Meta-Learning & Transfer Learning

CS 285

Instructor: Sergey Levine
UC Berkeley

What's the problem?

this is easy (mostly)

Why?

this is impossible

Montezuma’s revenge

- Getting key = reward
- Opening door = reward
- Getting killed by skull = bad

Montezuma’s revenge

- We know what to do because we **understand** what these sprites mean!
- Key: we know it opens doors!
- Ladders: we know we can climb them!
- Skull: we don’t know what it does, but we know it can’t be good!
- **Prior understanding of problem structure can help us solve complex tasks quickly!**

Can RL use the same prior knowledge as us?

- If we've solved prior tasks, we might acquire useful knowledge for solving a new task
- How is the knowledge stored?
  - Q-function: tells us which actions or states are good
  - Policy: tells us which actions are potentially useful
    - some actions are never useful!
  - Models: what are the laws of physics that govern the world?
  - Features/hidden states: provide us with a good representation
    - Don't underestimate this!

Transfer learning terminology

transfer learning: using experience from one set of tasks for faster learning and better performance on a new task

in RL, task = MDP!

source domain

target domain

“shot”: number of attempts in the target domain

0-shot: just run a policy trained in the source domain

1-shot: try the task once

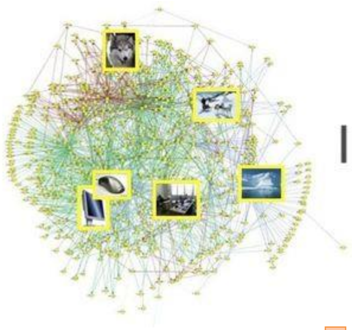
few shot: try the task a few times

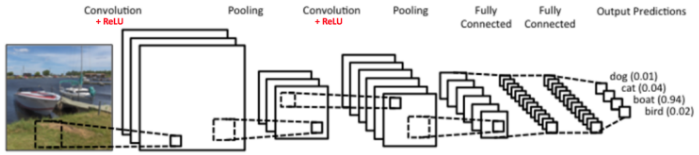
How can we frame transfer learning problems?

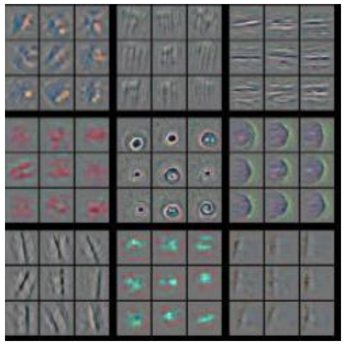
1. Forward transfer: learn policies that transfer effectively
   a) Train on source task, then run on target task (or finetune)
   b) Relies on the tasks being quite similar!
2. Multi-task transfer: train on many tasks, transfer to a new task
   a) Sharing representations and layers across tasks in multi-task learning
   b) New task needs to be similar to the *distribution* of training tasks
3. Meta-learning: learn to *learn* on many tasks
   a) Accounts for the fact that we'll be adapting to a new task during training!

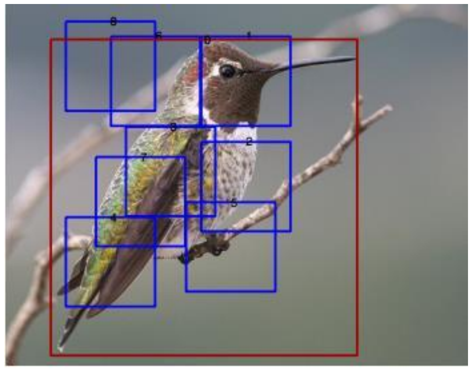
Pretraining + Finetuning

The most popular transfer learning method in (supervised) deep learning!









Sim 2 Real

What issues are we likely to face?


- **Domain shift:** representations learned in the source domain might not work well in the target domain


- **Difference in the MDP:** some things that are possible to do in the source domain are not possible to do in the target domain

- **Finetuning issues:** if pretraining & finetuning, the finetuning process may still need to explore, but optimal policy during finetuning may be deterministic!

Domain adaptation in computer vision

train here

z

correct answer

reversed gradient

(same network)

can we force this layer to be **invariant** to domain?

$D_\phi(z) = \text{domain}$ domain classifier: guess domain from *z*

z

incorrect answer

do well here

Is this true?

Invariance assumption: everything that is **different** between domains is **irrelevant**

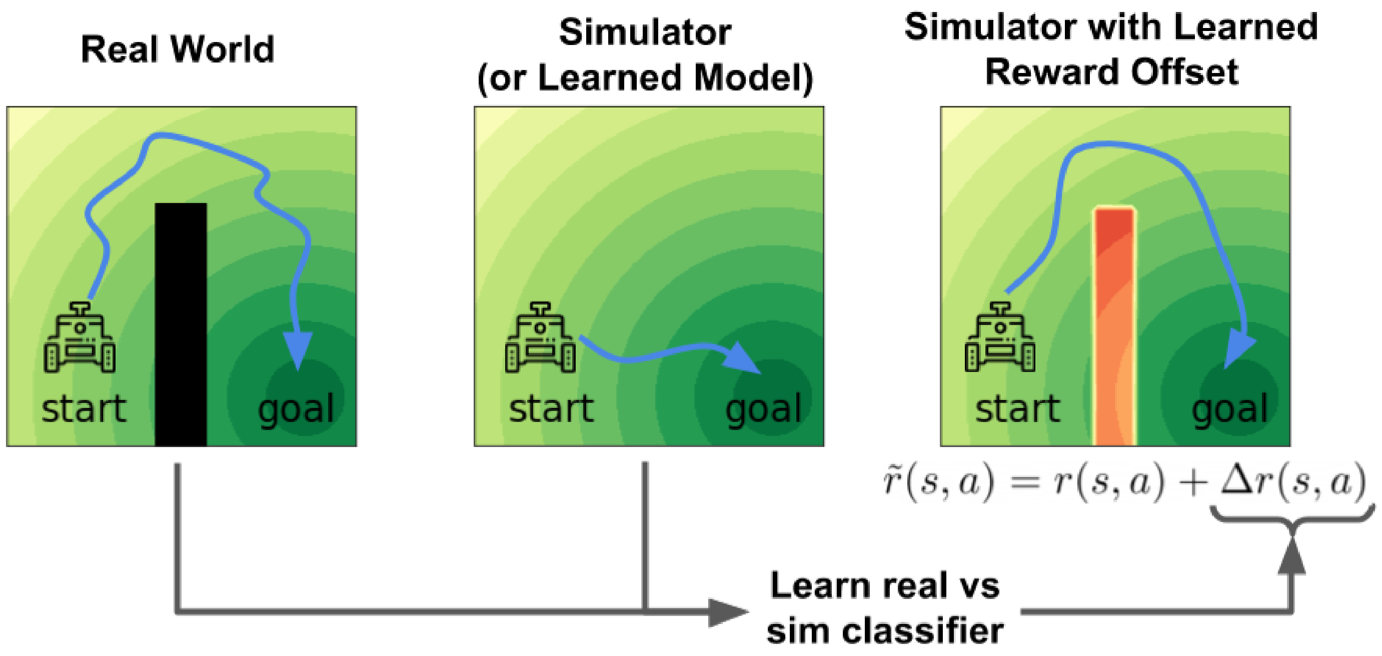
formally:

$p(x)$ is different exists some $z = f(x)$ such that $p(y|z) = p(y|x)$, but $p(z)$ is same

Domain adaptation in RL for dynamics?

s_t
a_t
s_{t+1}
D_1
~~Source~~ target

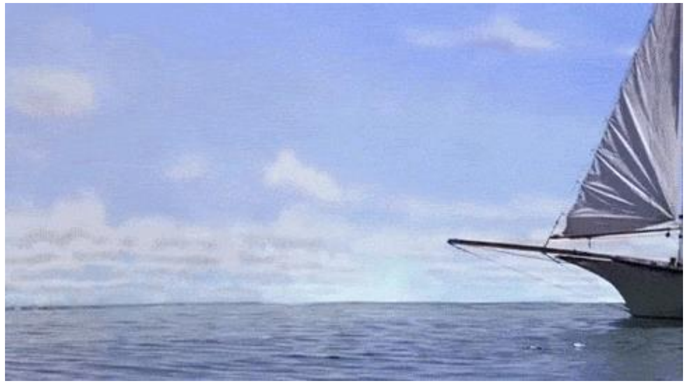
Why is **invariance** not enough when the dynamics don't match?



$$\tilde{r}(s, a) = r(s, a) + \Delta r(s, a)$$

$$\Delta r(s_t, a_t, s_{t+1}) = \log p_{\text{target}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{source}}(s_{t+1} \mid s_t, a_t).$$

$$\Delta r(s_t, a_t, s_{t+1}) = \log p(\text{target} \mid s_t, a_t, s_{t+1}) - \log p(\text{target} \mid s_t, a_t) - \log p(\text{source} \mid s_t, a_t, s_{t+1}) + \log p(\text{source} \mid s_t, a_t)$$



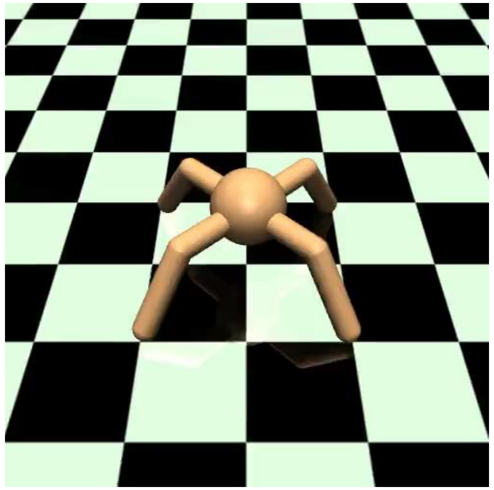
s_t
a_t
D_2
~~Source~~ targ

When might this **not** work?

Eysenbach et al., "Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers"

What if we can also finetune?

1. RL tasks are generally much less diverse
   - Features are less general
   - Policies & value functions become overly specialized
2. Optimal policies in fully observed MDPs are deterministic
   - Loss of exploration at convergence
   - Low-entropy policies adapt very slowly to new settings

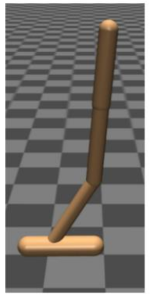


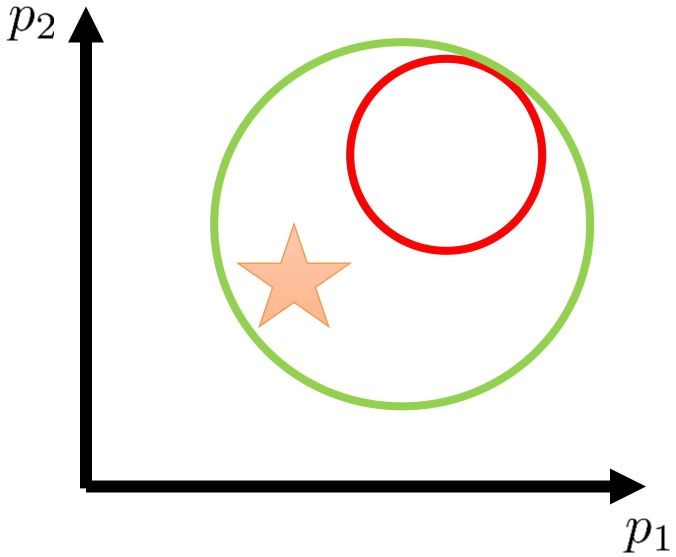
See “exploration 2” lecture on unsupervised skill discovery and “control as inference” lecture on MaxEnt RL methods!

How to maximize **forward** transfer?

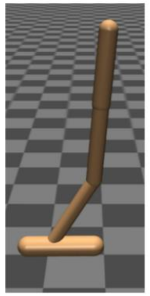
Basic intuition: the more **varied** the training domain is, the more likely we are to generalize in **zero shot** to a slightly different domain.

