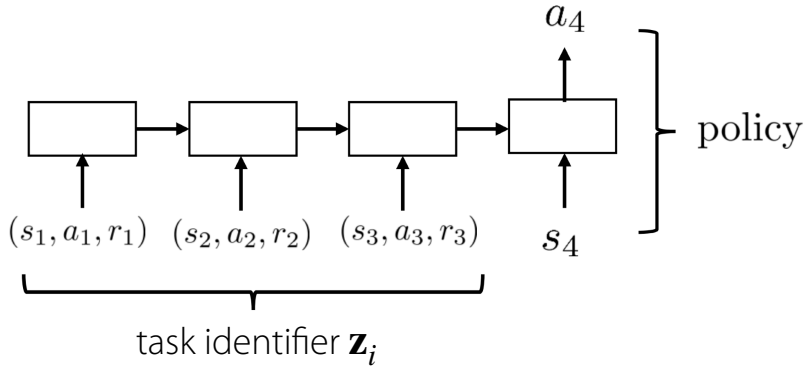
multi-task policy: $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z}_i)$



$\mathbf{z}_i$: stack location



$\mathbf{z}_i$: walking direction



Multi-task policy with experience as task identifier.

What about **goal-conditioned policies** / **value functions**?

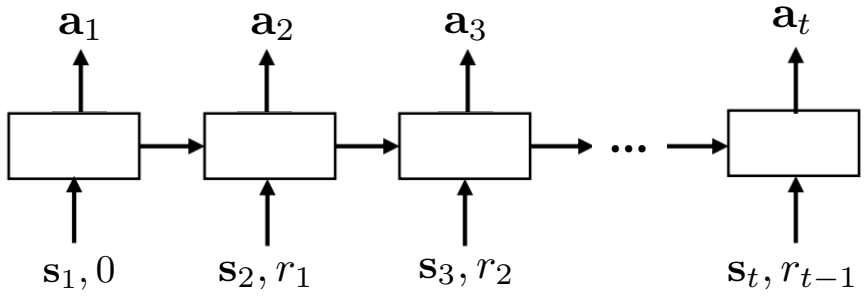
- rewards are a strict generalization of goals
- meta-RL objective is to ***adapt*** new tasks vs. ***generalize*** to new goals

(**k-shot** vs. **0-shot**)

Black-box meta-RL

Black-box network
(LSTM, NTM, Conv, …)

$$\mathbf{a}_t = f(\mathcal{D}_{\text{train}}, \mathbf{s}_t; \theta)$$



\+ general & expressive
\+ a variety of design choices in architecture
\- hard to optimize
~ inherits sample efficiency from outer RL optimizer

