

Reinforcement Learning

Naeemullah Khan

naeemullah.khan@kaust.edu.sa



جامعة الملك عبد الله
للغعلوم والتكنولوجية
King Abdullah University of
Science and Technology

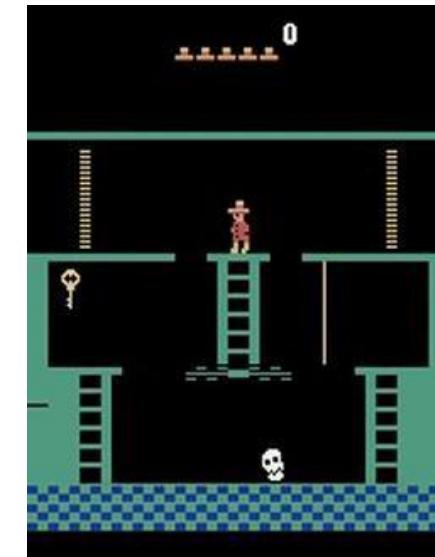
KAUST Academy
King Abdullah University of Science and Technology

What's the problem?

this is easy (mostly)

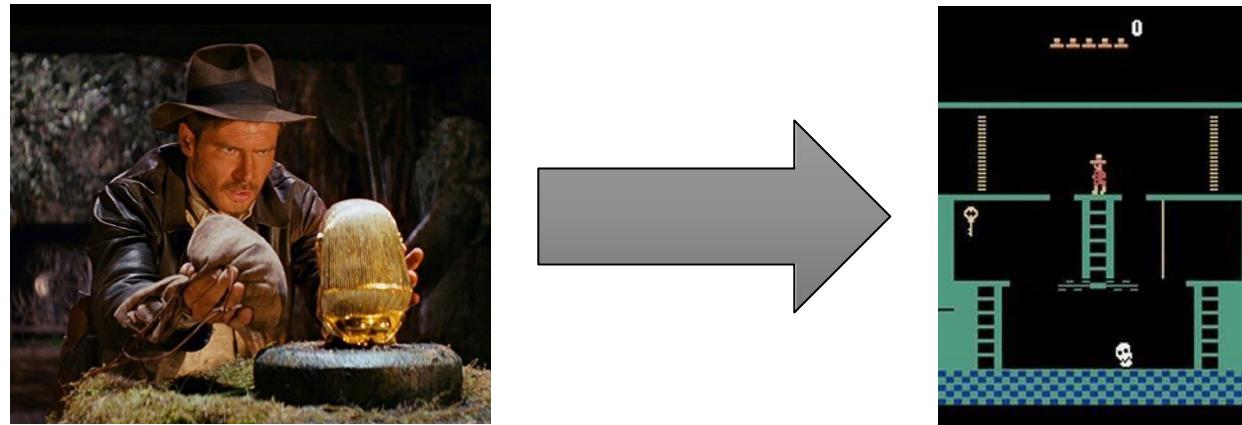


this is difficult



why?

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!

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!

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!

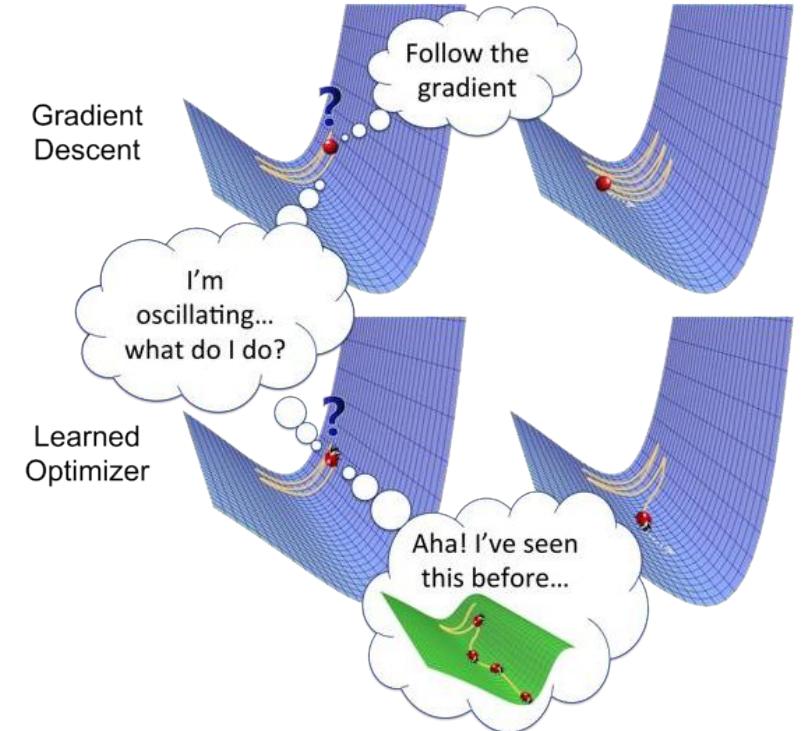


How can we frame transfer learning problems?

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!

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



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 known to be useless
 - Acquire the right features more quickly

Meta-learning with supervised learning

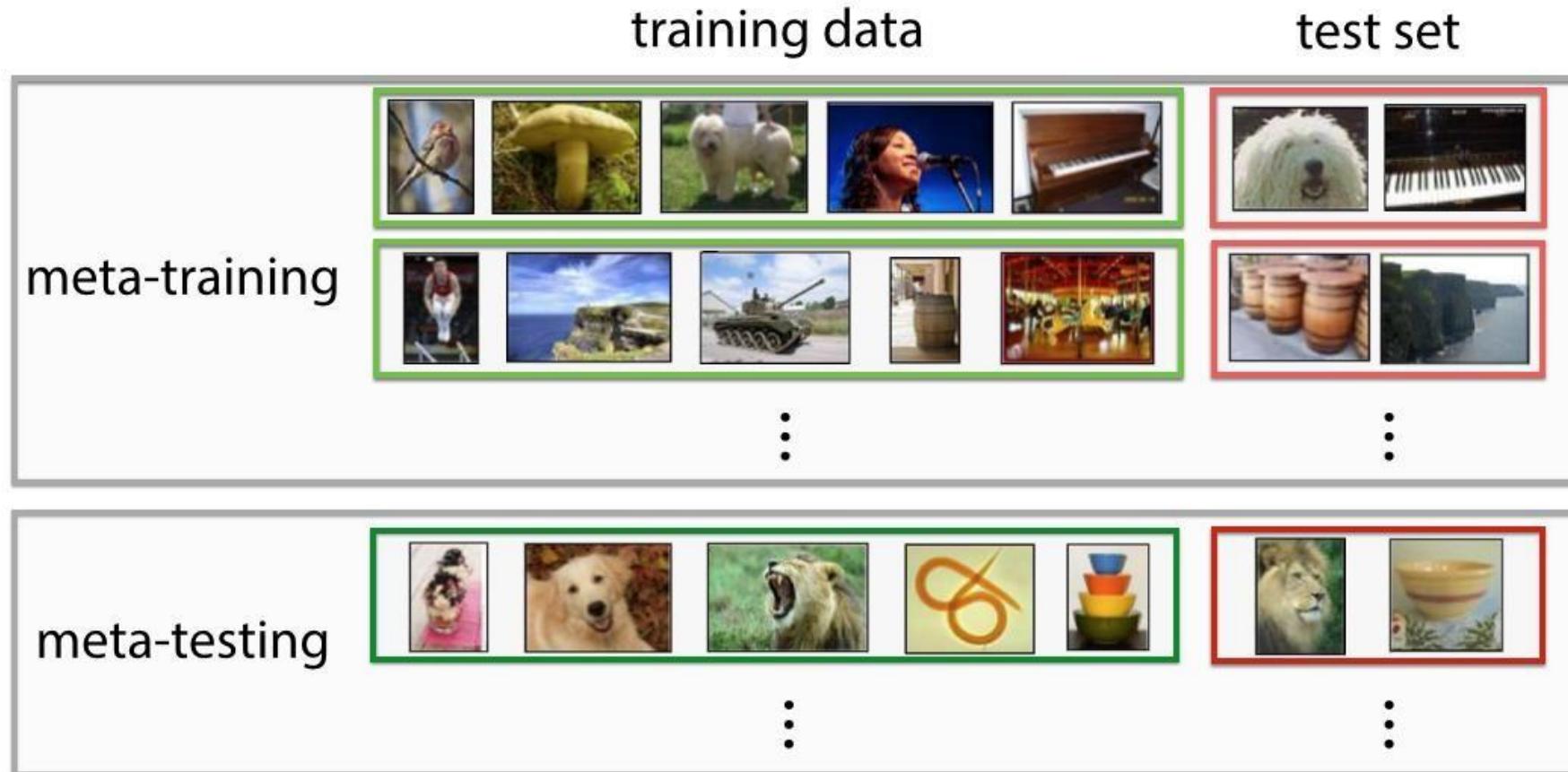
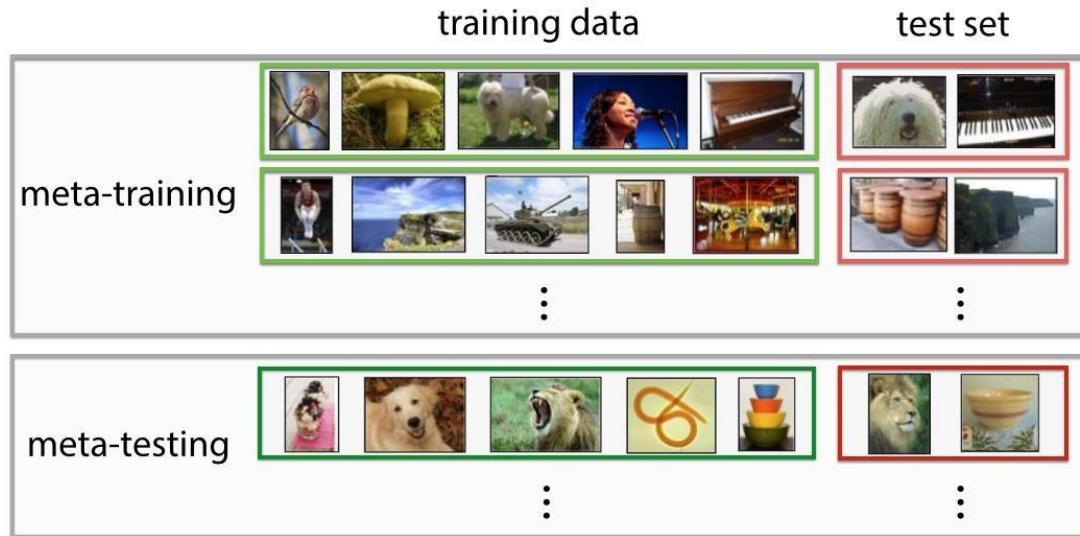