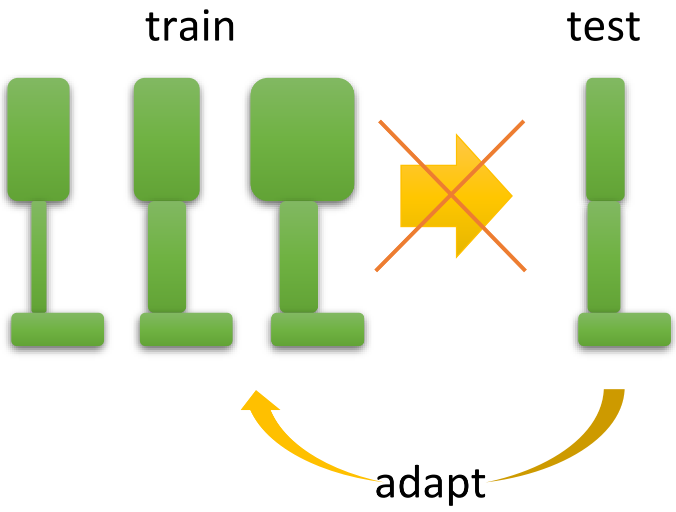
“Randomization” (dynamics/appearance/etc.): widely used for simulation to real world transfer (e.g., in robotics)



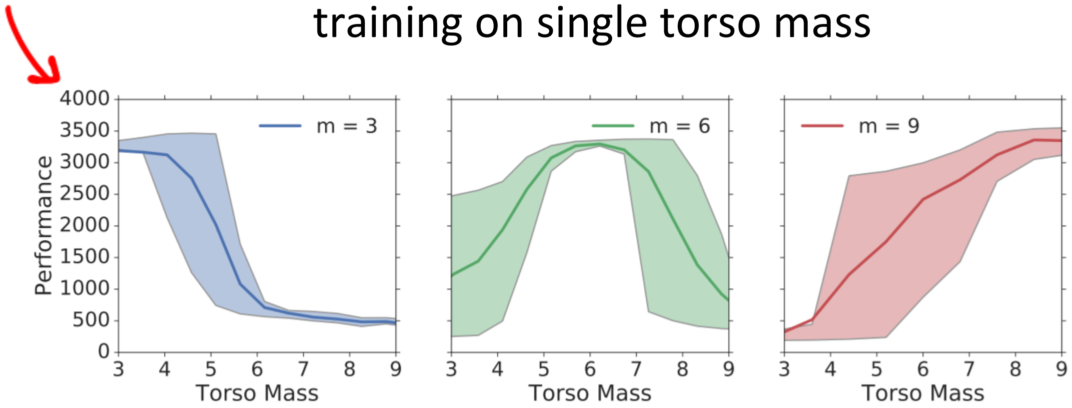


EPOpt: randomizing physical parameters

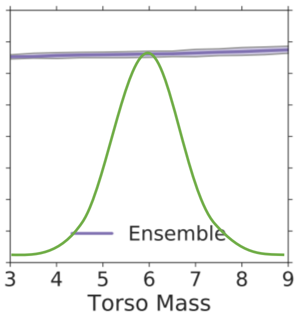




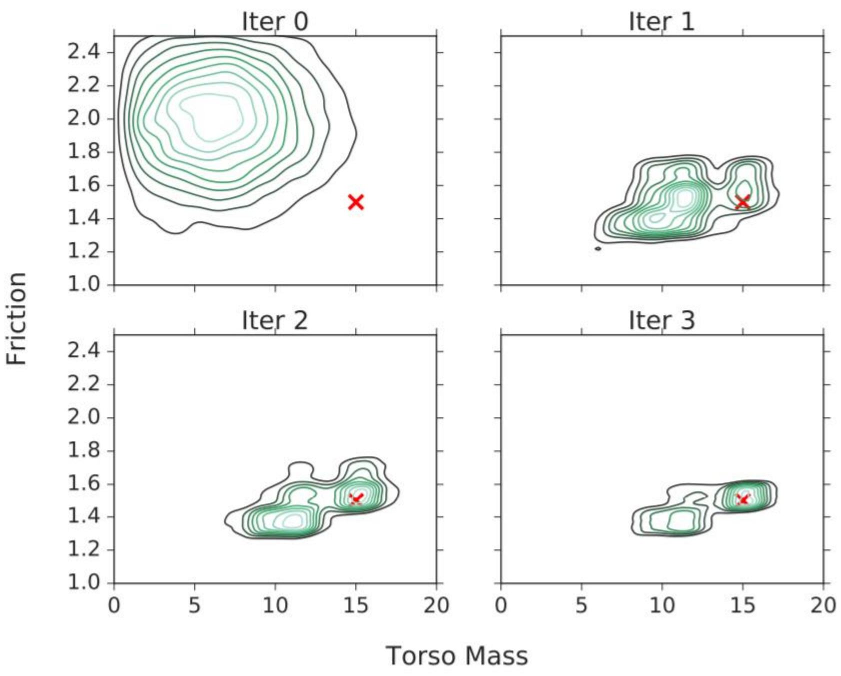
training on single torso mass



training on model ensemble

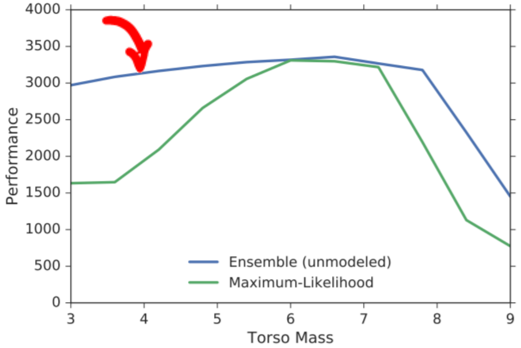


ensemble adaptation



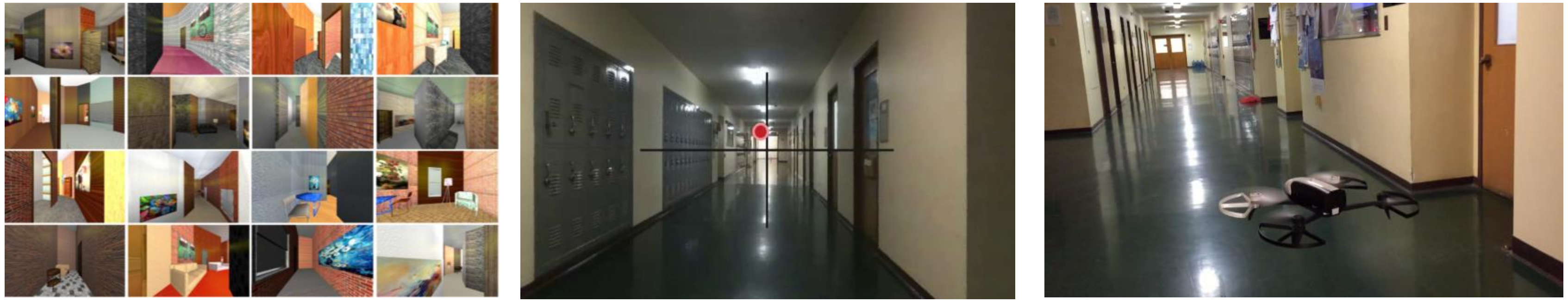
unmodeled effects

Hopper	μ	σ	low	high
~~mass~~	~~6.0~~	~~1.5~~	~~3.0~~	~~9.0~~
ground friction	2.0	0.25	1.5	2.5
joint damping	2.5	1.0	1.0	4.0
armature	1.0	0.25	0.5	1.5
Half-Cheetah	μ	σ	low	high
~~mass~~	~~6.0~~	~~1.5~~	~~3.0~~	~~9.0~~
ground friction	0.5	0.1	0.3	0.7
joint damping	1.5	0.5	0.5	2.5
armature	0.125	0.04	0.05	0.2

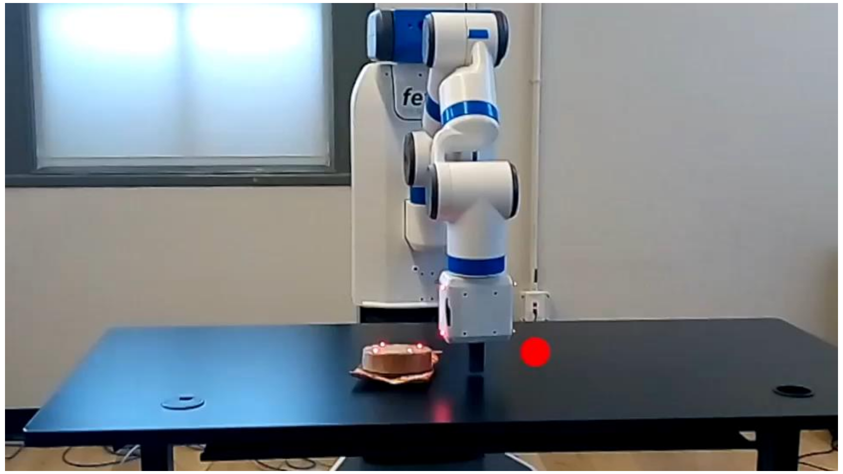


Rajeswaran et al., “EPOpt: Learning robust neural network policies...”

More randomization!



Sadeghi et al., “CAD2RL: Real Single-Image Flight without a Single Real Image.” 2016



Xue Bin Peng et al., “Sim-to-Real Transfer of Robotic Control with Dynamics Randomization.” 2018

Lee et al., “Learning Quadrupedal Locomotion over Challenging Terrain.” 2020

Some suggested readings

Domain adaptation:

Tzeng, Hoffman, Zhang, Saenko, Darrell. **Deep Domain Confusion: Maximizing for Domain Invariance**. 2014.

Ganin, Ustinova, Ajakan, Germain, Larochelle, Laviolette, Marchand, Lempitsky. **Domain-Adversarial Training of Neural Networks**. 2015.

Tzeng*, Devin*, et al., **Adapting Visuomotor Representations with Weak Pairwise Constraints**. 2016.

Eysenbach et al., **Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers**. 2020.

Finetuning:

Finetuning via MaxEnt RL: Haarnoja*, Tang*, et al. (2017). **Reinforcement Learning with Deep Energy-Based Policies.**

Andreas et al. **Modular multitask reinforcement learning with policy sketches.** 2017.

Florensa et al. **Stochastic neural networks for hierarchical reinforcement learning.** 2017.

Kumar et al. **One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL.** 2020

Simulation to real world transfer:

Rajeswaran, et al. (2017). **EPOpt: Learning Robust Neural Network Policies Using Model Ensembles.**

Yu et al. (2017). **Preparing for the Unknown: Learning a Universal Policy with Online System Identification.**

Sadeghi & Levine. (2017). **CAD2RL: Real Single Image Flight without a Single Real Image.**

Tobin et al. (2017). **Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.**

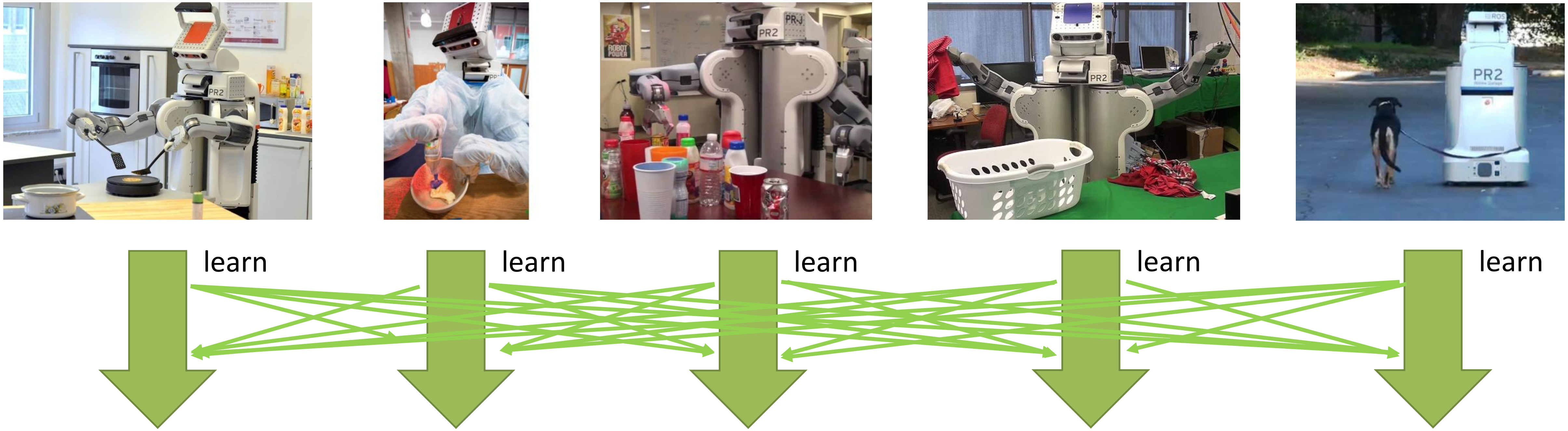
Tan et al. (2018). **Sim-to-Real: Learning Agile Locomotion For Quadruped Robots.**

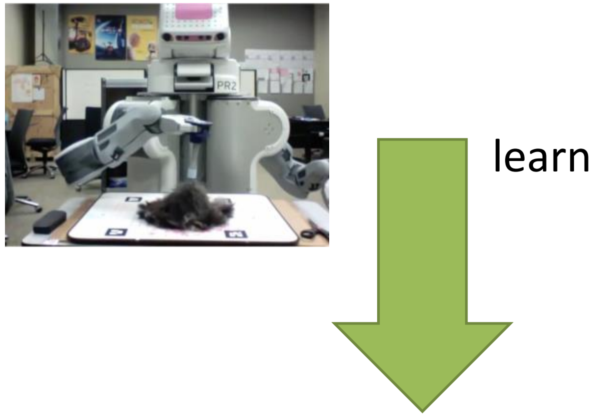
…and many many others!