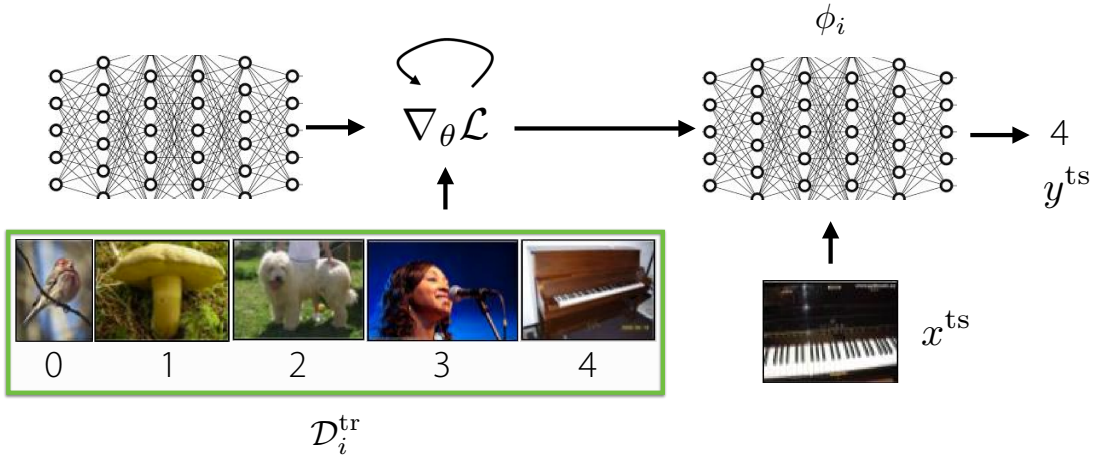
Plan for Today

Meta-RL problem statement

Black-box meta-RL methods

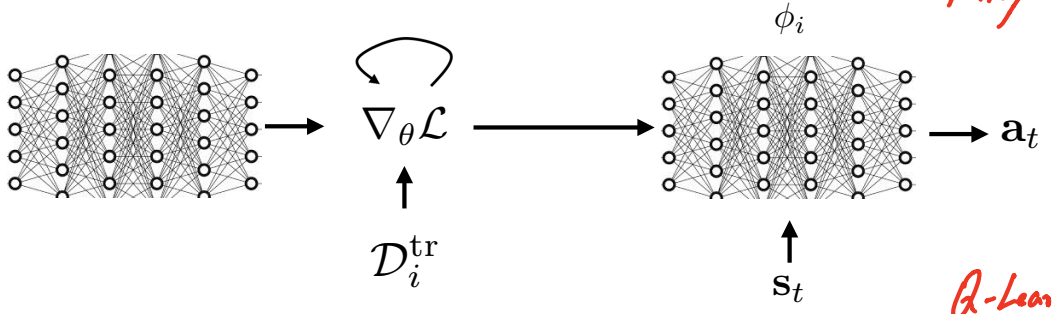
Optimization-based meta-RL methods

Recap: Optimization-Based Meta-Learning



Key idea: embed optimization inside the inner learning process

Recap: Optimization-Based ~~Meta-Learning~~ Meta-RL



k rollouts from π_θ

Key idea: embed optimization inside the inner learning process

Policy Gradient:

$$\sum_t \nabla_\theta \log \pi \sum_t r_t$$

doesn't work well with sparse rewards.

Q-Learning:

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

works one timestep at a time

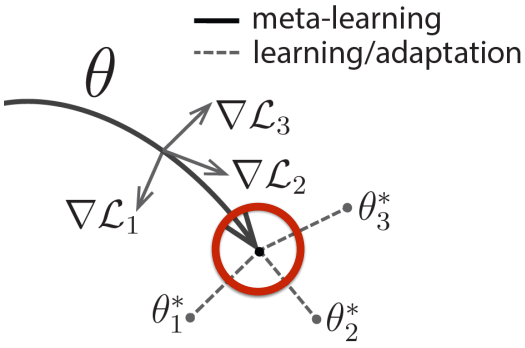
Question: What should we use for the inner optimization and why? (in chat or by raising hand)

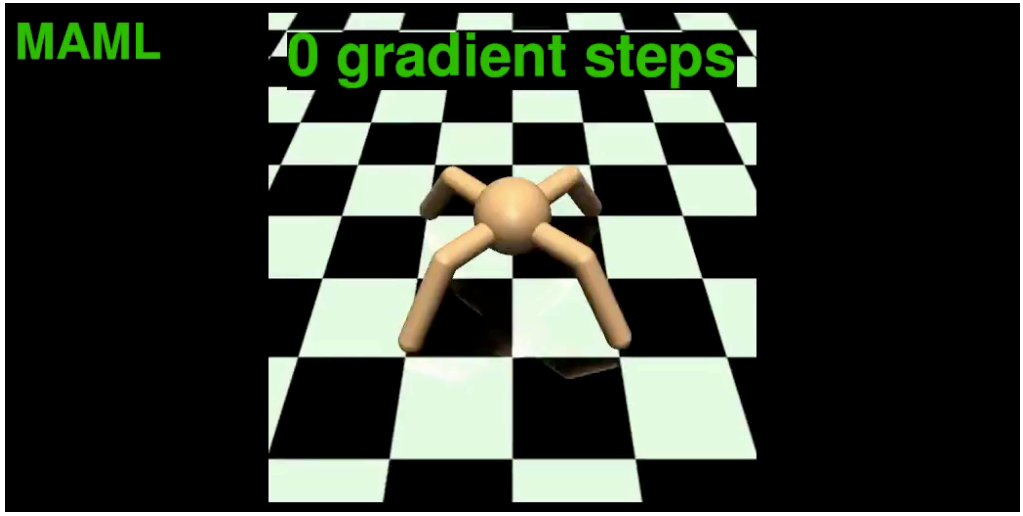
Policy gradients?	Q-learning?	Model-based RL?
+ gradient-based!	- dynamic programming (requires many steps)	+ gradient-based (model learning=supervised)
- on-policy (inefficient)	+ off-policy (efficient)	+ off-policy (efficient)