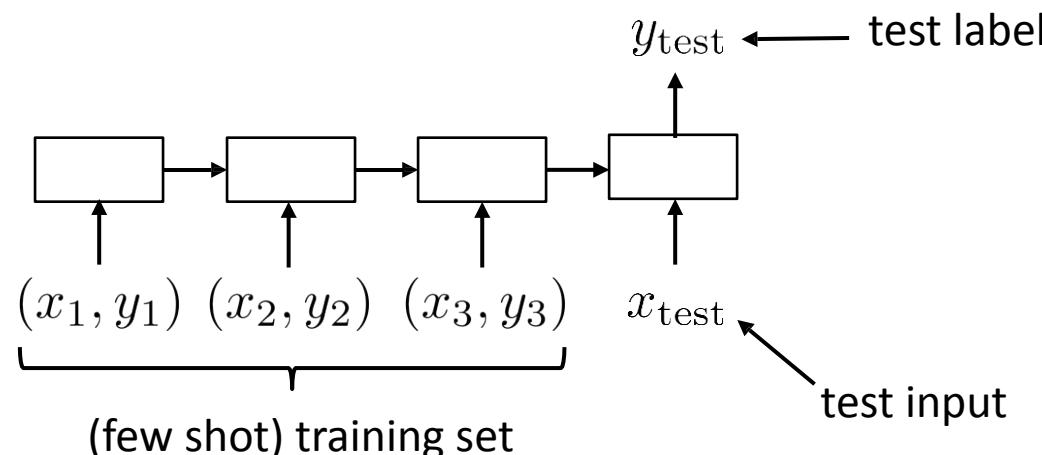
image credit: Ravi & Larochelle '17

Meta-learning with supervised learning



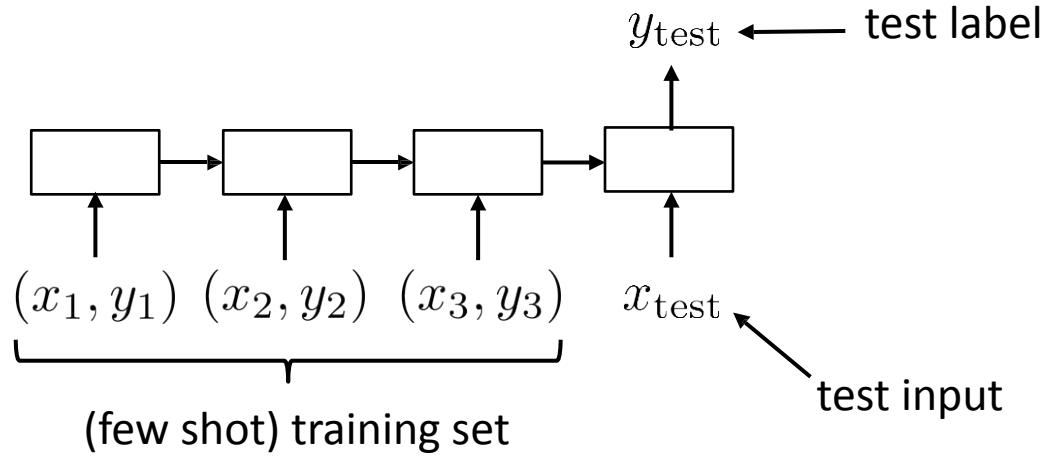
supervised learning: $f(x) \rightarrow y$
input (e.g., image) output (e.g., label)

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



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

What is being “learned”?



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

“Generic” learning:

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}) \\ &= f_{\text{learn}}(\mathcal{D}^{\text{tr}})\end{aligned}$$

“Generic” meta-learning:

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

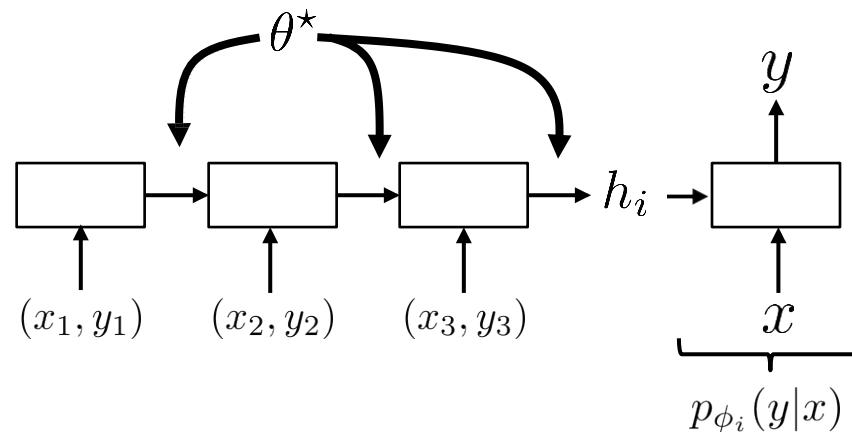
where $\phi_i = f_{\theta}(\mathcal{D}_i^{\text{tr}})$

What is being “learned”?

“Generic” learning:

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

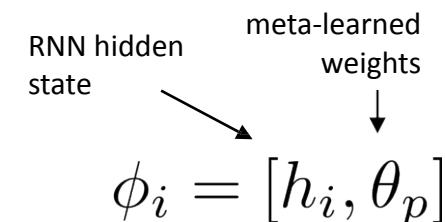
$$= f_{\text{learn}}(\mathcal{D}^{\text{tr}})$$



“Generic” meta-learning:

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

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

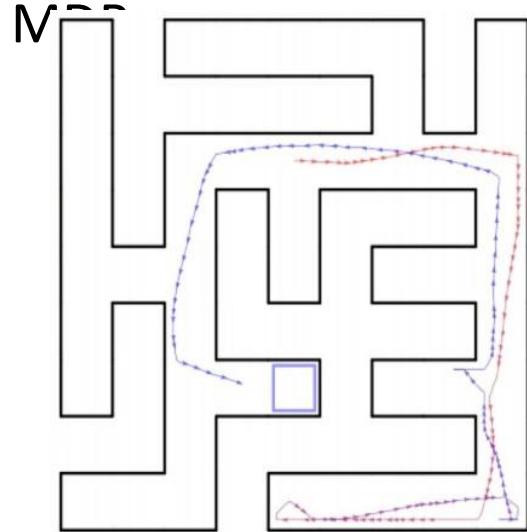


Meta Reinforcement Learning

Meta-RL Example: Maze Navigation

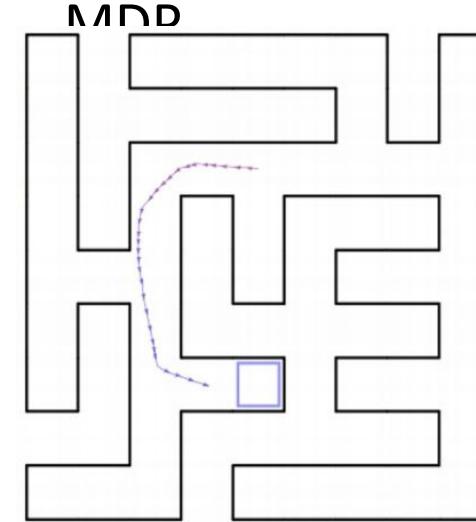
Goal:

Collect small amount
of experience in new



Collect $\mathcal{D}_{\text{tr}} \sim \pi^{\text{exp}}$

Learn policy
that solves that



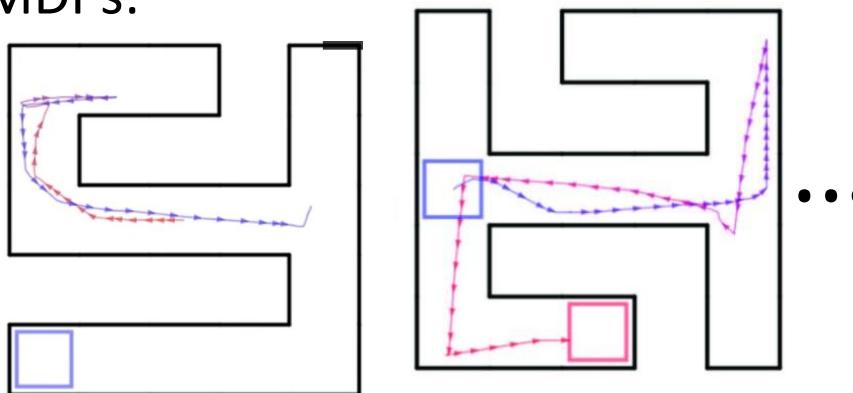
$\mathcal{D}_{\text{tr}} \rightarrow \pi^{\text{task}}$

diagram adapted from Duan et al.

Meta-RL Example: Maze Navigation

Meta-Train Time:

Learn how to efficiently explore & solve many MDPs:

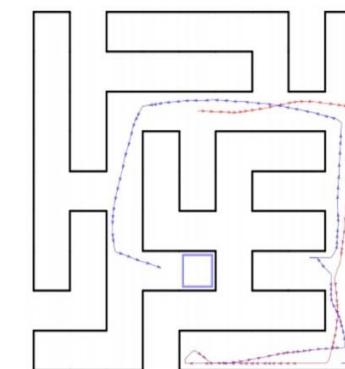


Meta-train π^{exp} ,
 π^{task}

meta-training
tasks

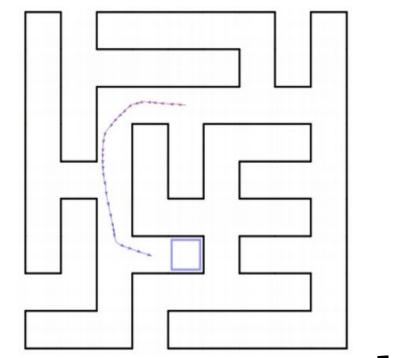
Meta-Test Time:

Collect small amount of experience in new MDP



Collect $\mathcal{D}_{\text{tr}} \sim \pi^{\text{exp}}$

Learn policy that solves that MDP



$\mathcal{D}_{\text{tr}} \rightarrow \pi^{\text{task}}$

Key assumption: Meta-training & meta-testing MDPs come from same distribution.

The meta reinforcement learning problem



“Generic” learning:

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}) \\ &= f_{\text{learn}}(\mathcal{D}^{\text{tr}})\end{aligned}$$

Reinforcement learning:

$$\begin{aligned}\theta^* &= \arg \max_{\theta} E_{\pi_{\theta}(\tau)}[R(\tau)] \\ &= f_{\text{RL}}(\mathcal{M}) \quad \mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}\end{aligned}$$

MDP

“Generic” meta-learning:

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \sum_{i=1}^n \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}}) \\ \text{where } \phi_i &= f_{\theta}(\mathcal{D}_i^{\text{tr}})\end{aligned}$$

Meta-reinforcement learning:

$$\begin{aligned}\theta^* &= \arg \max_{\theta} \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)] \\ \text{where } \phi_i &= f_{\theta}(\mathcal{M}_i)\end{aligned}$$

MDP for task i

The meta reinforcement learning problem

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

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

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

meta test-time:

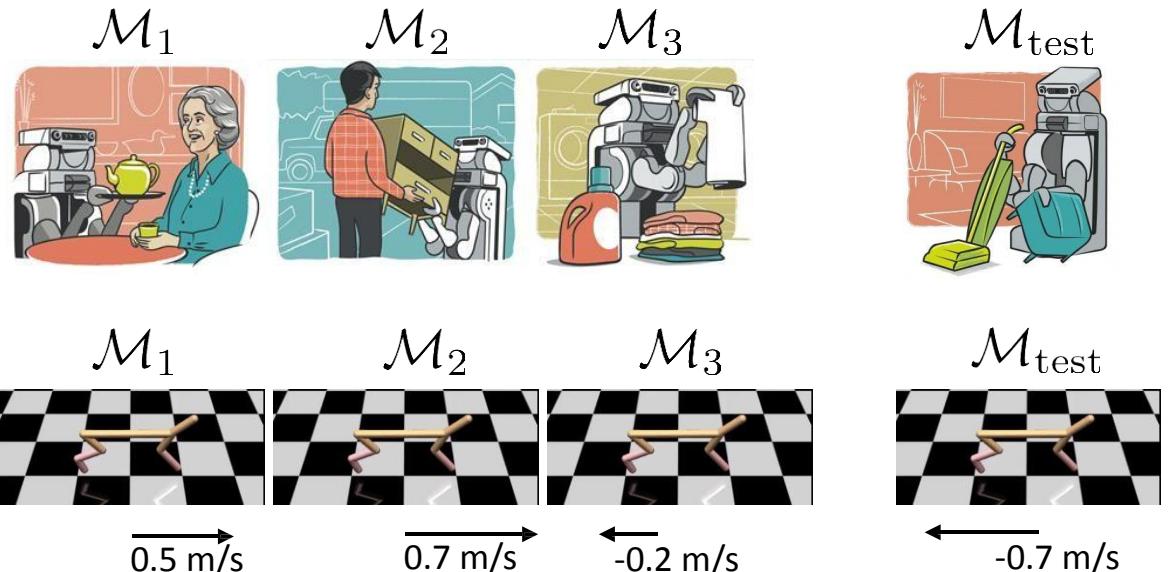
sample $\mathcal{M}_{\text{test}} \sim p(\mathcal{M})$, get $\phi_i = f_{\theta}(\mathcal{M}_{\text{test}})$