How can we frame transfer learning problems?

1. Forward transfer: learn policies that transfer effectively
   a) Train on source task, then run on target task (or finetune)
   b) Relies on the tasks being quite similar!
2. Multi-task transfer: train on many tasks, transfer to a new task
   a) Sharing representations and layers across tasks in multi-task learning
   b) New task needs to be similar to the *distribution* of training tasks
3. Meta-learning: learn to *learn* on many tasks
   a) Accounts for the fact that we'll be adapting to a new task during training!

Can we learn **faster** by learning multiple tasks?





Multi-task learning can:

- Accelerate learning of all tasks that are learned together
- Provide better pre-training for down-stream tasks

Can we solve multiple tasks at once?

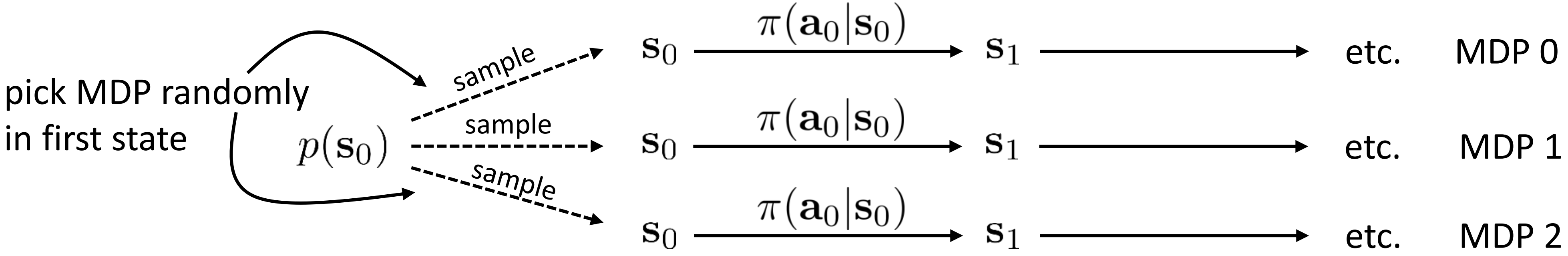
for task $\sim T(\omega_T)$

for episode j in task

Multi-task RL corresponds to single-task RL in a **joint MDP** RL Alg. $\rightarrow \theta$

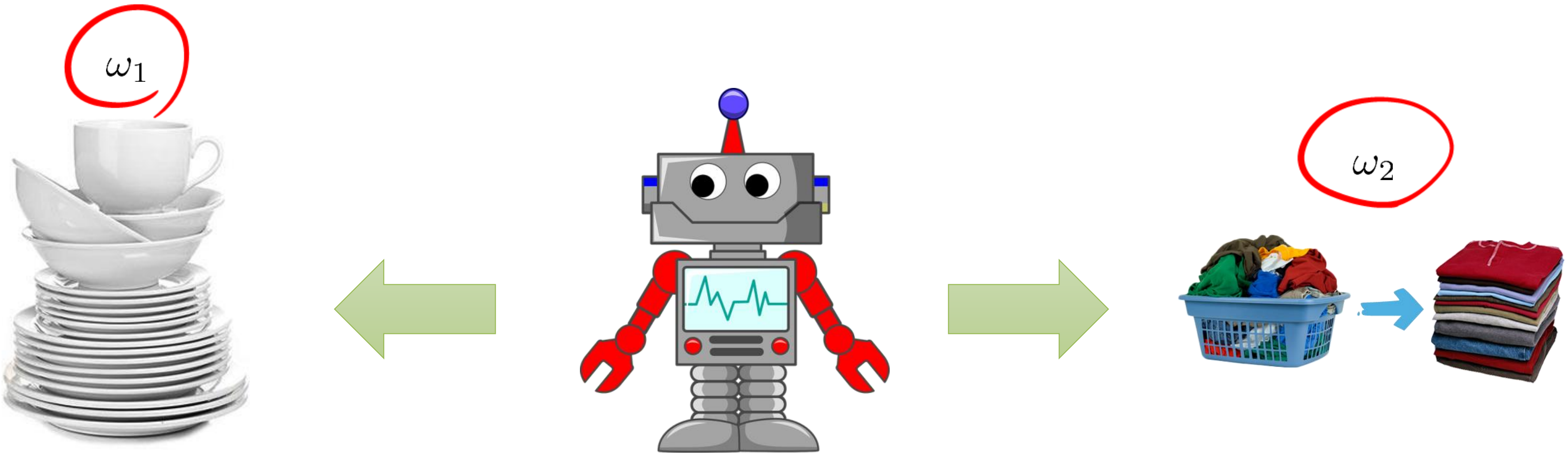
with $\tilde{s} = \begin{bmatrix} s \\ \omega_T \end{bmatrix}$

π_θ



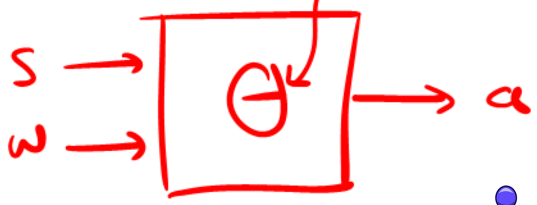
How does the model know what to do?

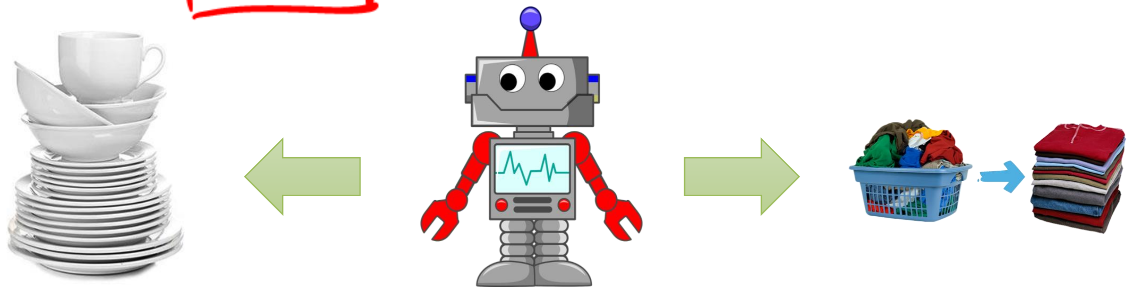
- What if the policy can do *multiple* things in the *same* environment?



Contextual policies

Captures shared sub-tasks



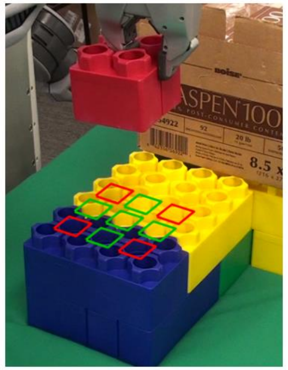


standard policy: $\pi_\theta(\mathbf{a}|\mathbf{s})$

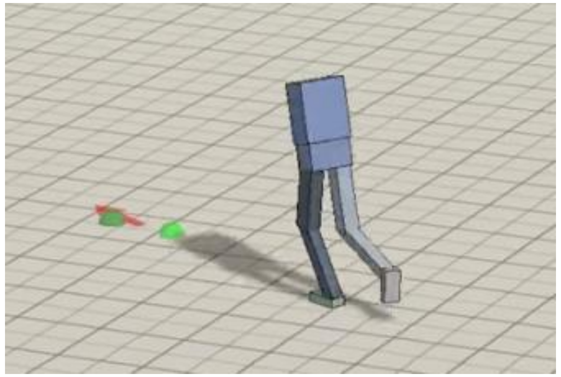
contextual policy: $\pi_\theta(\mathbf{a}|\mathbf{s}, \omega)$

e.g., do dishes or laundry

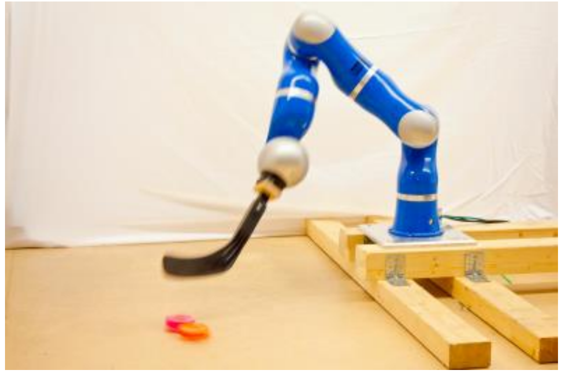
formally, simply defines augmented state space: $\tilde{\mathbf{s}} = \begin{bmatrix} \mathbf{s} \\ \omega \end{bmatrix}$ $\quad \tilde{\mathcal{S}} = \mathcal{S} \times \Omega$



ω: stack location



ω: walking direction



ω: where to hit puck

images: Peng, van de Panne, Peters

Goal-conditioned policies

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

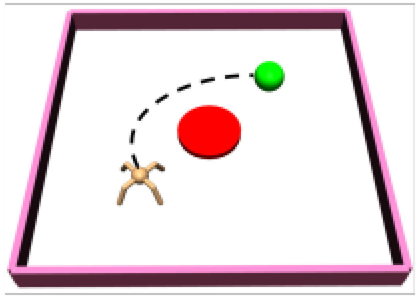
another state

$$r(\mathbf{s}, \mathbf{a}, \mathbf{g}) = \delta(\mathbf{s} = \mathbf{g})$$

$$r(\mathbf{s}, \mathbf{a}, \mathbf{g}) = \delta(||\mathbf{s} - \mathbf{g}|| \leq \varepsilon)$$

- Convenient: no need to manually define rewards for each task
- Can transfer in **zero shot** to a new task if it's another goal!

- Often hard to train in practice (see references)
- Not all tasks are goal reaching tasks!

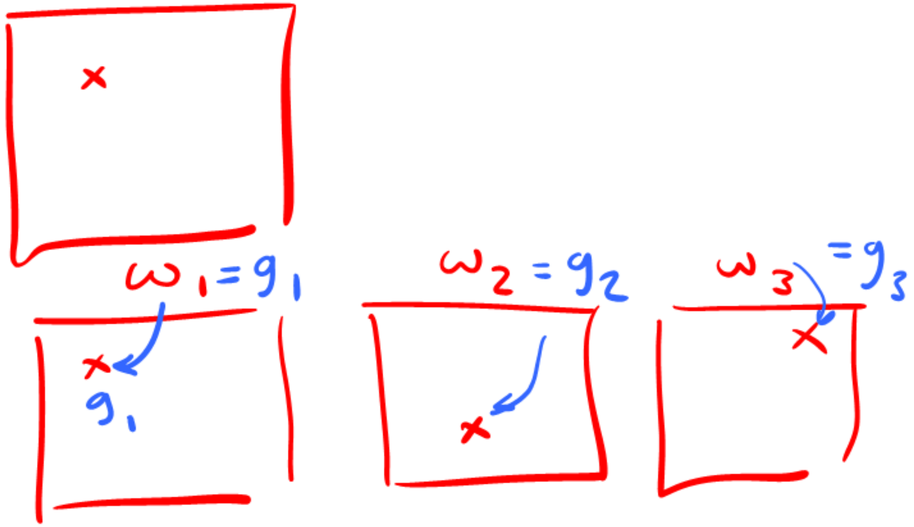


A few relevant papers:

- Kaelbling. **Learning to achieve goals.**
- Schaul et al. **Universal value function approximators.**
- Andrychowicz et al. **Hindsight experience replay**.
- Eysenbach et al. **C-learning: Learning to achieve goals via recursive classification.**

$\pi_\theta(a|s)$

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



Meta-Learning

What is meta-learning?