Eric Mitchell et al. MACAL

Kind of MAML

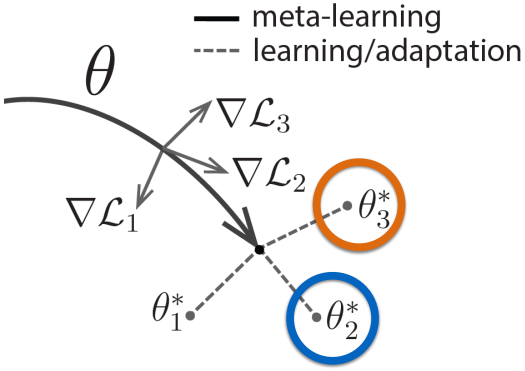
MAML + Policy Gradients

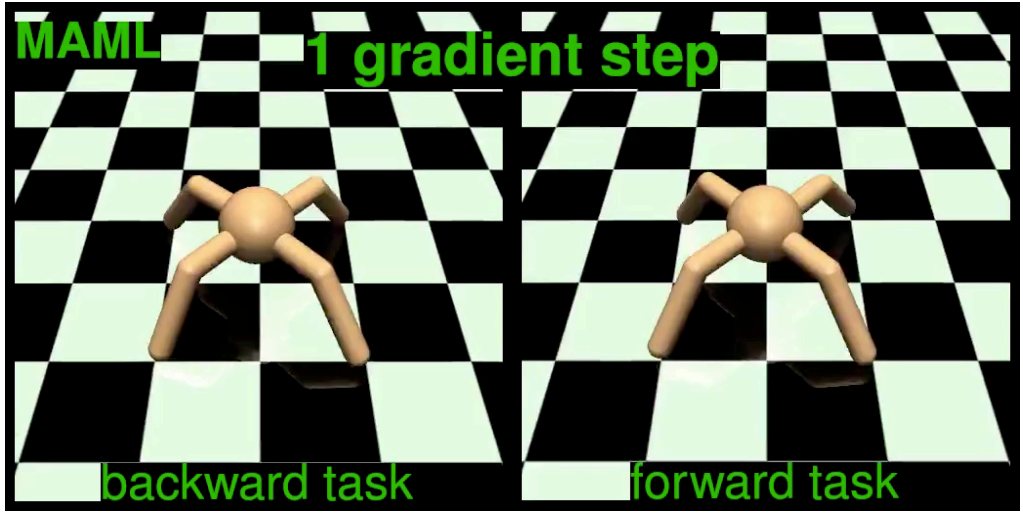




Finn, Abbeel, Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. ICML 2017

MAML + Policy Gradients





Finn, Abbeel, Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. ICML 2017

MAML + Model-Based RL

Online Variant

Inputs: $\mathcal{D}_{\text{train}}$ $\mathbf{s}_t$ Outputs: $\mathbf{a}_t$

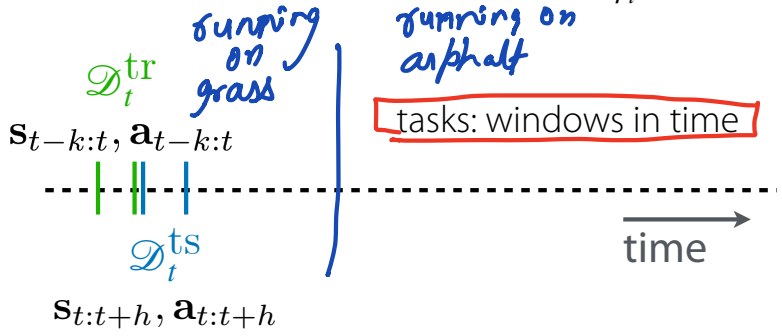
k timesteps from π

Meta-test time:

1. Adapt model $f_\theta \rightarrow f_{\phi_t}$ to last k time steps
2. Plan $a_t, \ldots, a_{t+h}$ using adapted model f_{ϕ_t}

$\| f_\theta(s,a) - s' \|^2$

Meta-training:

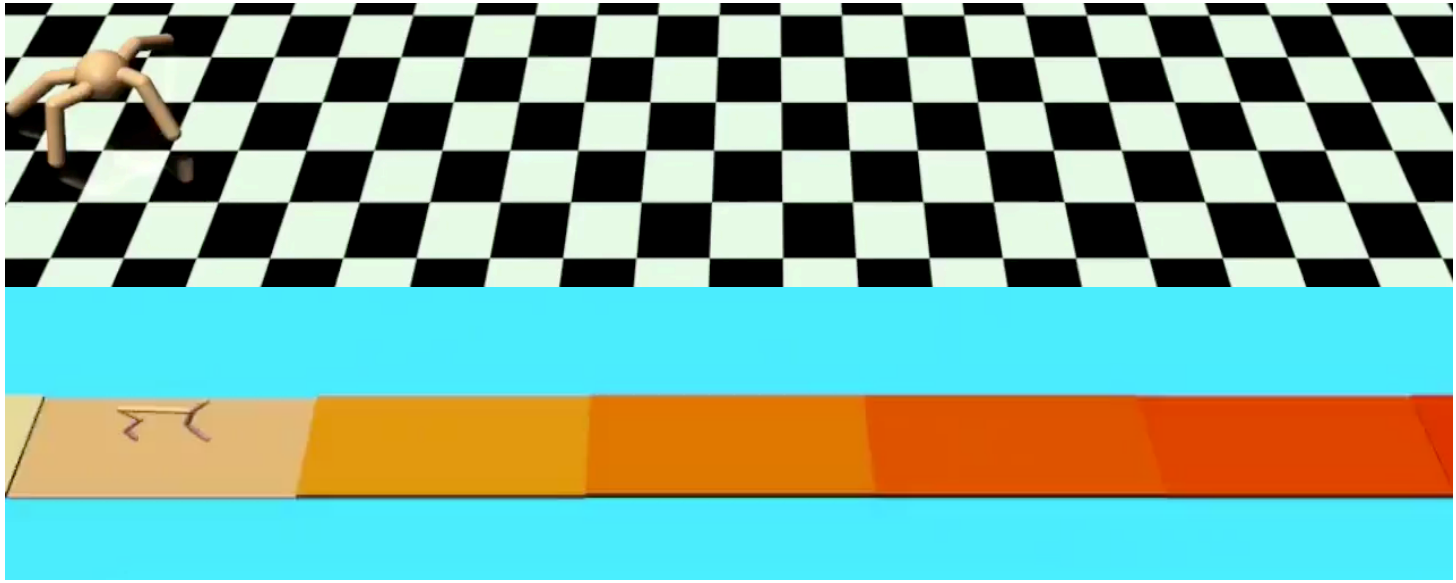


Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR '19

Dynamic Environments without Adaptation

Model-Based RL Only

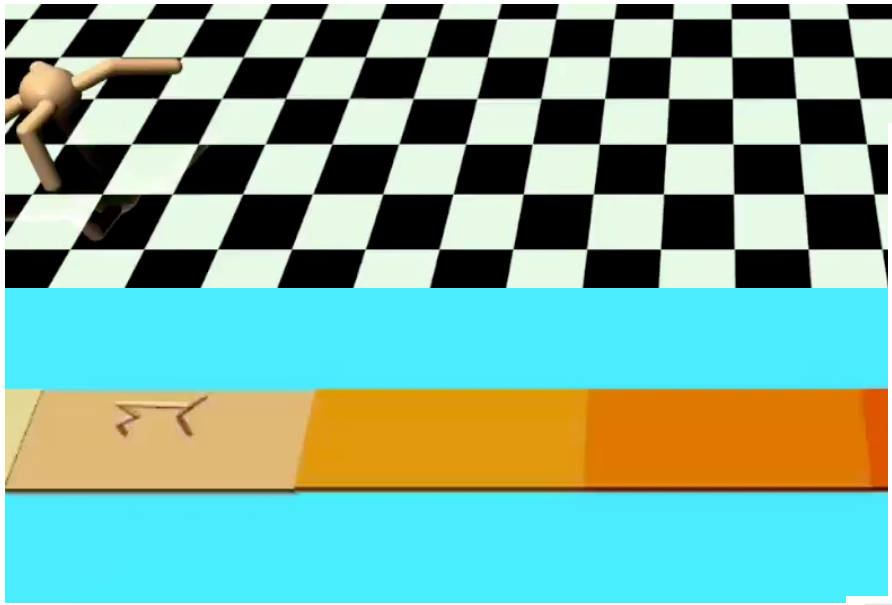
Tries to fit single model $f(s' \mid s, a)$ to varying $p_t(s' \mid s, a)$.

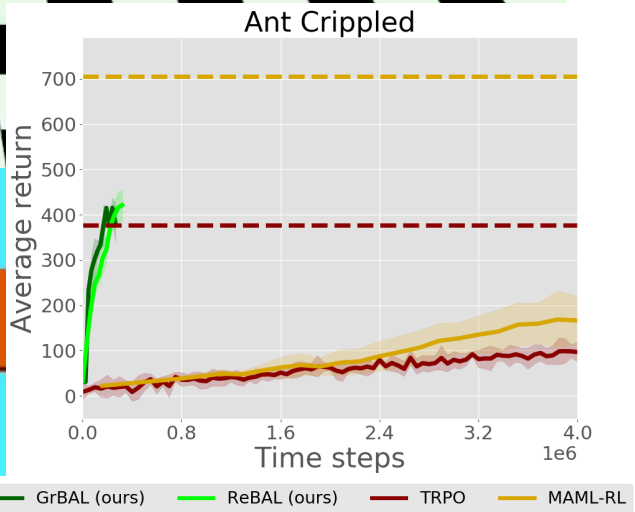


Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. *Learning to Adapt in Dynamic Real-World Environments*. ICLR '19

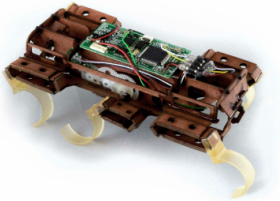
Dynamic Environments without Adaptation

MAML+Model-based RL





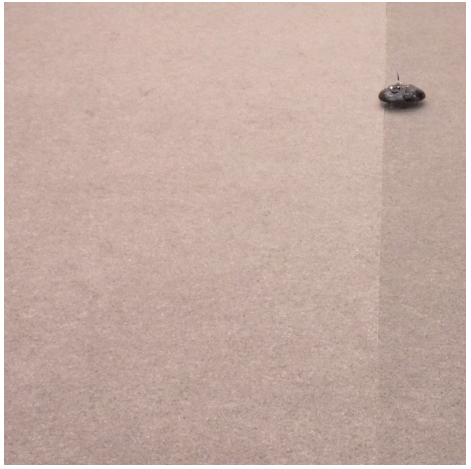
Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. *Learning to Adapt in Dynamic Real-World Environments*. ICLR'19