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



meta-training MDPs

Some examples:



Contextual policies and meta-learning

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

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



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

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

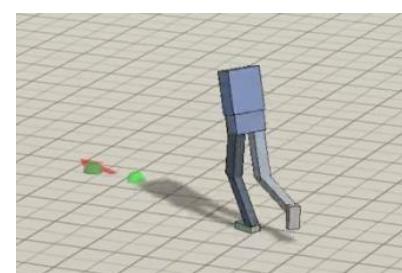
context used to infer whatever we need to solve \mathcal{M}_i
i.e., z_t or ϕ_i (which are really the same thing)

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

in multi-task RL, the context is typically given



ϕ : stack location



ϕ : walking direction



ϕ : where to hit puck

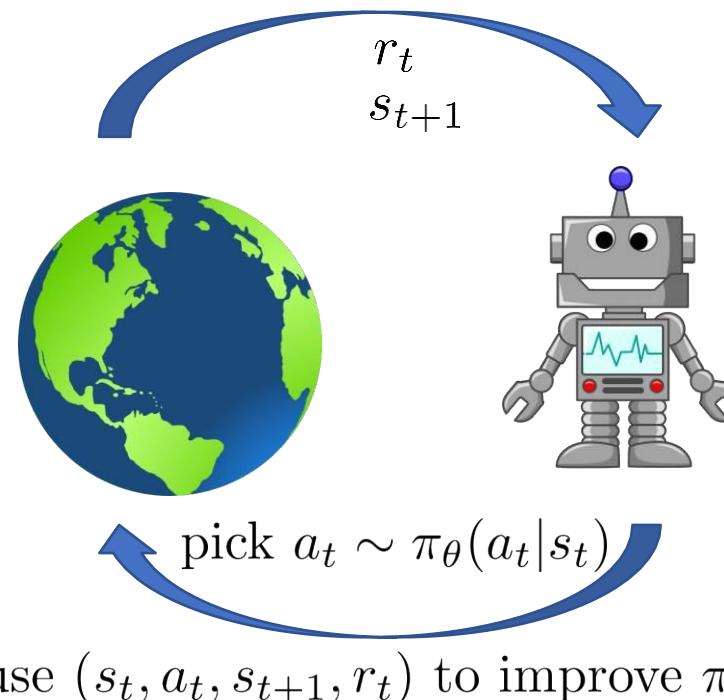


Meta-RL with recurrent policies

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

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

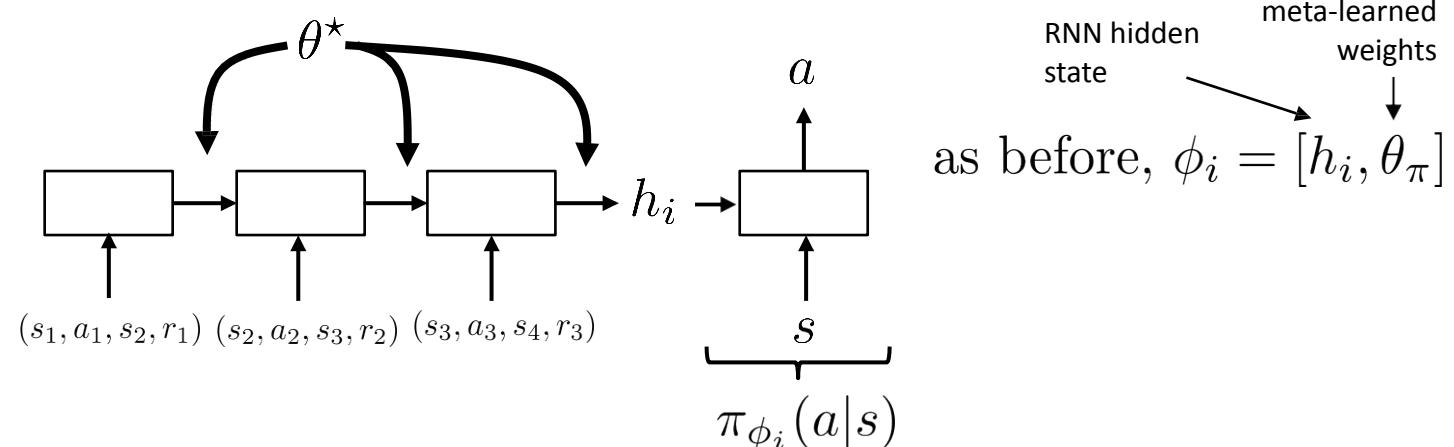
where $\phi_i = f_\theta(\mathcal{M}_i)$



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
 $\{(s_1, a_1, s_2, r_1), \dots, (s_T, a_T, s_{T+1}, r_T)\}$
 2. (new in RL): choose how to interact, i.e. choose a_t
meta-RL must also *choose* how to *explore*!



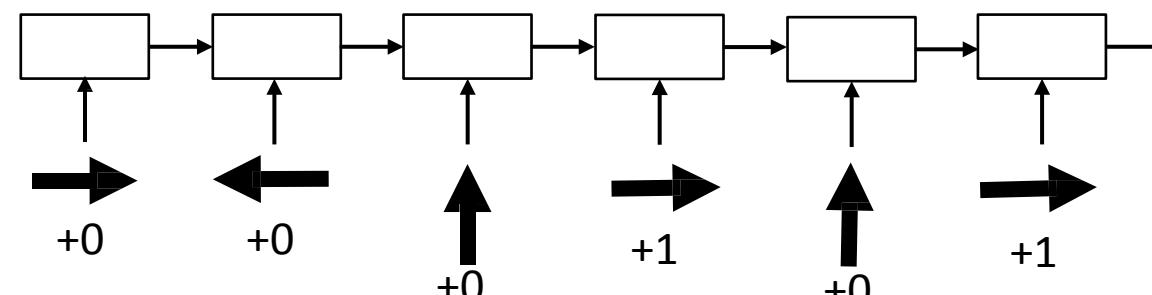
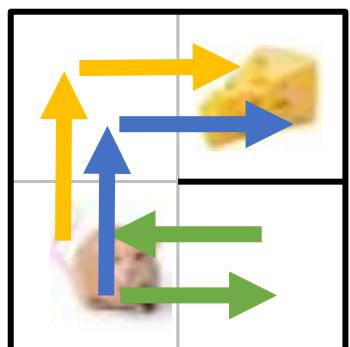
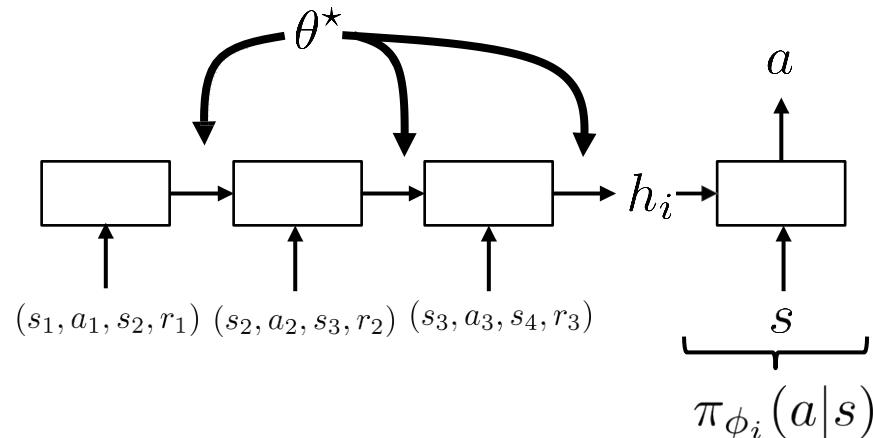
Meta-RL with recurrent policies

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

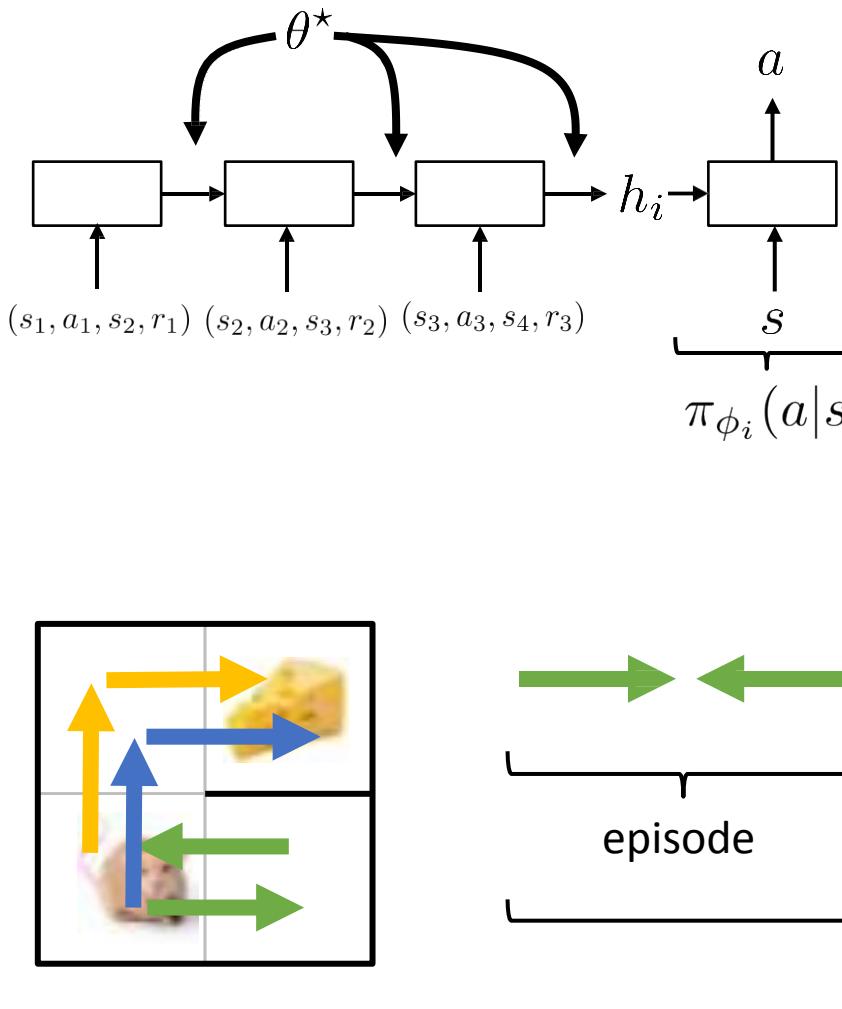
where $\phi_i = f_{\theta}(\mathcal{M}_i)$

so... we just train an RNN policy?

yes!