- If you've learned 100 tasks already, can you figure out how to *learn* more efficiently?
  - Now having multiple tasks is a huge advantage!
- Meta-learning = *learning to learn*
- In practice, very closely related to multi-task learning
- Many formulations
  - Learning an optimizer
  - Learning an RNN that ingests experience
  - Learning a representation

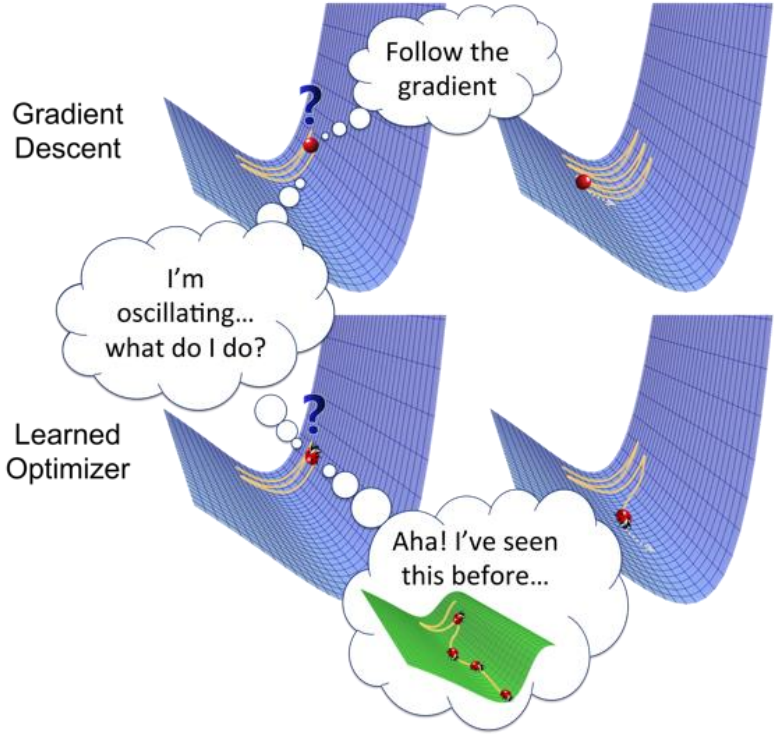


image credit: Ke Li

Why is meta-learning a good idea?

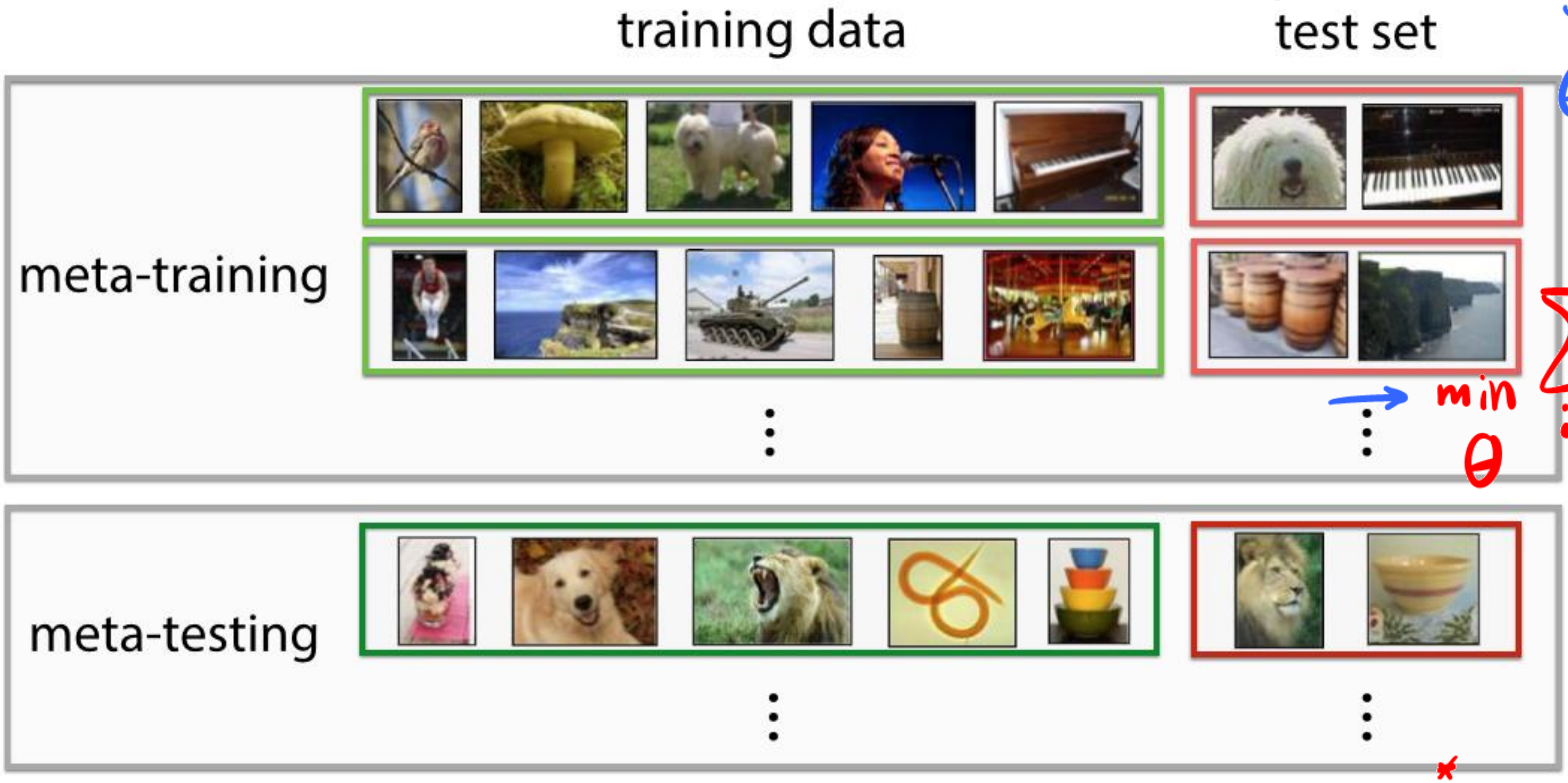
- Deep reinforcement learning, especially model-free, requires a huge number of samples
- If we can *meta-learn* a faster reinforcement learner, we can learn new tasks efficiently!
- What can a *meta-learned* learner do differently?
  - Explore more intelligently
  - Avoid trying actions that are know to be useless
  - Acquire the right features more quickly

Meta-learning with supervised learning

$$\theta = \arg\min_{\theta} \mathcal{L}(\mathcal{D}_{tr}, \theta)$$

$$\theta^* = g(\mathcal{D}_{tr}, \theta)$$

$\mathcal{D}_{tr}$



$$\min_{\theta} \sum_i \mathcal{L}\Big(\mathcal{D}_{ts}^{(i)}, \underbrace{g(\mathcal{D}_{tr}^{(i)}, \theta)}_{\theta^*}\Big)$$

Assume $g \rightarrow$ SGD with initialization of θ

θ θ_1^* θ_2^* θ_3^*

image credit: Ravi & Larochelle ‘17

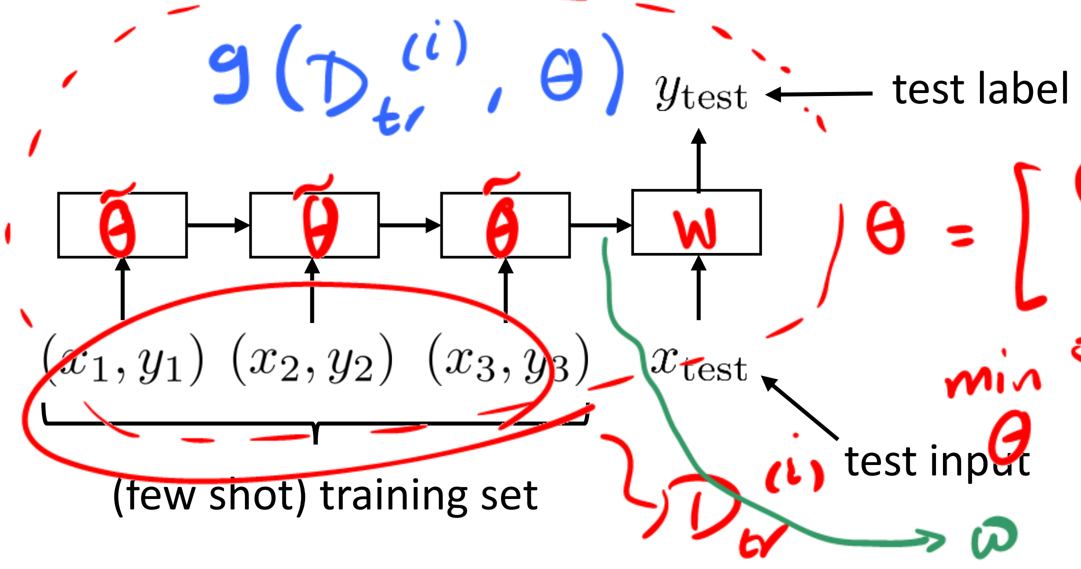
Meta-learning with supervised learning

supervised learning: $f(x) \to y$

input (e.g., image) output (e.g., label)

supervised meta-learning: $f(\mathcal{D}^{\mathrm{tr}}, x) \to y$

training set

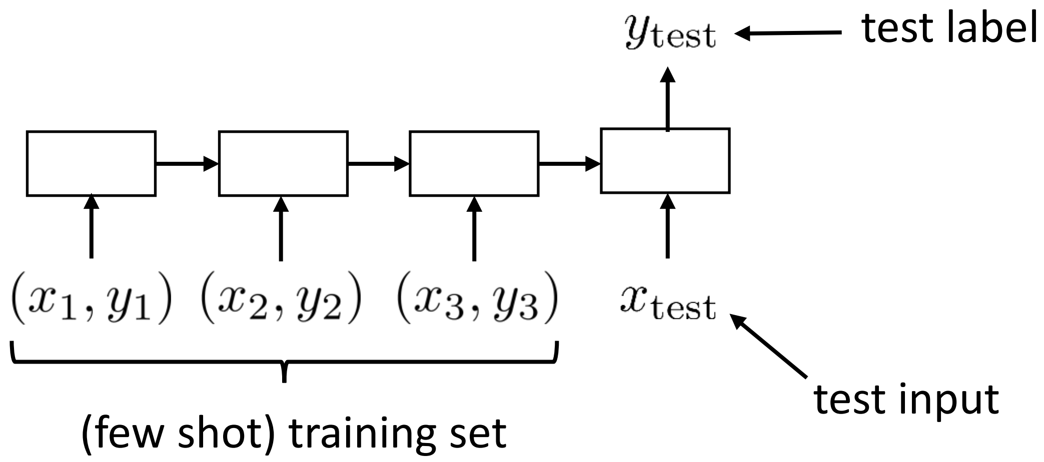


$\theta = \begin{bmatrix} \tilde{\theta} \\ w \end{bmatrix}$

- How to read in training set?
  - Many options, RNNs can work
  - More on this later

$$\min_{\theta} \sum_i \left([y_{test}^{(i)}] - \underbrace{g(\mathcal{D}_{tr}^{(i)}, \theta)}_{f(\theta - \alpha \nabla_\theta loss(\mathcal{D}_{tr}^{(i)}))}[x_{test}^{(i)}] \right)^2$$

What is being "learned"?



supervised meta-learning: $f(\mathcal{D}^{\text{tr}}, x) \rightarrow y$

"Generic" learning:

$$\theta^\star = \arg\min_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

$$= f_{\text{learn}}(\mathcal{D}^{\text{tr}})$$

"Generic" meta-learning:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}})$$

$$\text{where } \phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$$

What is being "learned"?