VelociRoACH Robot

Meta-train on **variable terrains**





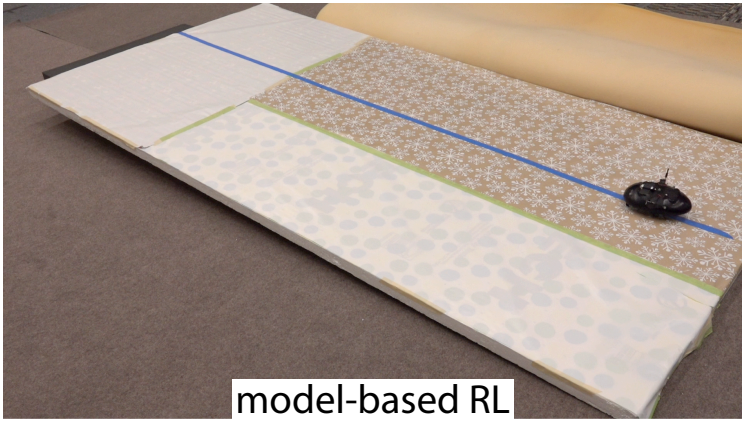
Meta-test with **slope, missing leg, payload, calibration errors**

Model being adapted at test time

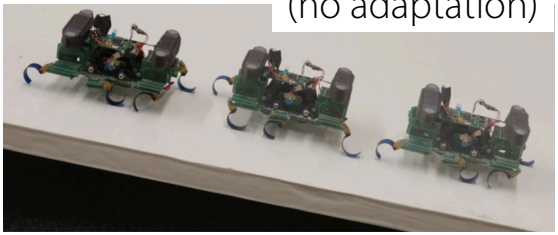
Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. *Learning to Adapt in Dynamic Real-World Environments*. ICLR '19

VelociRoACH Robot

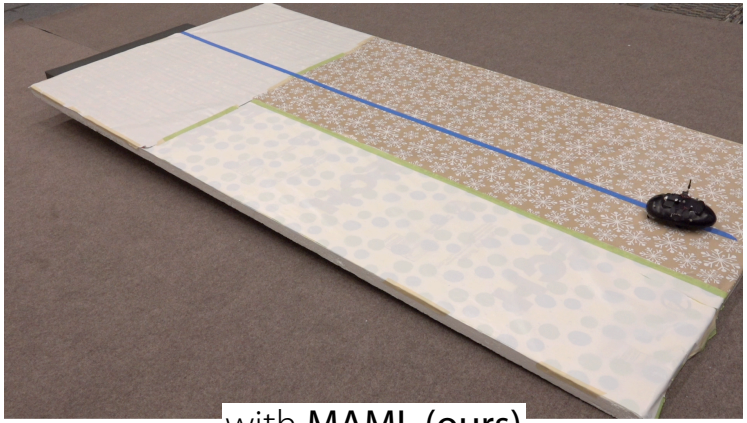
Meta-train on **variable terrains**



model-based RL
(no adaptation)