Why recurrent policies learn to explore



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

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

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

Meta-Training

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

Meta-Test Time

1. Sample *new* task \mathcal{T}_j
2. Roll-out policy $\pi(a | s, \mathcal{D}^{\text{tr}})$ for up to N episodes

Meta-RL Example

Experiment: Learning to visually navigate a maze

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

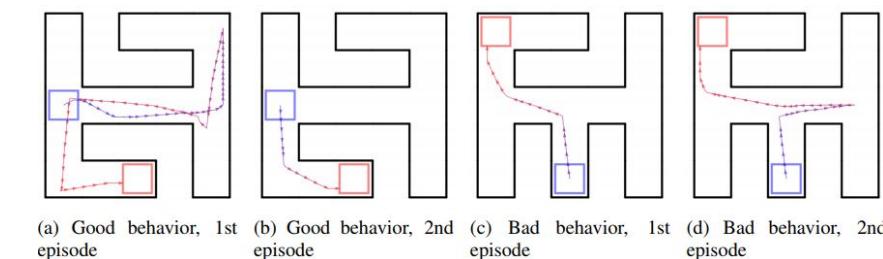
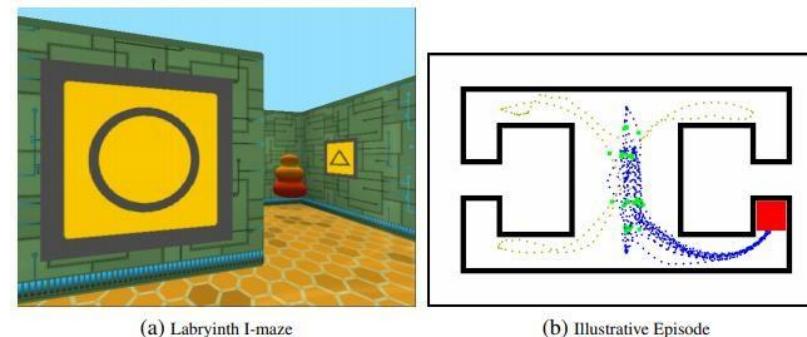
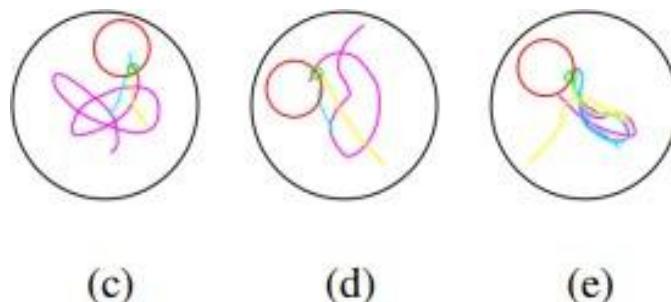
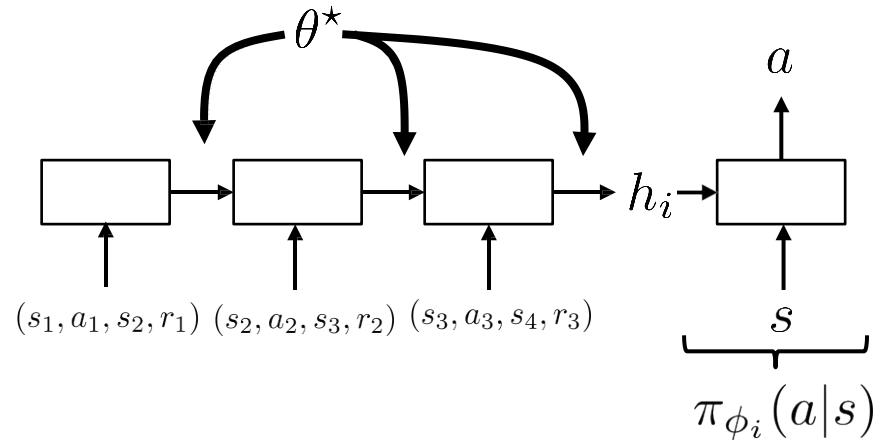
Method	Small Maze		Large Maze	
	Episode 1	Episode 2	Episode 1	Episode 2
Random	188.6 ± 3.5	187.7 ± 3.5	420.2 ± 1.2	420.8 ± 1.2
LSTM	52.4 ± 1.3	39.1 ± 0.9	180.1 ± 6.0	150.6 ± 5.9
SNAIL (ours)	50.3 ± 0.3	34.8 ± 0.2	140.5 ± 4.2	105.9 ± 2.4

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

Meta-RL with recurrent policies

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

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

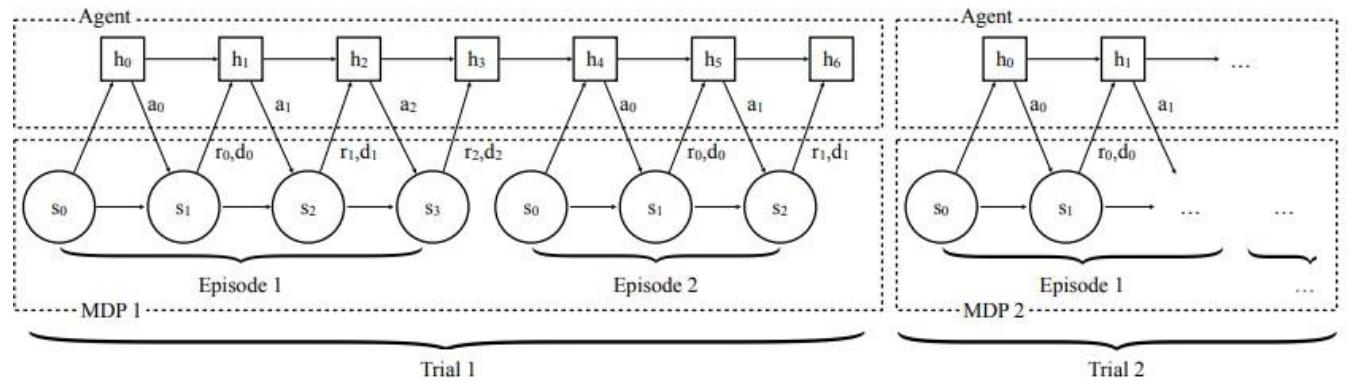


Heess, Hunt, Lillicrap, Silver. **Memory-based control with recurrent neural networks.** 2015.

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

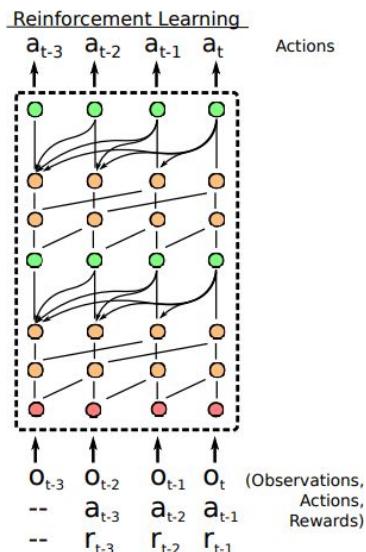
Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. **RL2: Fast Reinforcement Learning via Slow Reinforcement Learning.** 2016.

Architectures for meta-RL



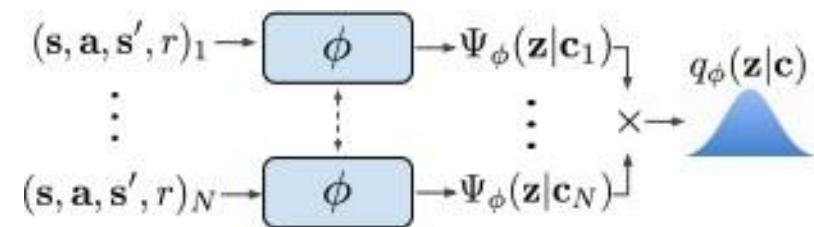
standard RNN (LSTM) architecture

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.**

Challenges and Open Problems in RL

What's the problem?

Challenges with **core algorithms**:

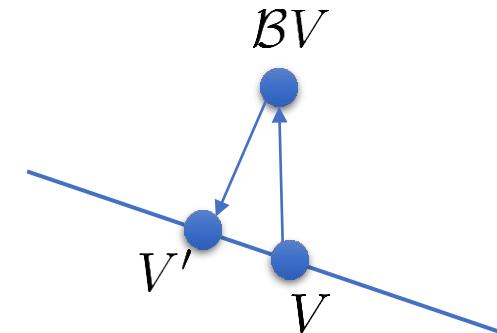
- Stability: does your algorithm converge?
- Efficiency: how long does it take to converge? (how many samples)
- Generalization: after it converges, does it generalize?

Challenges with **assumptions**:

- Is this even the right problem formulation?
- What is the source of *supervision*?

Stability and hyperparameter tuning

- Devising stable RL algorithms is very hard
- Q-learning/value function estimation
 - Fitted Q/fitted value methods with deep network function estimators are typically not contractions, hence no guarantee of convergence
 - Lots of parameters for stability: target network delay, replay buffer size, clipping, sensitivity to learning rates, etc.
- Policy gradient/likelihood ratio/REINFORCE
 - Very high variance gradient estimator
 - Lots of samples, complex baselines, etc.
 - Parameters: batch size, learning rate, design of baseline
- Model-based RL algorithms
 - Model class and fitting method
 - Optimizing policy w.r.t. model non-trivial due to backpropagation through time
 - More subtle issue: policy tends to *exploit* the model

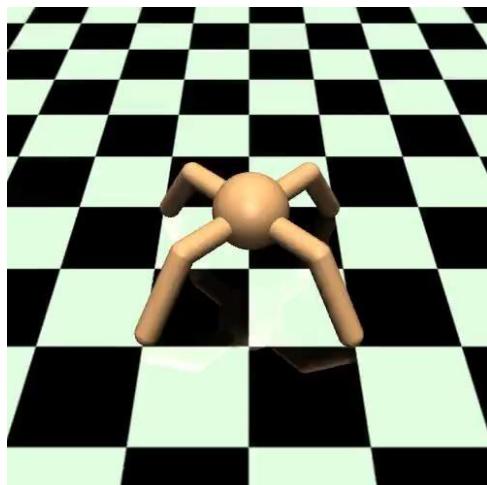


The challenge with sample complexity

- Need to wait for a long time for your homework to finish running
- Real-world learning becomes difficult or impractical
- Precludes the use of expensive, high-fidelity simulators
- Limits applicability to real-world problems