"Generic" learning:

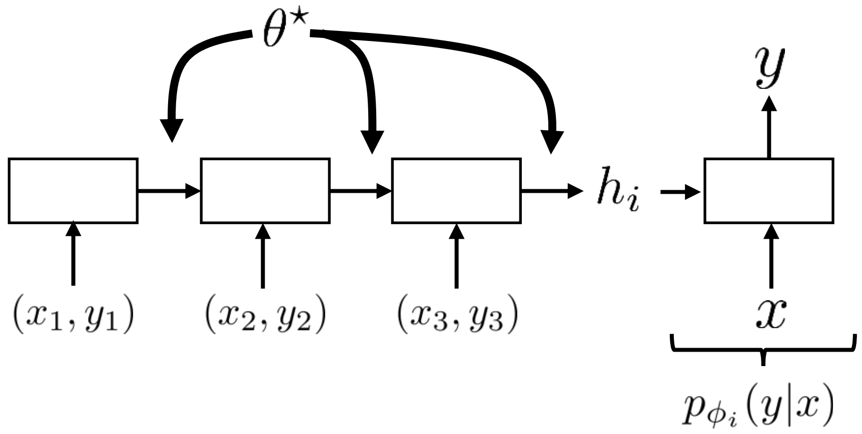
$$\theta^\star = \arg\min_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

$$= f_{\text{learn}}(\mathcal{D}^{\text{tr}})$$

"Generic" meta-learning:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}})$$

$$\text{where } \phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$$



RNN hidden state

meta-learned weights

$$\phi_i = [h_i, \theta_p]$$

Meta **Reinforcement** Learning

The meta reinforcement learning problem

"Generic" learning:

$$\theta^\star = \arg\min_\theta \mathcal{L}(\theta, \mathcal{D}^{\mathrm{tr}})$$

$$= \cancel{f_{\mathrm{learn}}}(\mathcal{D}^{\mathrm{tr}})$$

g

"Generic" meta-learning:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^n \mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{ts}})$$

$$\text{where } \phi_i = \cancel{f_\theta}(\mathcal{D}_i^{\mathrm{tr}}, \theta)$$

g

Reinforcement learning:

$$\theta^\star = \arg\max_\theta E_{\pi_\theta(\tau)}[R(\tau)]$$

$$= \cancel{f_{\mathrm{RL}}}(\mathcal{M}) \qquad \mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}$$

g_{RL}

MDP

Meta-reinforcement learning:

Potentially Contains everything that characterize the RL Alg.

$$\theta^\star = \arg\max_\theta \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

$$\text{where } \phi_i = \cancel{f_\theta}(\mathcal{M}_i, \theta)$$

g

Customized Policy Param.

MDP for task i

The meta reinforcement learning problem

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

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

assumption: $\mathcal{M}_i \sim p(\mathcal{M})$

meta test-time:

sample $\mathcal{M}_{\text{test}} \sim p(\mathcal{M})$, get $\phi_i = f_\theta(\mathcal{M}_{\text{test}})$

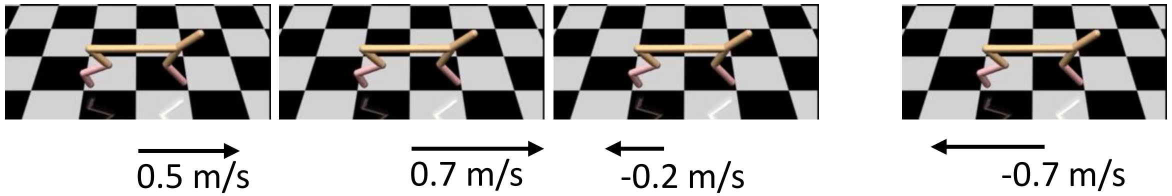
$\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$

meta-training MDPs

Some examples:

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$ $\mathcal{M}_{\text{test}}$

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$ $\mathcal{M}_{\text{test}}$



Contextual policies and meta-learning

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

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

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_\theta}[R(\tau)]$$

$$\pi_\theta(a_t|s_t, \underbrace{s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1}})$$

context used to infer whatever we need to solve $\mathcal{M}_i$
i.e., z_t or ϕ_i (which are really the same thing)

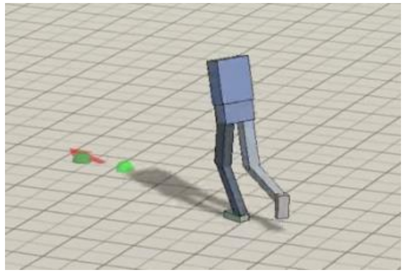
in meta-RL, the *context* is inferred from experience from $\mathcal{M}_i$

in multi-task RL, the context is typically given

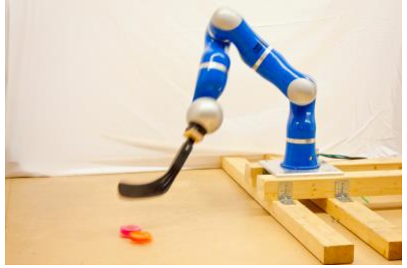
$\pi_\theta(a_t|s_t, \phi_i)$

“context”

ϕ: stack location



ϕ: walking direction



ϕ: where to hit puck

Meta-RL with recurrent policies

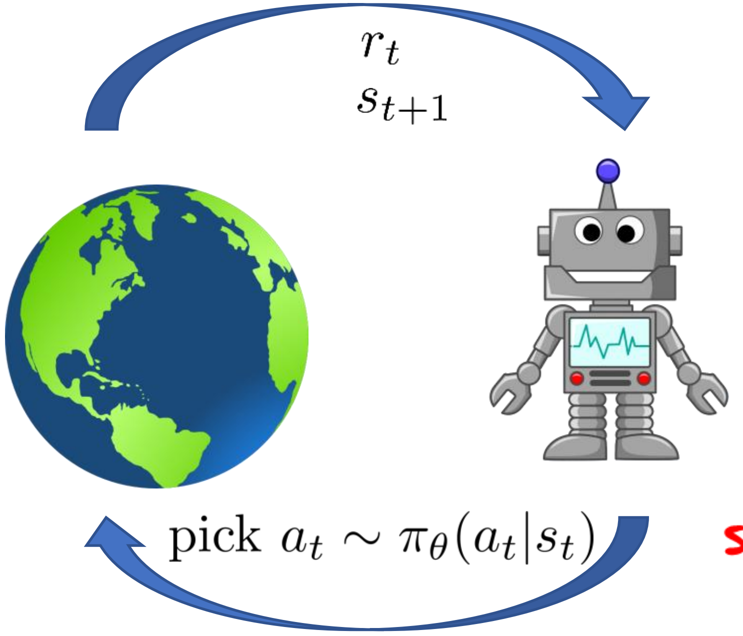
$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

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

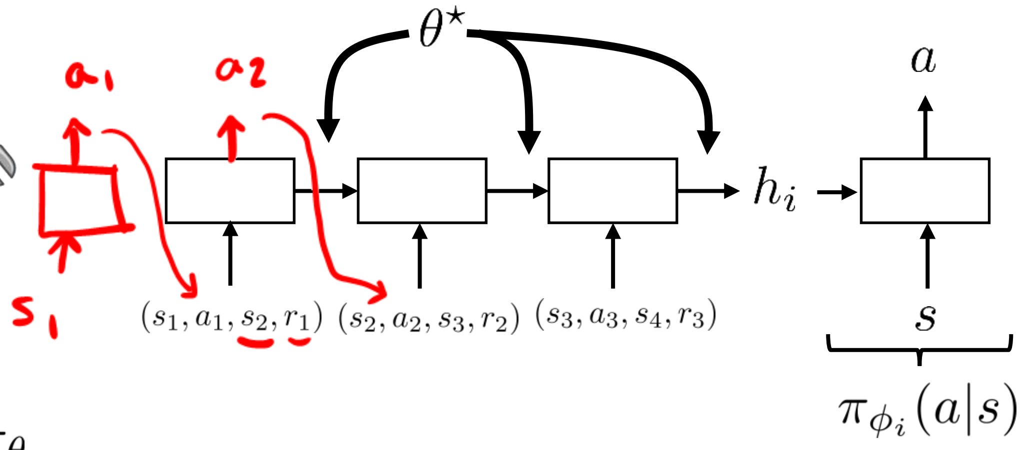
main question: how to implement $f_\theta(\mathcal{M}_i)$?

what should $f_\theta(\mathcal{M}_i)$ *do*?

1. improve policy with experience from $\mathcal{M}_i$
   $\{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\}$
2. (new in RL): choose how to interact, i.e. choose a_t
   meta-RL must also *choose* how to *explore*!



use (s_t, a_t, s_{t+1}, r_t) to improve π_θ



RNN hidden state

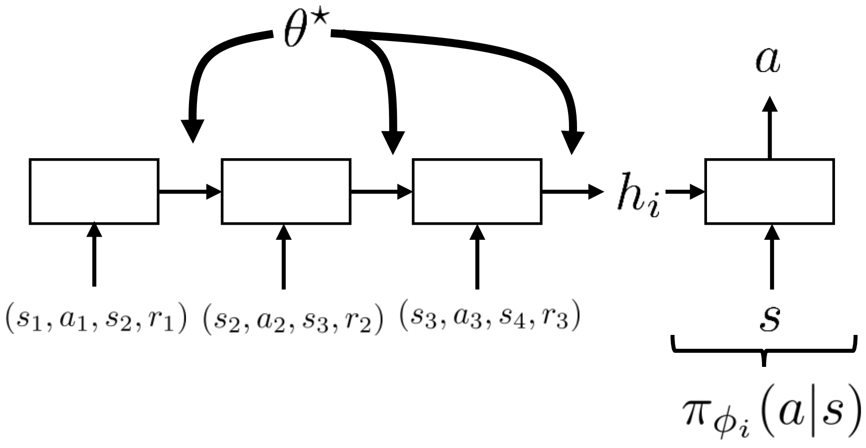
meta-learned weights

as before, $\phi_i = [h_i, \theta_\pi]$

Meta-RL with recurrent policies

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

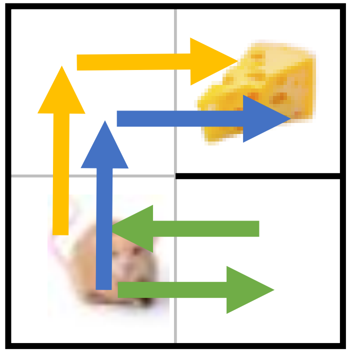
where $\phi_i = f_\theta(\mathcal{M}_i)$

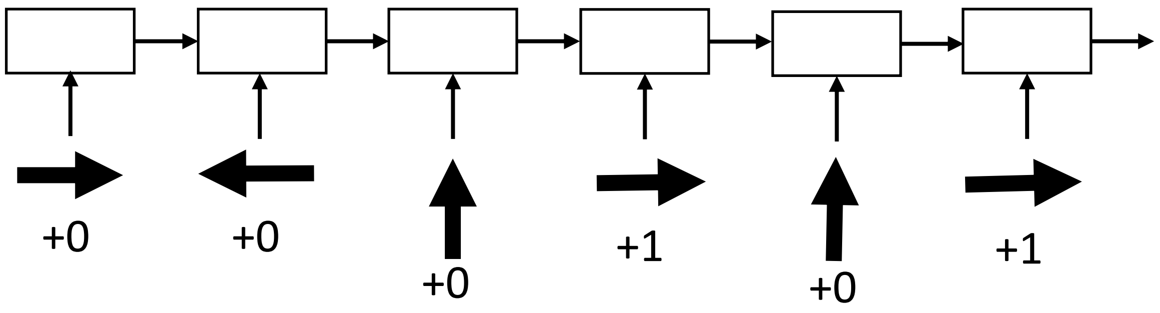


so... we just train an RNN policy?

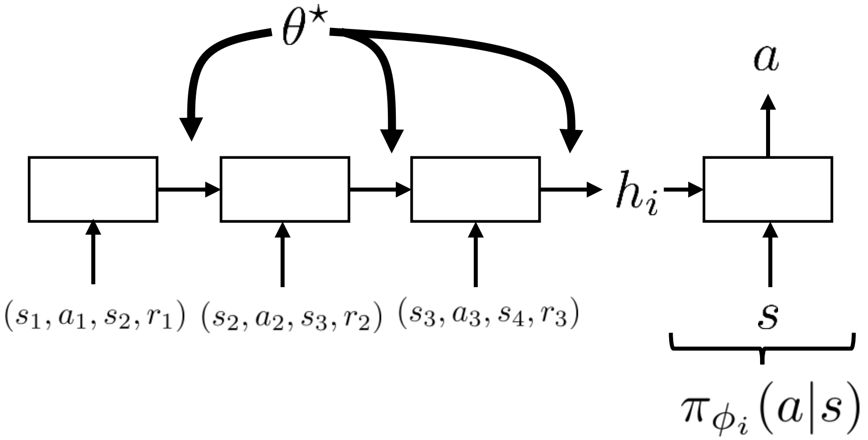
yes!

crucially, RNN hidden state is **not** reset between episodes!