Meta-test with **slope**, missing leg, payload, calibration errors

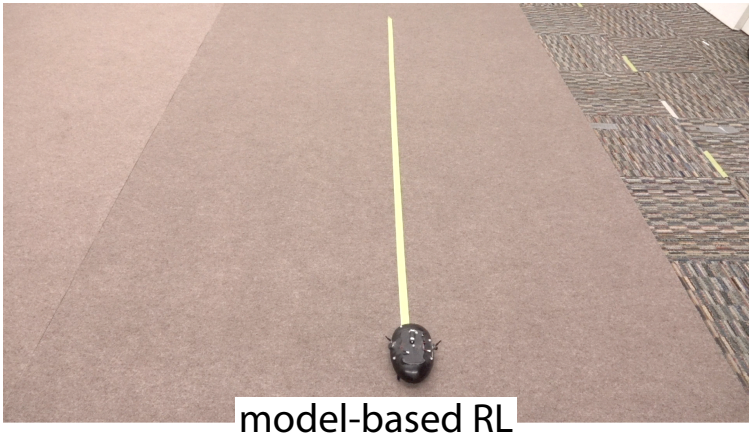


with **MAML (ours)**

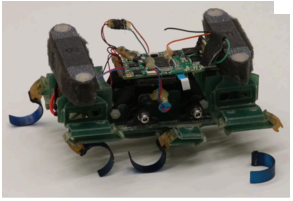
Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. *Learning to Adapt in Dynamic Real-World Environments*. ICLR '19

VelociRoACH Robot

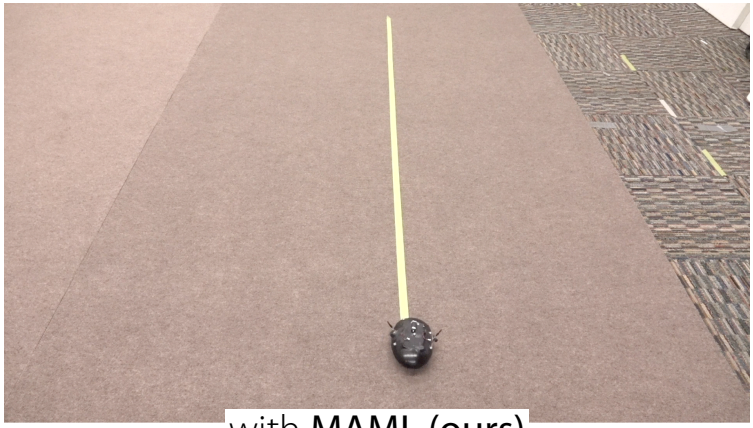
Meta-train on **variable terrains**



model-based RL
(no adaptation)



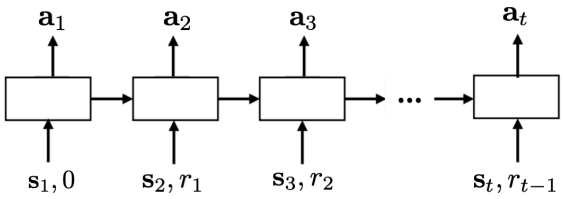
Meta-test with **slope, missing leg, payload, calibration errors**



with **MAML (ours)**

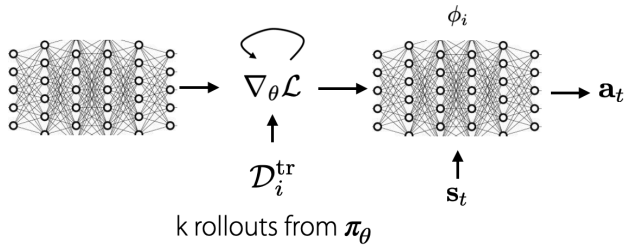
Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. *Learning to Adapt in Dynamic Real-World Environments*. ICLR '19

Black-Box Meta-RL



+ general & expressive
+ a variety of design choices in architecture & objective
-- hard to optimize

Optimization-Based Meta-RL



+ inductive bias of optimization built in
+ easy to combine with policy gradients, model-based methods
-- policy gradients very noisy
-- hard to combine with value-based RL methods

Both: inherit sample efficiency from outer RL optimizer

Plan for Today

Meta-RL problem statement

Black-box meta-RL methods

Optimization-based meta-RL methods

Next time: Learning to explore.

Lecture goals:

- Understand the meta-RL problem statement & set-up
- Understand the basics of black-box meta RL algorithms
- Understand the basics & challenges of optimization-based meta RL algorithms

Next time

Today: meta-RL basics

Next Monday: learning to explore via meta-RL

Next Wednesday: Bayesian perspective on meta-RL

Reminders

Homework 3 due **Monday 10/26.**

Project milestone due **Monday 11/2.**