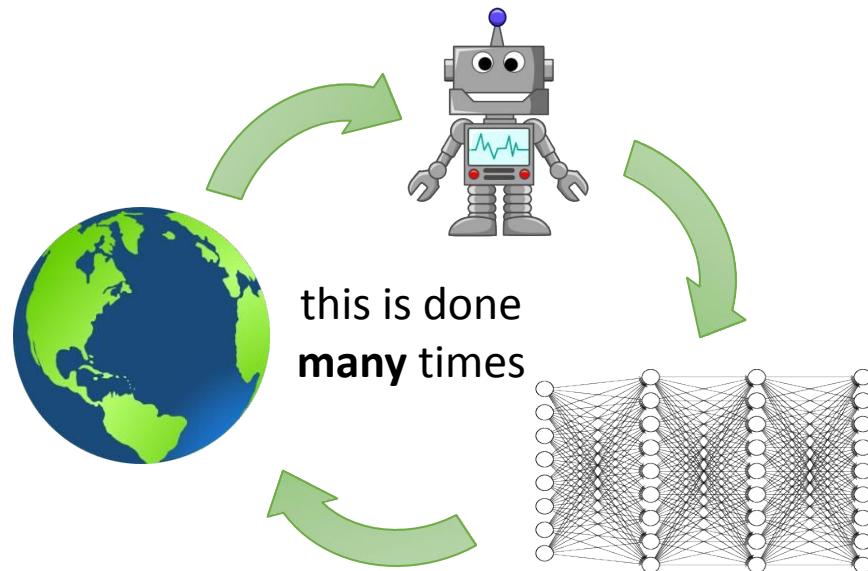
Scaling up deep RL & generalization



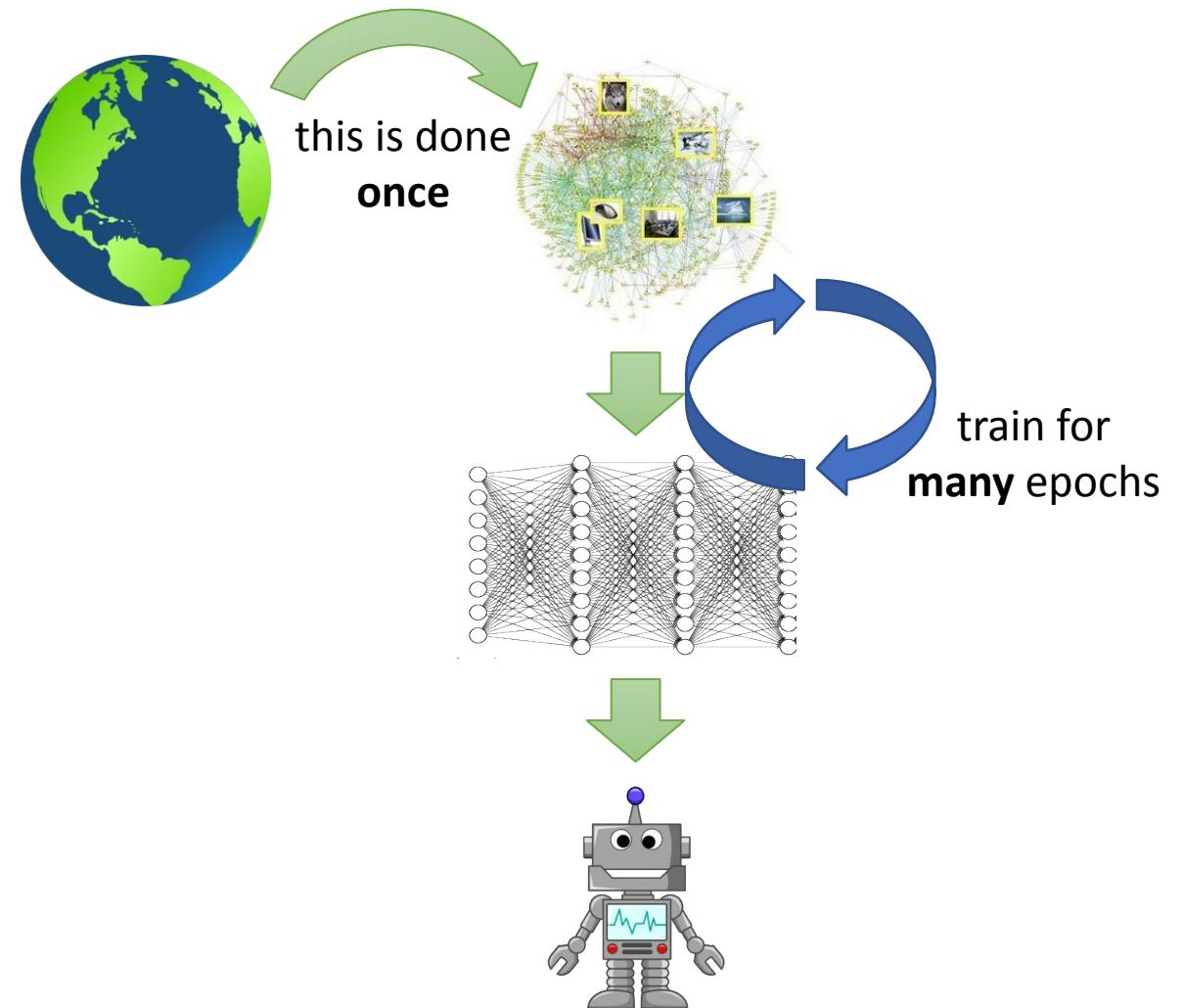
- Large-scale
 - Emphasizes diversity
 - Evaluated on generalization
-
- Small-scale
 - Emphasizes mastery
 - Evaluated on performance
 - Where is the generalization?

RL has a big problem

reinforcement learning

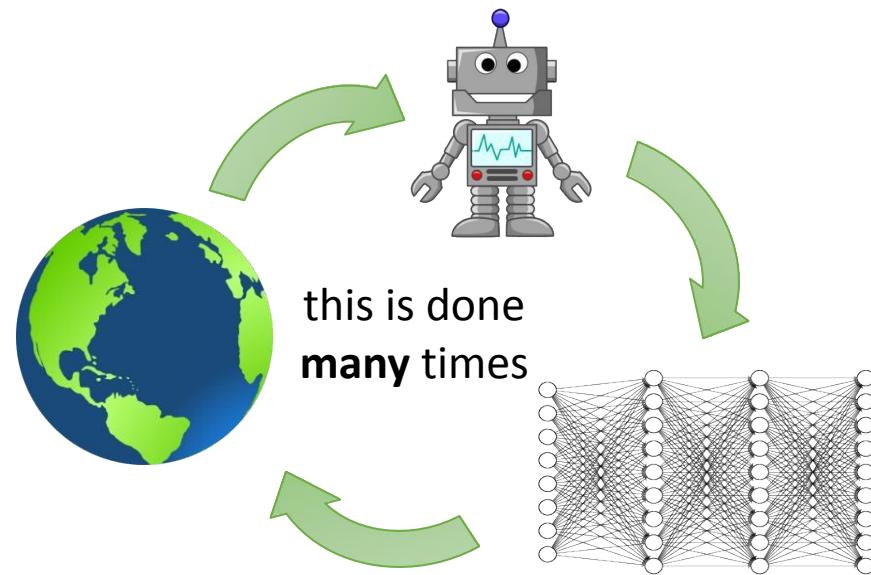


supervised machine learning

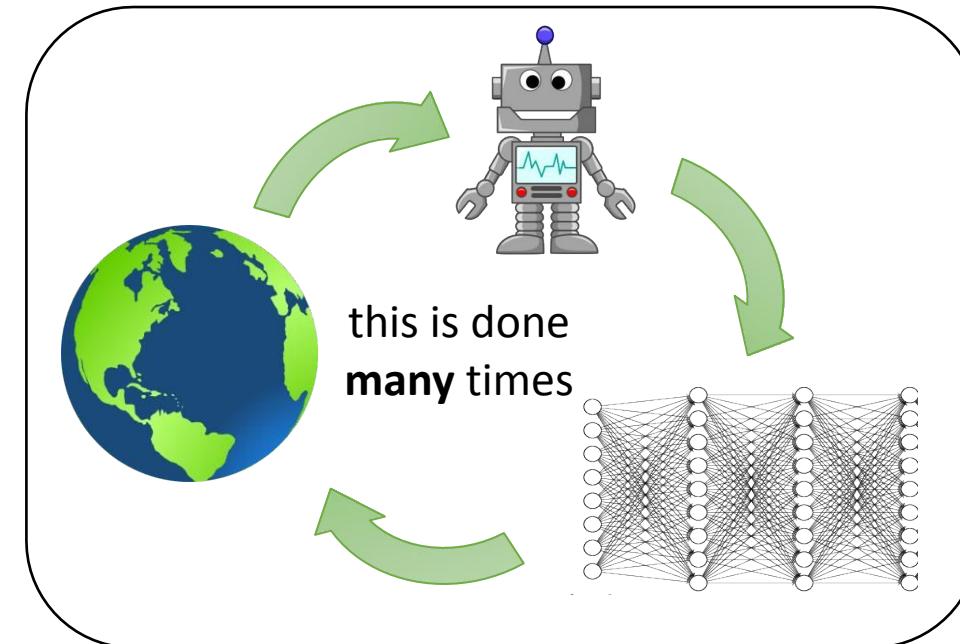


RL has a big problem

reinforcement learning

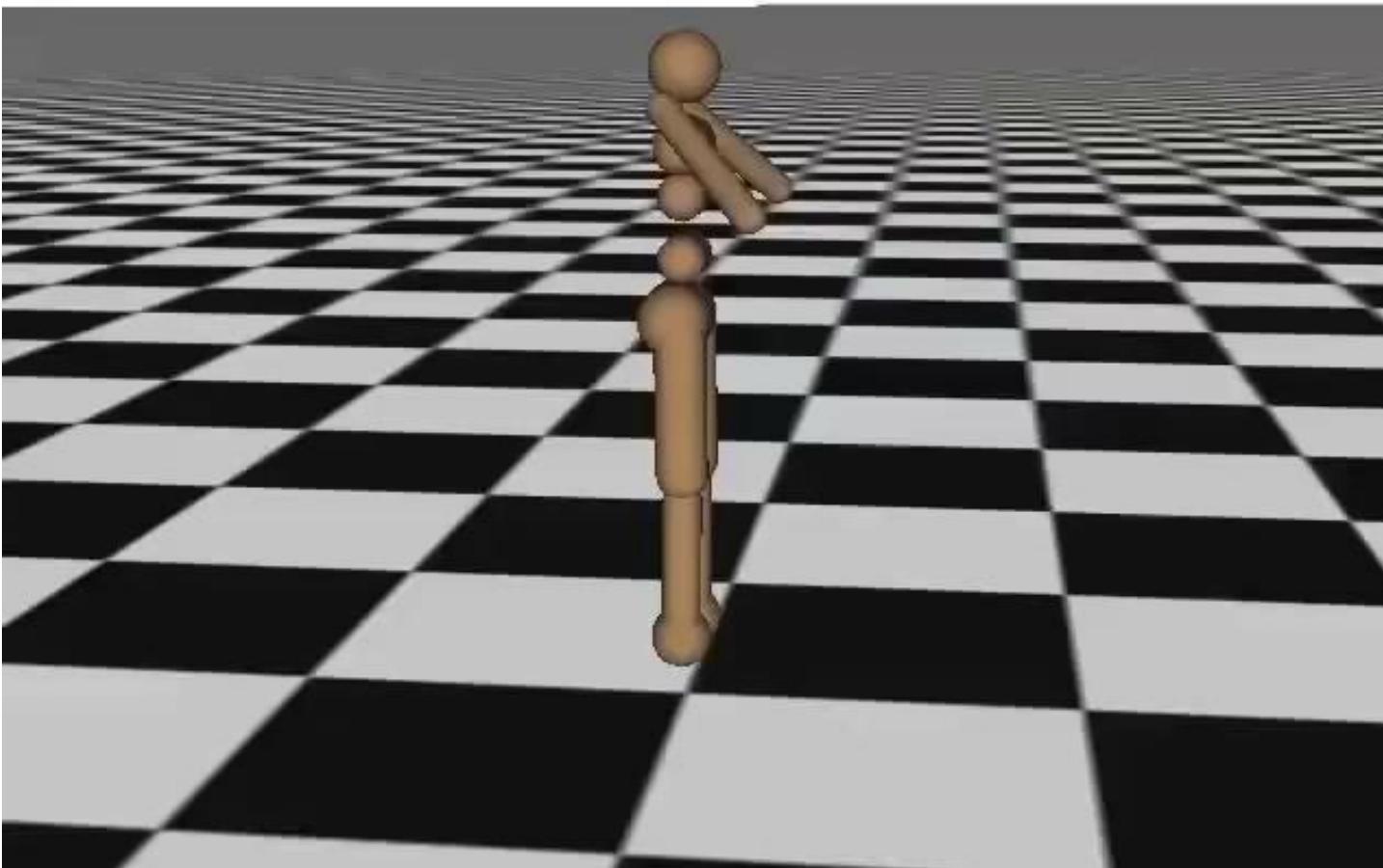


actual reinforcement learning



How bad is it?