Why recurrent policies *learn to explore*



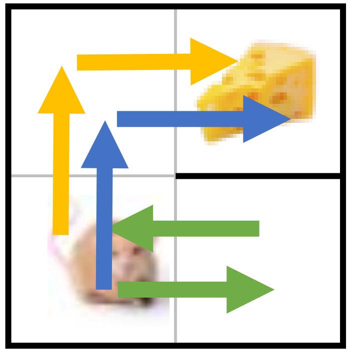
1. improve policy with experience from $\mathcal{M}_i$

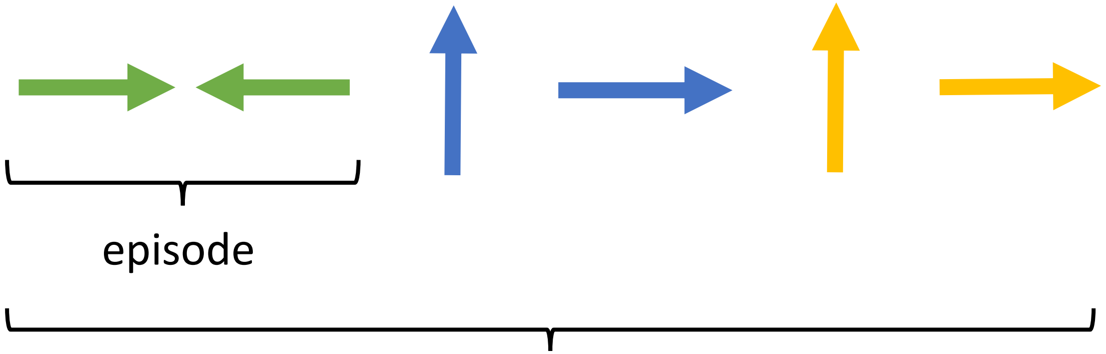
   $\{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\}$

2. (new in RL): choose how to interact, i.e. choose a_t

   meta-RL must also *choose* how to *explore*!

$$\theta^\star = \arg\max_\theta E_{\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$





episode

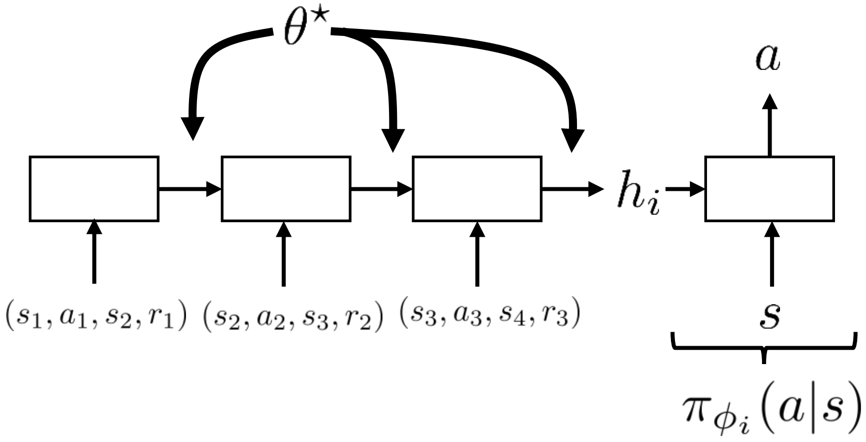
meta-episode

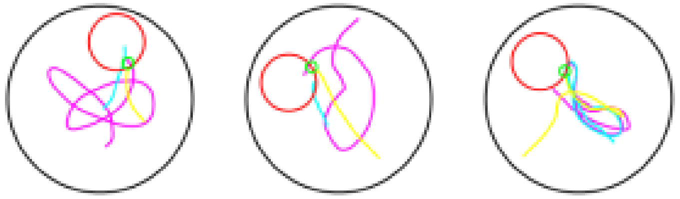
optimizing total reward over the entire **meta-**episode with RNN policy **automatically** learns to explore!

Meta-RL with recurrent policies

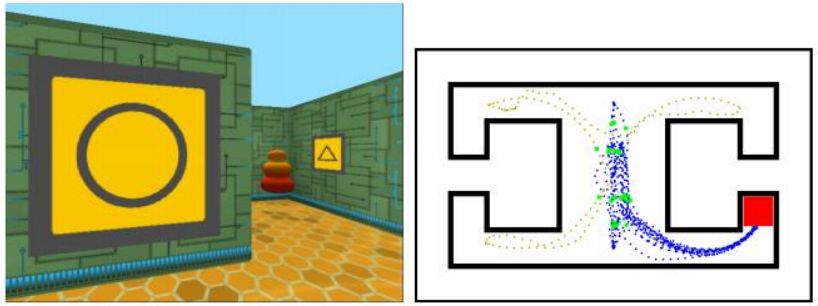
$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

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

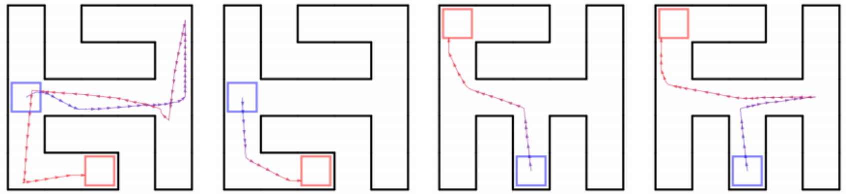




(c) (d) (e)



(a) Labryinth I-maze (b) Illustrative Episode



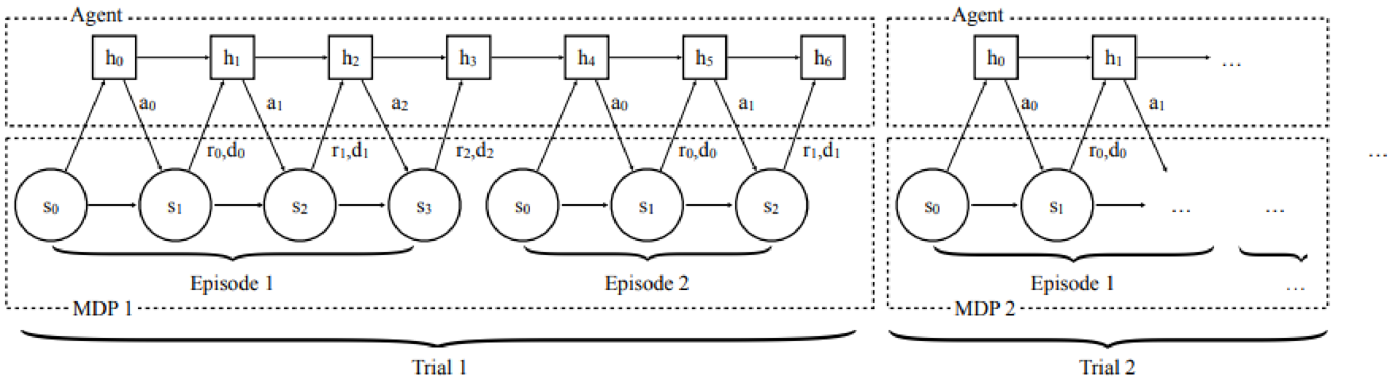
(a) Good behavior, 1st episode (b) Good behavior, 2nd episode (c) Bad behavior, 1st episode (d) Bad behavior, 2nd episode

Heess, Hunt, Lillicrap, Silver. **Memory-based control with recurrent neural networks.** 2015.

Wang, Kurth-Nelson, Tirumala, Soyer, Leibo, Munos, Blundell, Kumaran, Botvinick. **Learning to Reinforcement Learning.** 2016.

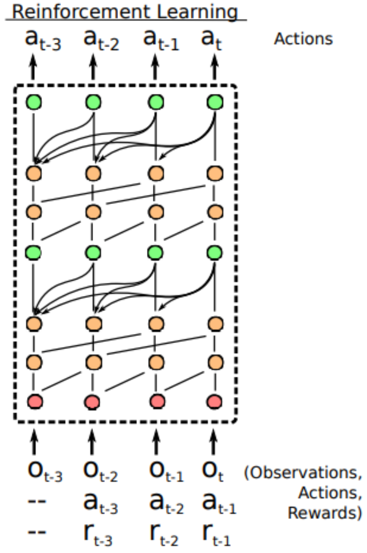
Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. **RL2: Fast Reinforcement Learning via Slow Reinforcement Learning.** 2016.

Architectures for meta-RL



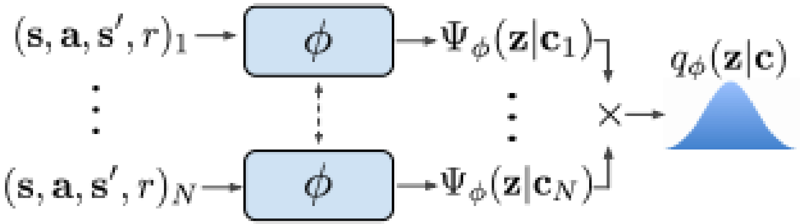
standard RNN (LSTM) architecture

Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. **RL2: Fast Reinforcement Learning via Slow Reinforcement Learning.** 2016.



attention + temporal convolution

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

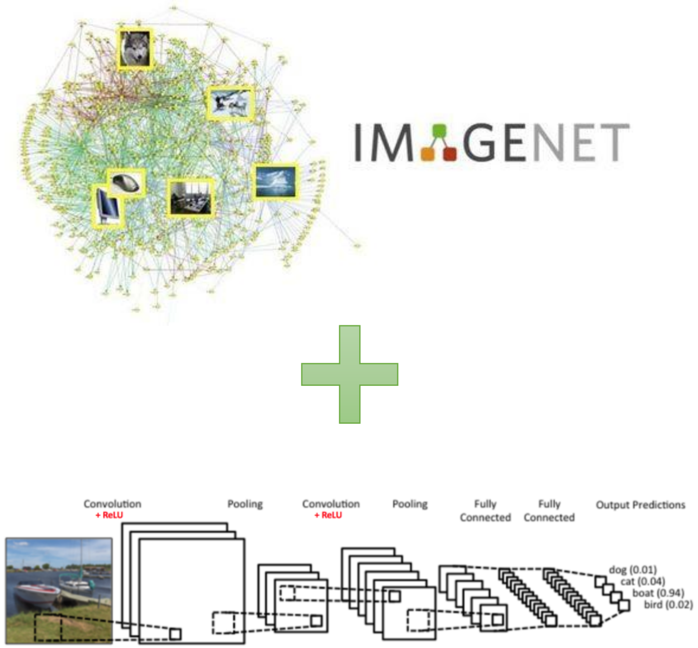


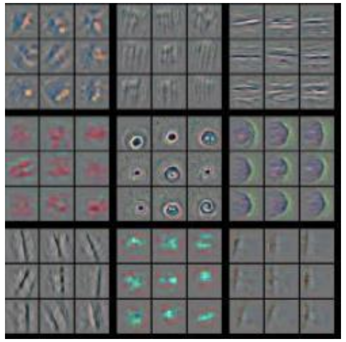
parallel permutation-invariant context encoder

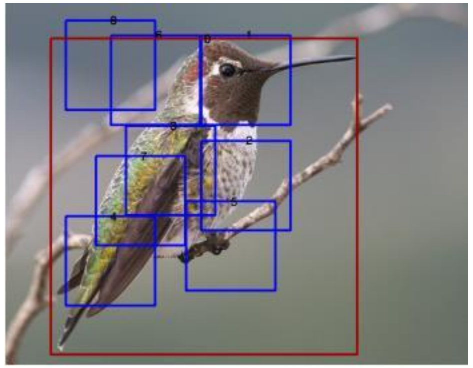
Rakelly*, Zhou*, Quillen, Finn, Levine. **Efficient Off-Policy Meta-Reinforcement learning via Probabilistic Context Variables.**

Gradient-Based Meta-Learning

Back to representations...







is pretraining a *type* of meta-learning?

better features = faster learning of new task!

Meta-RL as an optimization problem

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

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

1. improve policy with experience from $\mathcal{M}_i$

$$\{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\}$$

what if $f_\theta(\mathcal{M}_i)$ is *itself* an RL algorithm?