Iteration 0



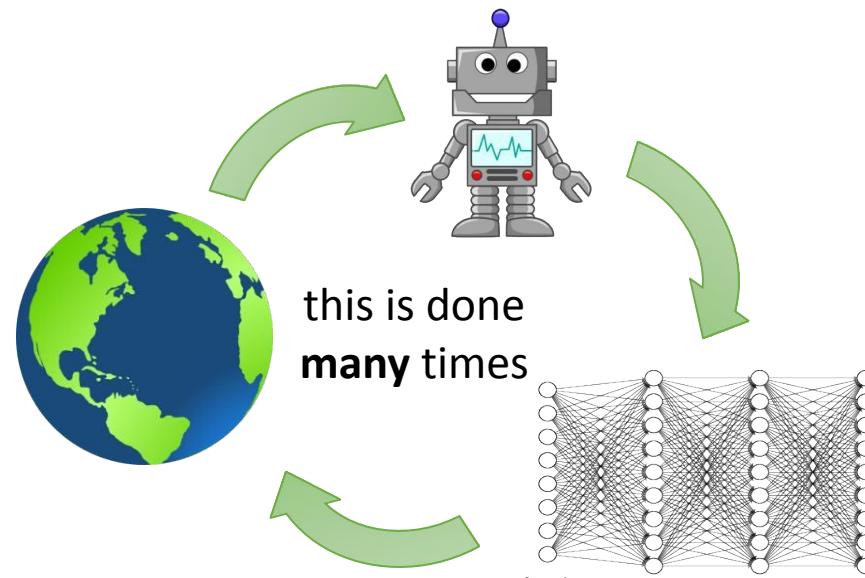
- This is quite cool
- It takes 6 days of real time (if it was real time)
- ...to run on an infinite flat plane



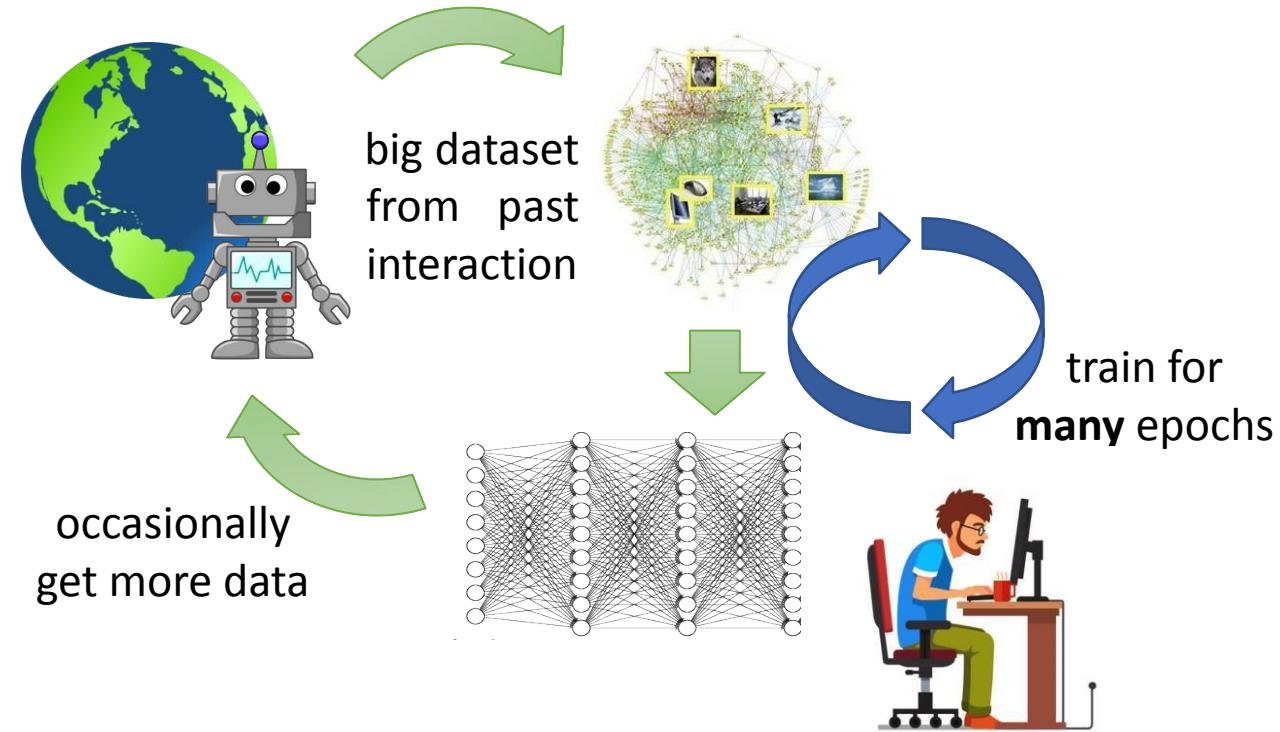
The real world is not so simple!

Off-policy RL?

reinforcement learning



off-policy reinforcement learning



Single task or multi-task?

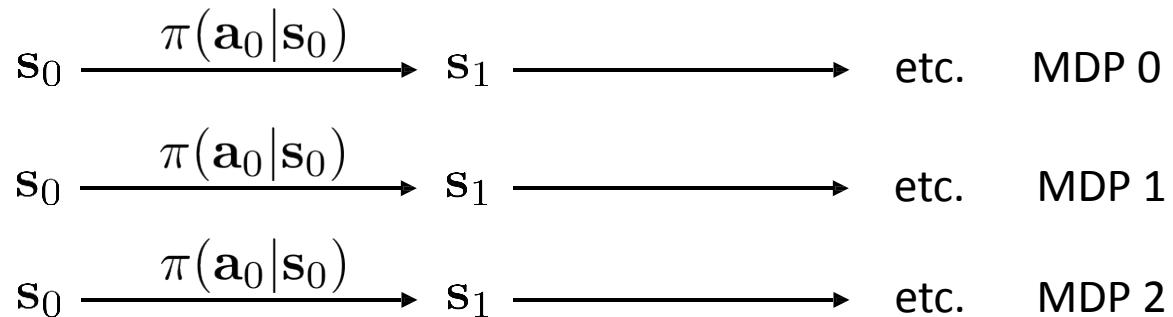
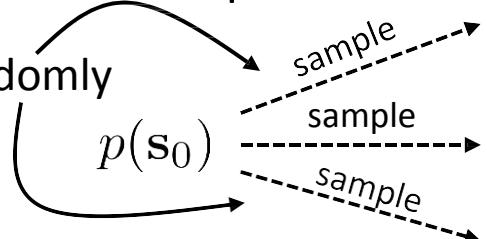


this is where generalization can come from...

maybe doesn't require any new assumption, but might merit additional treatment

The real world is not so simple!

pick MDP randomly
in first state



Where does the supervision come from?

- If you want to learn from many different tasks, you need to get those tasks somewhere!
- Learn objectives/rewards from demonstration (inverse reinforcement learning)
- Generate objectives automatically?



reinforcement learning agent



what is the **reward**?

Rethinking the Problem Formulation

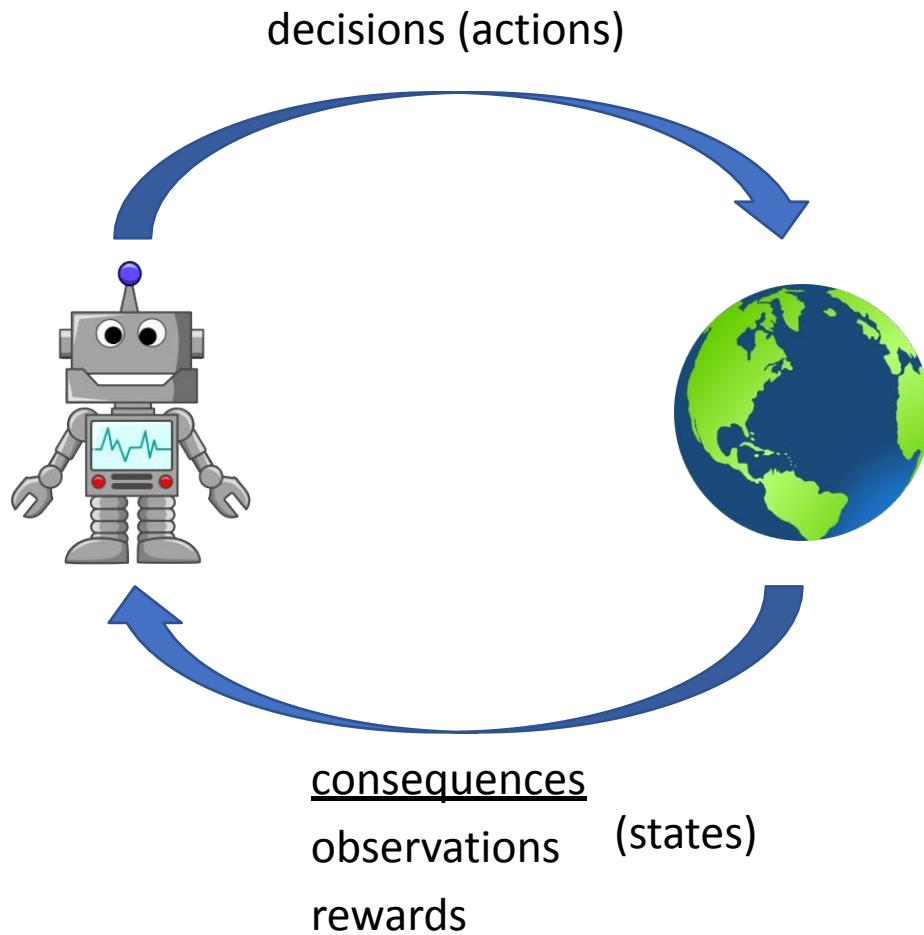
- How should we define a *control* problem?
 - What is the data?
 - What is the goal?
 - What is the supervision?
 - may not be the same as the goal...
- Think about the assumptions that fit your problem setting!
- Don't assume that the basic RL problem is set in stone

Some perspectives...

Reinforcement Learning as an Engineering Tool
Reinforcement Learning and the Real World
Reinforcement Learning as “Universal”
Learning

Reinforcement Learning as an Engineering Tool

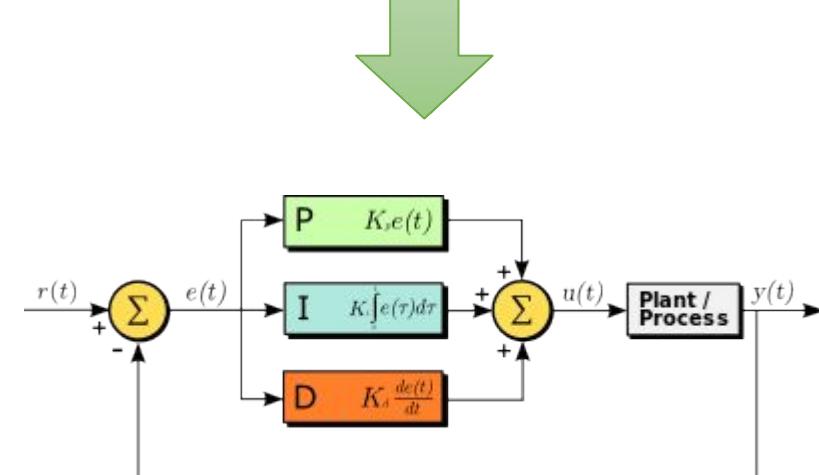
What we think RL is...



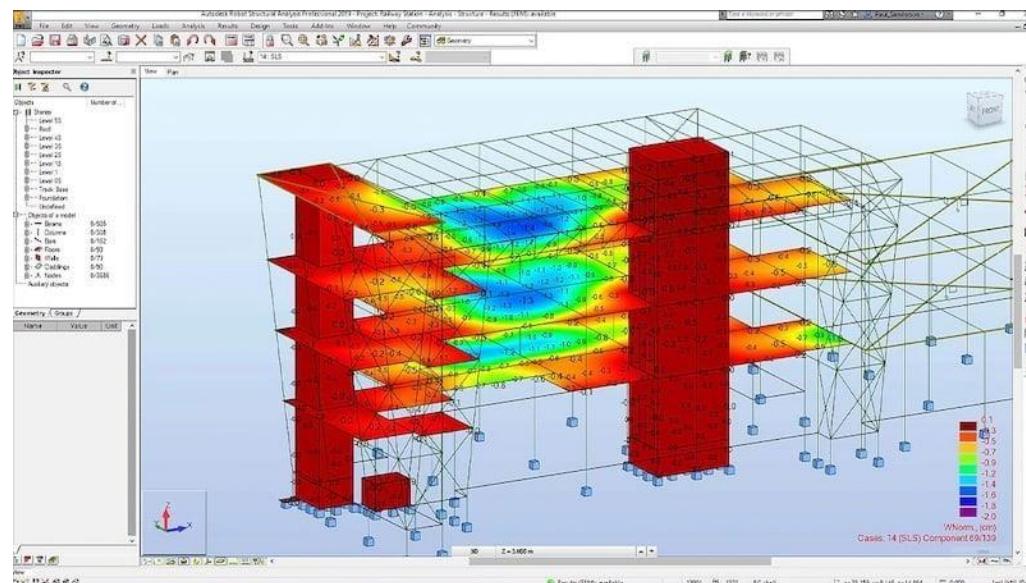
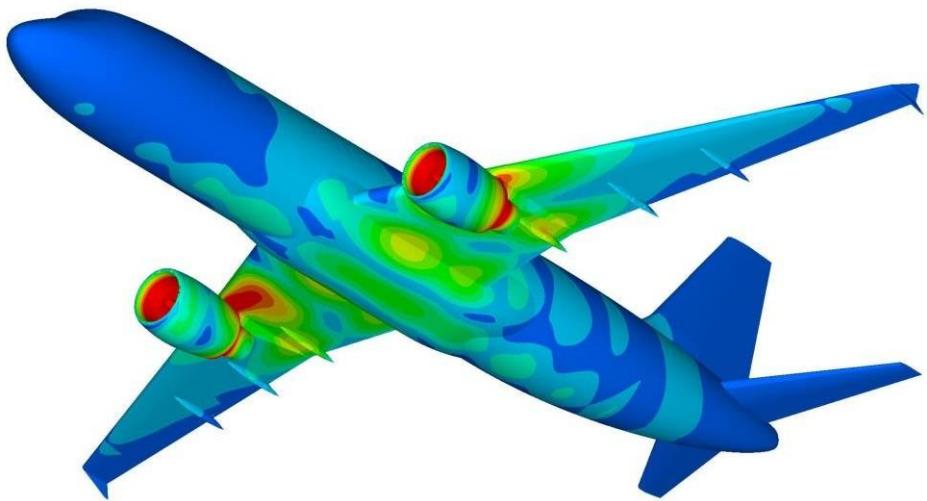
Engineering a control system



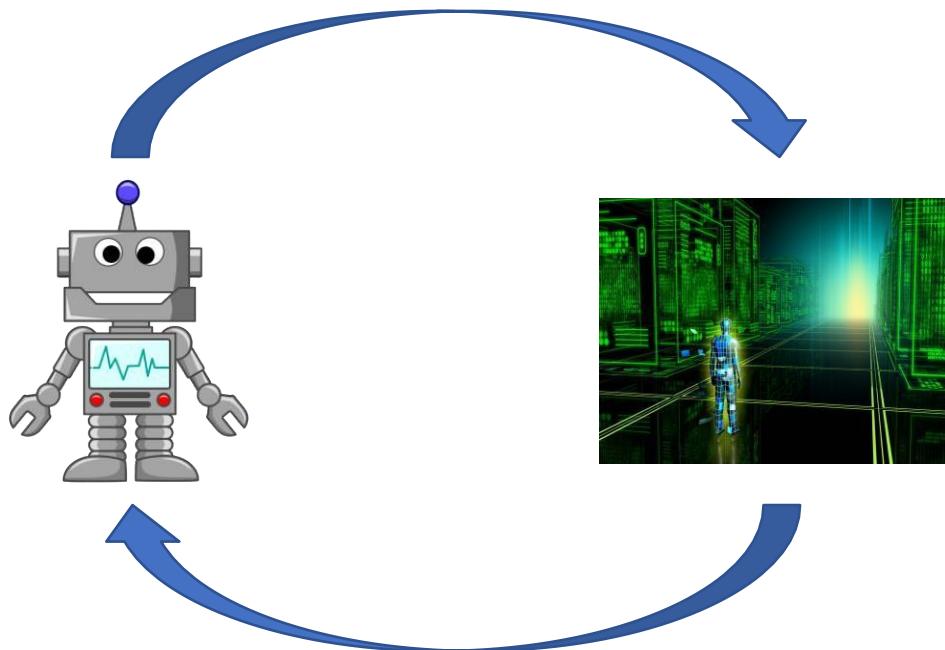
$$\begin{aligned}\mathbf{r} &= \mathbf{r}(t) = r\hat{\mathbf{e}}_r \\ \mathbf{v} &= v\hat{\mathbf{e}}_r + r \frac{d\theta}{dt}\hat{\mathbf{e}}_\theta + r \frac{d\varphi}{dt} \sin \theta \hat{\mathbf{e}}_\varphi \\ \mathbf{a} &= \left(a - r \left(\frac{d\theta}{dt} \right)^2 - r \left(\frac{d\varphi}{dt} \right)^2 \sin^2 \theta \right) \hat{\mathbf{e}}_r \\ &\quad + \left(r \frac{d^2\theta}{dt^2} + 2v \frac{d\theta}{dt} - r \left(\frac{d\varphi}{dt} \right)^2 \sin \theta \cos \theta \right) \hat{\mathbf{e}}_\theta \\ &\quad + \left(r \frac{d^2\varphi}{dt^2} \sin \theta + 2v \frac{d\varphi}{dt} \sin \theta + 2r \frac{d\theta}{dt} \frac{d\varphi}{dt} \cos \theta \right) \hat{\mathbf{e}}_\varphi\end{aligned}$$



Characterization and simulation...



RL: anything you can simulate you can control



- Provides a powerful engineering tool
- Now *that* different from conventional engineering approach!
 - Before: characterize, simulate, control
 - Now: characterize, simulate, run RL
- Main role: powerful *inversion* engine
- Main weakness: still need to simulate!

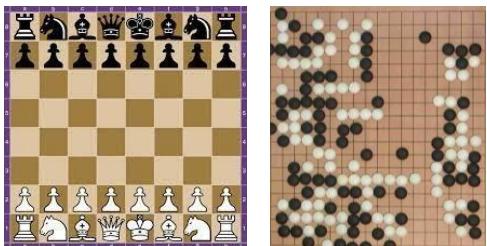
Reinforcement Learning and the Real World



Moravec's paradox

Moravec's paradox seems like a statement about AI
but it is actually a statement about the physical universe

“easy” universes



Why?

“hard” universes



We are all prodigious olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it.

- Hans Moravec

The main lesson of thirty-five years of AI research is that the hard problems are easy and the easy problems are hard. The mental abilities of a four-year-old that we take for granted – recognizing a face, lifting a pencil, walking across a room, answering a question – in fact solve some of the hardest engineering problems ever conceived.

- Steven Pinker

What does this have to do with RL?



How do we engineer a system that can deal with the unexpected?

- Minimal external supervision about what to do
- Unexpected situations that require adaptation
- Must discover solutions autonomously
- Must “stay alive” long enough to discover them!

- Humans are extremely good at this
- Current AI systems are extremely bad at this
- RL *in principle* can do this, and nothing else can

So what's the problem?



RL **should** be really good in the
“hard” universes!

□ RL *in principle* can do this, and nothing else can

But we rarely study this kind of setting in RL research!

“easy” universes

success = high reward
 (“optimal control”)

closed world, rules
are known

lots of simulation

Main question: can RL
algorithms **optimize**
really well

“hard” universes

success = “survival”
 (“good enough control”)

open world, everything
must come from data

no simulation (because
rules are unknown)

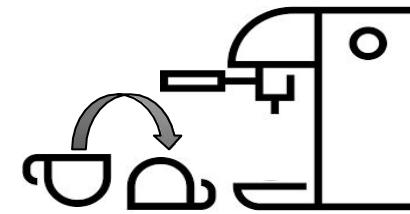
Main question: can RL
generalize and adapt

Some questions that come up in the real world

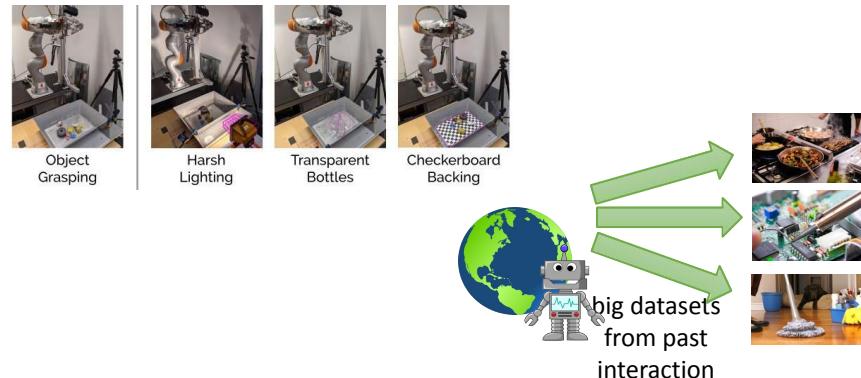
How do we tell RL agents **what we want them to do?**



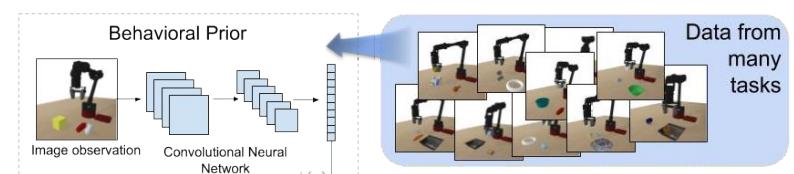
How can we learn **fully autonomously** in continual environments?