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta \underbrace{J_i(\theta)}$$

requires interacting with $\mathcal{M}_i$
to estimate $\nabla_\theta E_{\pi_\theta}[R(\tau)]$

standard RL:

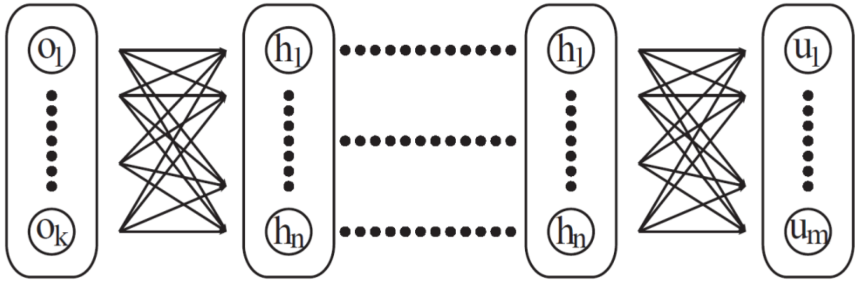
$$\theta^\star = \arg\max_\theta \underbrace{E_{\pi_\theta(\tau)}[R(\tau)]}_{J(\theta)}$$

$$\theta^{k+1} \leftarrow \theta_k + \alpha \nabla_{\theta^k} J(\theta^k)$$

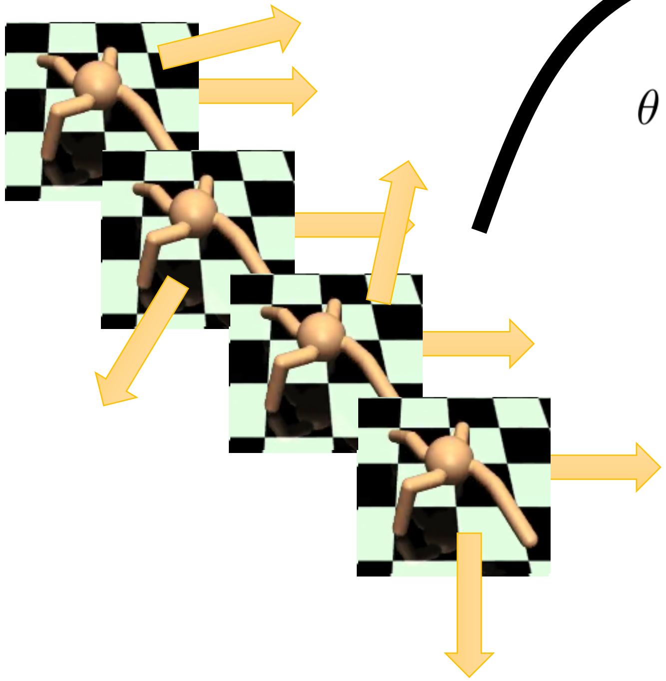
model-agnostic meta-learning (MAML)

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

MAML for RL in pictures



θ

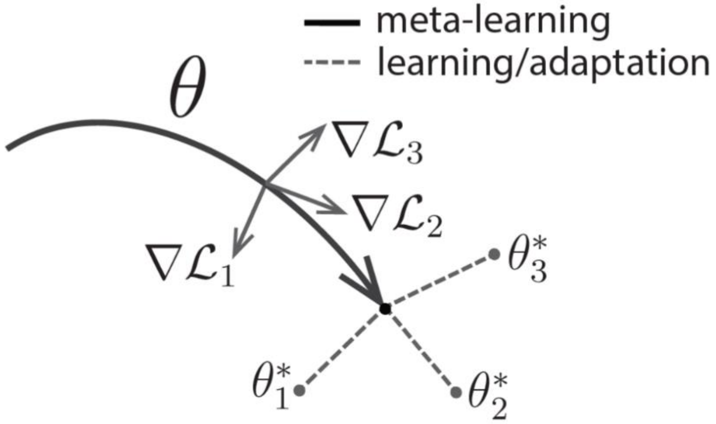


$$\cancel{\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)}$$

$$\theta \leftarrow \theta + \beta \sum_i \nabla_\theta J_i[\underbrace{\theta + \alpha \nabla_\theta J_i(\theta)}_{\text{Adaptain}}]$$

Adaptain → applying g_{RL}

θ'



What did we just do??

supervised learning: $f(x) \to y$

supervised meta-learning: $f(\mathcal{D}^{\text{tr}}, x) \to y$

model-agnostic meta-learning: $f_{\text{MAML}}(\mathcal{D}^{\text{tr}}, x) \to y$

$$f_{\text{MAML}}(\mathcal{D}^{\text{tr}}, x) = f_{\theta'}(x)$$

$$\theta' = \theta - \alpha \sum_{(x,y) \in \mathcal{D}^{\text{tr}}} \nabla_\theta \mathcal{L}(f_\theta(x), y)$$

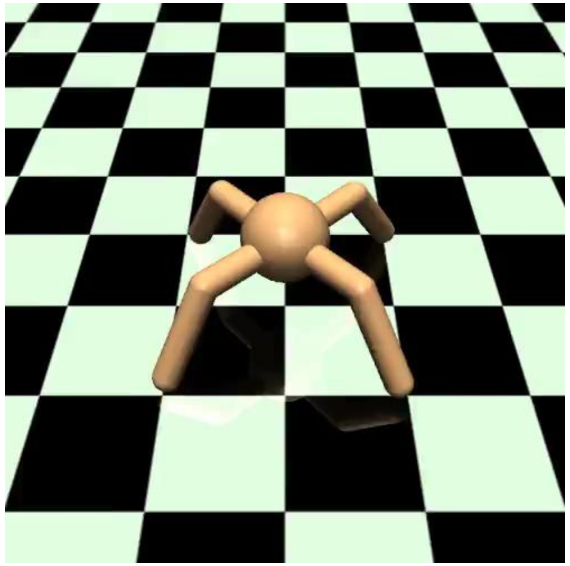
Just another computation graph…

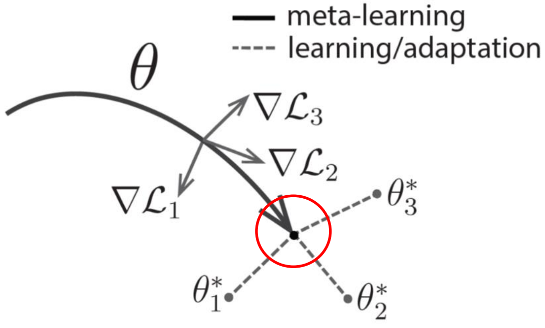
Can implement with any autodiff package (e.g., TensorFlow)

But has favorable inductive bias…

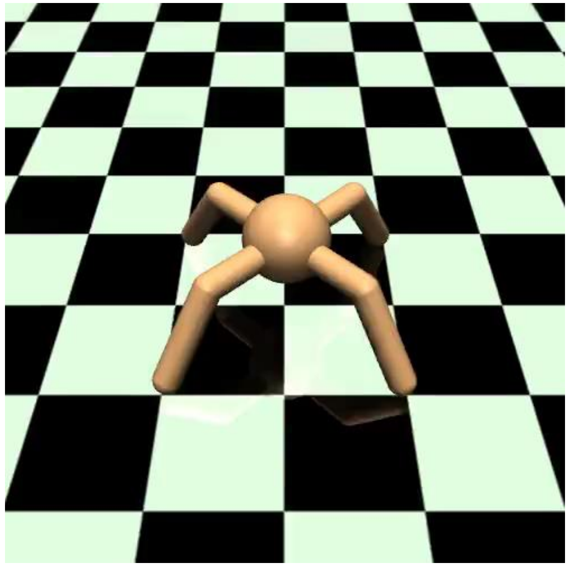
MAML for RL in videos

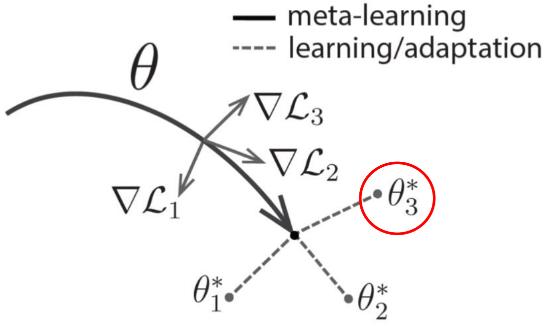
after MAML training



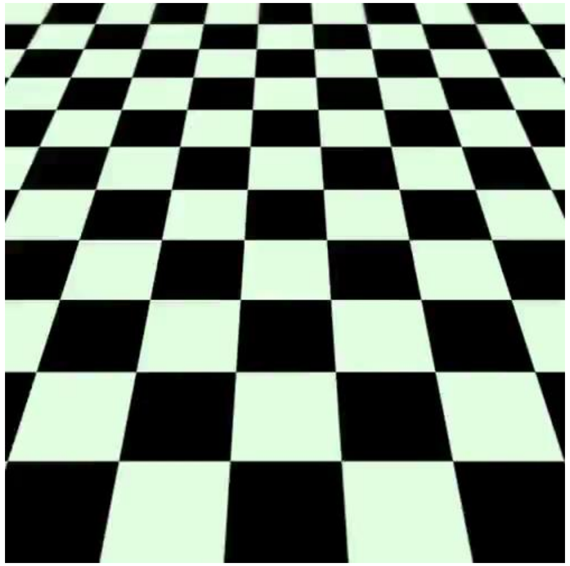


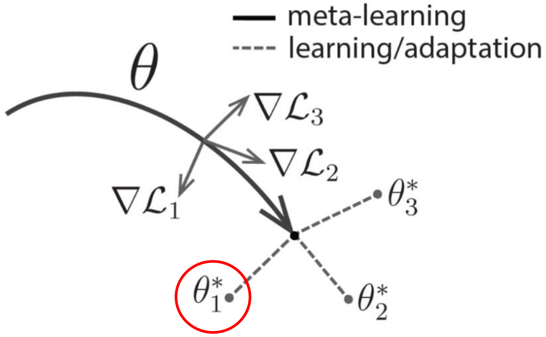
after 1 gradient step (forward reward)





after 1 gradient step (backward reward)





More on MAML/gradient-based meta-learning for RL

MAML meta-policy gradient estimators:

- Finn, Abbeel, Levine. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.**
- Foerster, Farquhar, Al-Shedivat, Rocktaschel, Xing, Whiteson. **DiCE: The Infinitely Differentiable Monte Carlo Estimator.**
- Rothfuss, Lee, Clavera, Asfour, Abbeel. **ProMP: Proximal Meta-Policy Search.**

Improving exploration:

- Gupta, Mendonca, Liu, Abbeel, Levine. **Meta-Reinforcement Learning of Structured Exploration Strategies.**
- Stadie*, Yang*, Houthooft, Chen, Duan, Wu, Abbeel, Sutskever. **Some Considerations on Learning to Explore via Meta-Reinforcement Learning.**

Hybrid algorithms (not necessarily gradient-based):

- Houthooft, Chen, Isola, Stadie, Wolski, Ho, Abbeel. **Evolved Policy Gradients.**
- Fernando, Sygnowski, Osindero, Wang, Schaul, Teplyashin, Sprechmann, Pirtzel, Rusu. **Meta-Learning by the Baldwin Effect.**

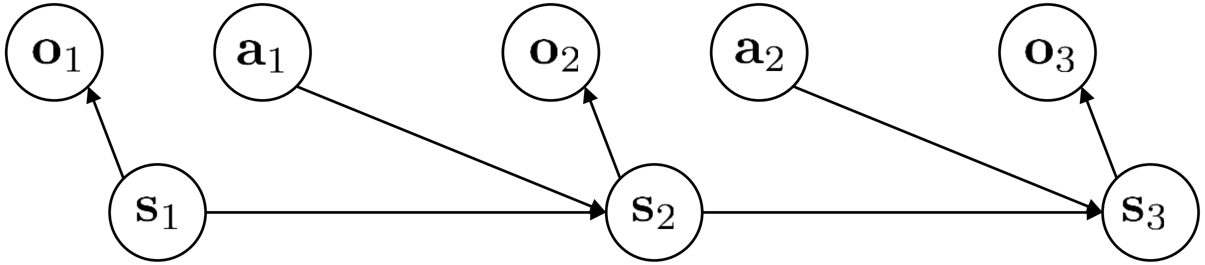
Meta-RL as a POMDP

Meta-RL as... partially observed RL?

$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{E}, r\}$

$\mathcal{O}$ – observation space $\qquad$ observations $o \in \mathcal{O}$ (discrete or continuous)

$\mathcal{E}$ – emission probability $p(o_t|s_t)$



policy must act on observations o_t!

$\pi_\theta(a|o)$

typically requires *either*:

explicit state estimation, i.e. to estimate $p(s_t|o_{1:t})$

policies with memory

Meta-RL as… partially observed RL?

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

encapsulates information policy needs to solve current task

learning a task $=$ inferring z

from *context* $(s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), \ldots$

this is just a POMDP!

before: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}$

now: $\tilde{\mathcal{M}} = \{\tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{O}}, \tilde{\mathcal{P}}, \mathcal{E}, r\}$

$\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}$ $\qquad \tilde{s} = (s, z)$

$\tilde{\mathcal{O}} = \mathcal{S}$ $\qquad \tilde{o} = s$

key idea: solving the POMDP $\tilde{\mathcal{M}}$ is equivalent to meta-learning!

Meta-RL as... partially observed RL?

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

encapsulates information policy
needs to solve current task

learning a task = inferring z

from *context* $(s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), \ldots$

this is just a POMDP!

typically requires *either*:

explicit state estimation, i.e. to estimate $p(s_t|o_{1:t})$

policies with memory

need to estimate $p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$

exploring via posterior sampling with latent context

1. sample $z \sim \hat{p}(z_t|s_{1:t}, a_{1:t}, r_{1:t})$

some approximate posterior
(e.g., variational)

2. act according to $\pi_\theta(a|s,z)$ to collect more data

act *as though z was correct!*

this is *not* optimal!
why?

but it's pretty good,
both in theory and in
practice!

See, e.g. Russo, Roy. **Learning to Optimize via Posterior Sampling.**

Variational inference for meta-RL

policy: $\pi_\theta(a_t|s_t, z_t)$

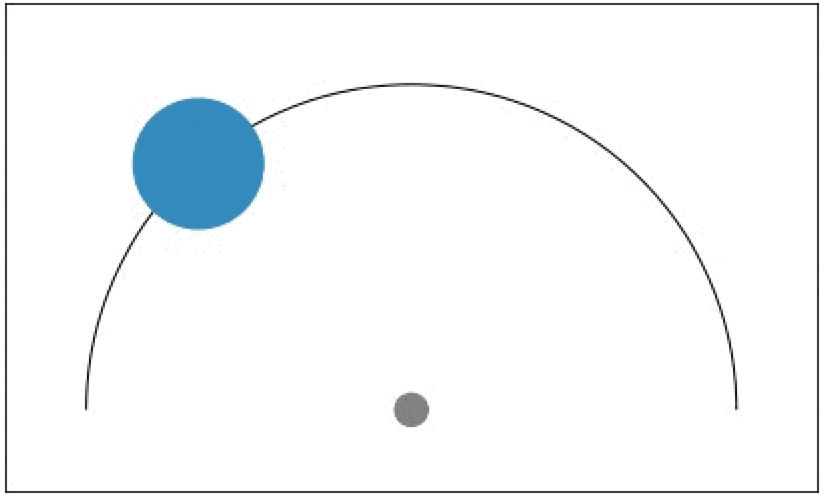
inference network: $q_\phi(z_t|s_1, a_1, r_1, \ldots, s_t, a_t, r_t)$

$$(\theta, \phi) = \arg\max_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{n} E_{z\sim q_\phi, \tau\sim\pi_\theta}[R_i(\tau) - D_{\mathrm{KL}}(q(z|\ldots)\|p(z))]$$

maximize *post-update* reward
(same as standard meta-RL)

stay close to prior

$z_t \sim q_\phi(z_t|s_1, a_1, r_1, \ldots, s_t, a_t, r_t)$



conceptually *very* similar to RNN meta-RL, but with stochastic z

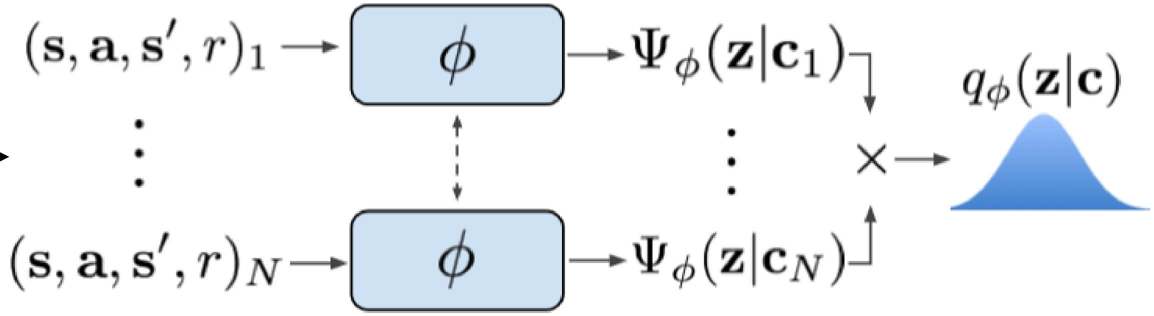
stochastic z enables exploration via *posterior sampling*

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

Specific instantiation: PEARL

policy: $\pi_\theta(a_t|s_t, z_t)$

inference network: $q_\phi(z_t|s_1, a_1, r_1, \dots, s_t, a_t, r_t)$



$$(\theta, \phi) = \arg\max_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{n} E_{z\sim q_\phi, \tau\sim\pi_\theta}[R_i(\tau) - D_{\mathrm{KL}}(q(z|\dots)\|p(z))]$$

perform maximization using soft actor-critic (SAC), state-of-the-art off-policy RL algorithm

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

References on meta-RL, inference, and POMDPs

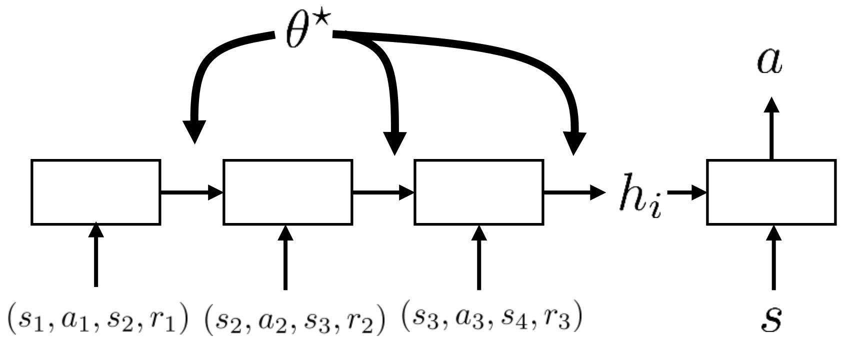
- Rakelly*, Zhou*, Quillen, Finn, Levine. **Efficient Off-Policy Meta-Reinforcement learning via Probabilistic Context Variables.** ICML 2019.

- Zintgraf, Igl, Shiarlis, Mahajan, Hofmann, Whiteson. **Variational Task Embeddings for Fast Adaptation in Deep Reinforcement Learning.**

- Humplik, Galashov, Hasenclever, Ortega, Teh, Heess. **Meta reinforcement learning as task inference.**

The three perspectives on meta-RL

Perspective 1: just RNN it



Perspective 2: bi-level optimization

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

MAML for RL

Perspective 3: it's an inference problem!

$$\pi_\theta(a|s, z) \qquad z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$$

everything needed to solve task

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

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

what should $f_\theta(\mathcal{M}_i)$ *do*?

1. improve policy with experience from $\mathcal{M}_i$

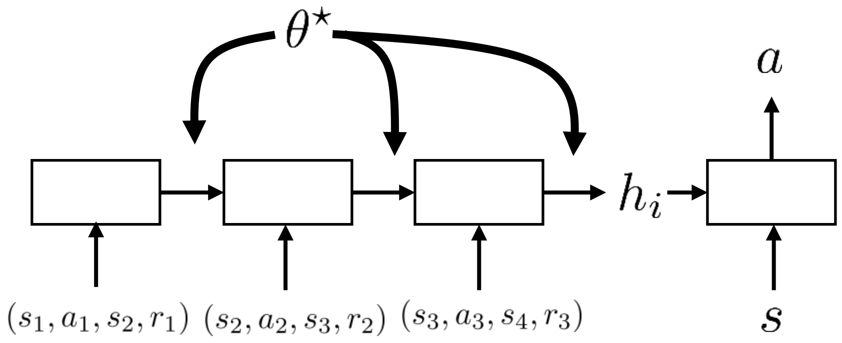
   $\{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\}$

2. (new in RL): choose how to interact, i.e. choose a_t

   meta-RL must also *choose* how to *explore*!

The three perspectives on meta-RL

Perspective 1: just RNN it



+ conceptually simple
+ relatively easy to apply
- vulnerable to *meta-overfitting*
- challenging to optimize in practice

Perspective 2: bi-level optimization

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

MAML for RL

+ good extrapolation ("consistent")
+ conceptually elegant
- complex, requires many samples

Perspective 3: it's an inference problem!

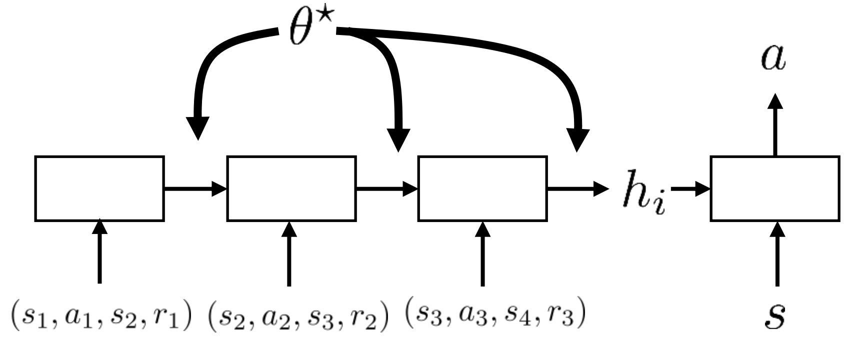
$$\pi_\theta(a|s, z) \qquad z_t \sim p(z_t | s_{1:t}, a_{1:t}, r_{1:t})$$

everything needed to solve task

+ simple, effective exploration via posterior sampling
+ elegant reduction to solving a special POMDP
- vulnerable to *meta-overfitting*
- challenging to optimize in practice

But they're not that different!

Perspective 1: just RNN it



just perspective 1, but with stochastic hidden variables!

i.e., $\phi = \mathbf{z}$

Perspective 2: bi-level optimization

$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

MAML for RL

just a particular architecture choice for these

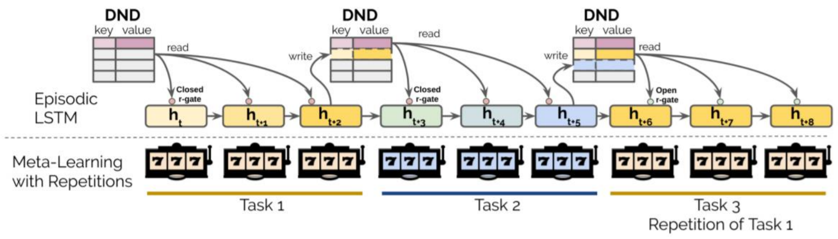
Perspective 3: it's an inference problem!

$\pi_\theta(a|s, z)$ $\qquad z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t})$

everything needed to solve task

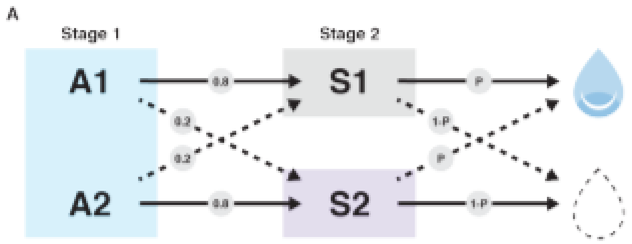
Meta-RL and emergent phenomena

meta-RL gives rise to episodic learning



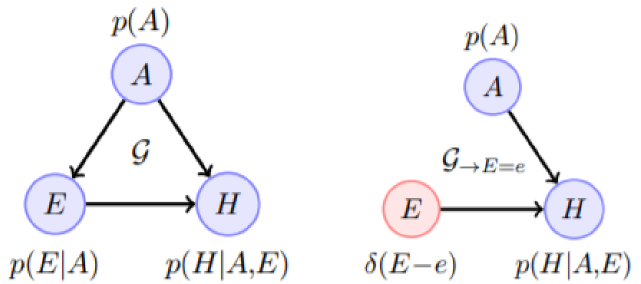
Ritter, Wang, Kurth-Nelson, Jayakumar, Blundell, Pascanu, Botvinick. **Been There, Done That: Meta-Learning with Episodic Recall.**

model-free meta-RL gives rise to model-based adaptation



Wang, Kurth-Nelson, Kumaran, Tirumala, Soyer, Leibo, Hassabis, Botvinick. **Prefrontal Cortex as a Meta-Reinforcement Learning System.**

meta-RL gives rise to causal reasoning (!)



Dasgupta, Wang, Chiappa, Mitrovic, Ortega, Raposo, Hughes, Battaglia, Botvinick, Kurth-Nelson. **Causal Reasoning from Meta-Reinforcement Learning.**

Humans and animals *seemingly* learn behaviors in a variety of ways:

- Highly efficient but (apparently) model-free RL
- Episodic recall
- Model-based RL
- Causal inference
- etc.

Perhaps each of these is a separate "algorithm" in the brain

But maybe these are all emergent phenomena resulting from meta-RL?