How do remain **robust** as the environment changes around us?



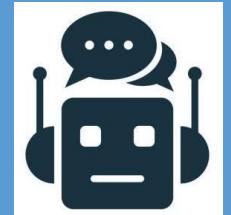
What is the right way to **generalize** using **experience & prior data**?



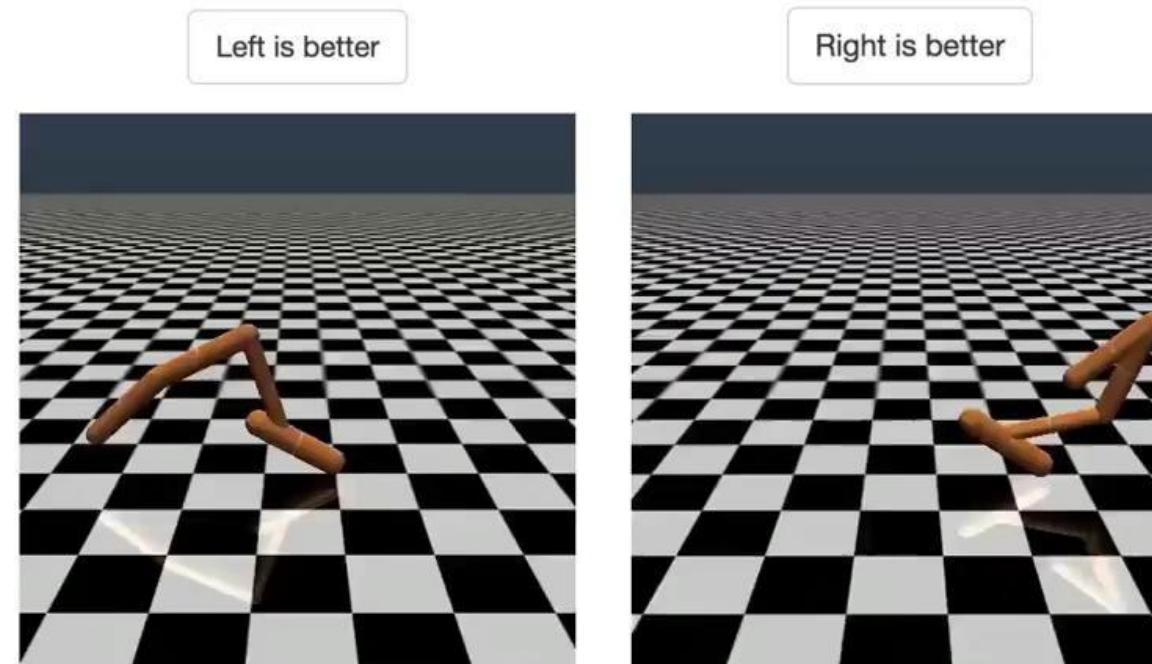
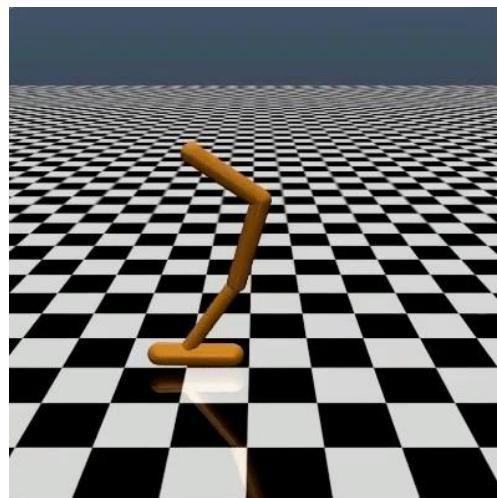
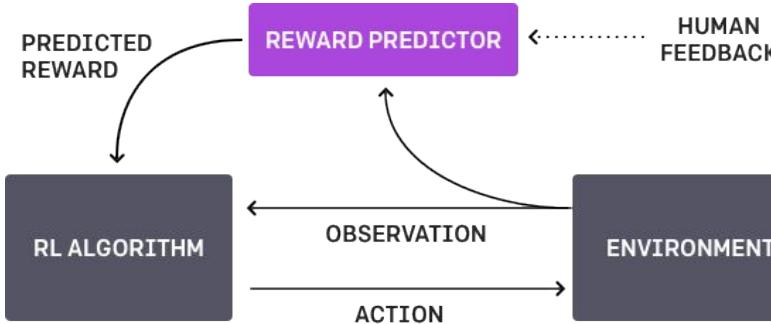
What is the right way to **bootstrap exploration** with **prior experience**?

This is not about robots

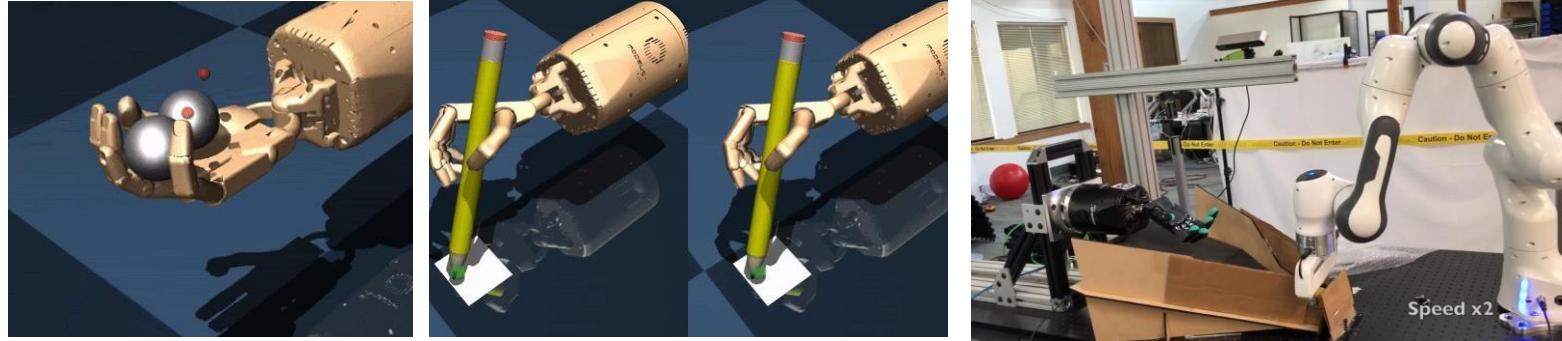
Robots are the most natural for us to think about, because they are embodied like we are



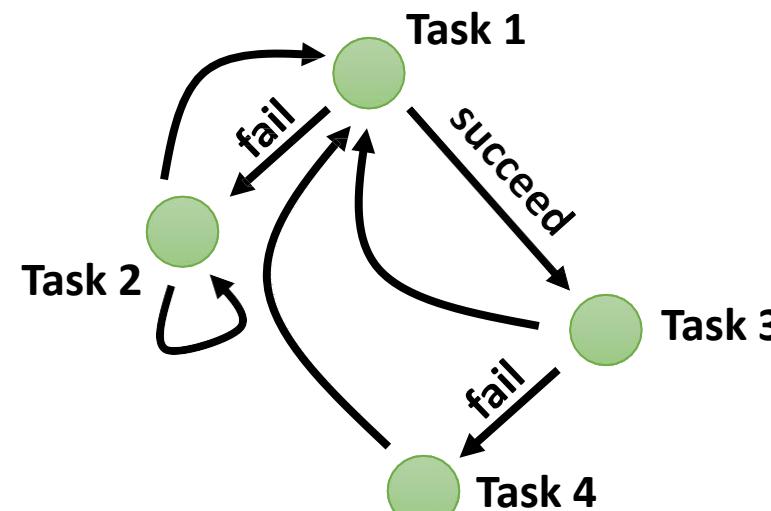
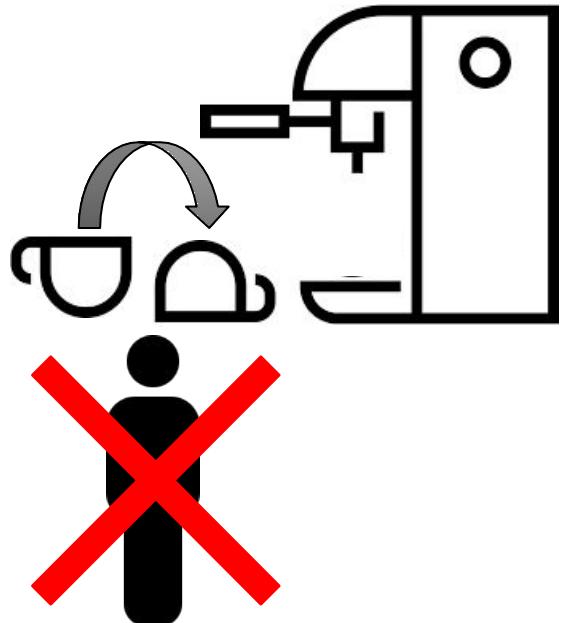
Other ways to communicate objectives?



How can we learn fully autonomously?



Nagabandi, Konolige, Levine, Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. CoRL 2019.



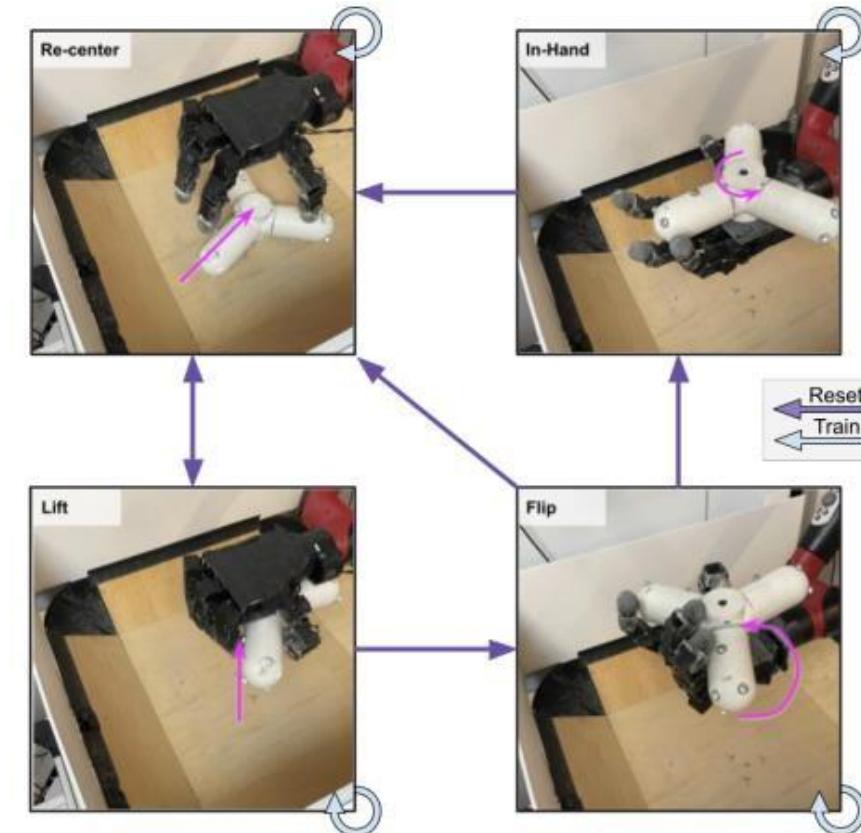
Task 1: put cup in coffee machine

Task 2: pick up cup

Task 3: replace cup

Task 4: clean up spill from cup...

How can we learn fully autonomously?

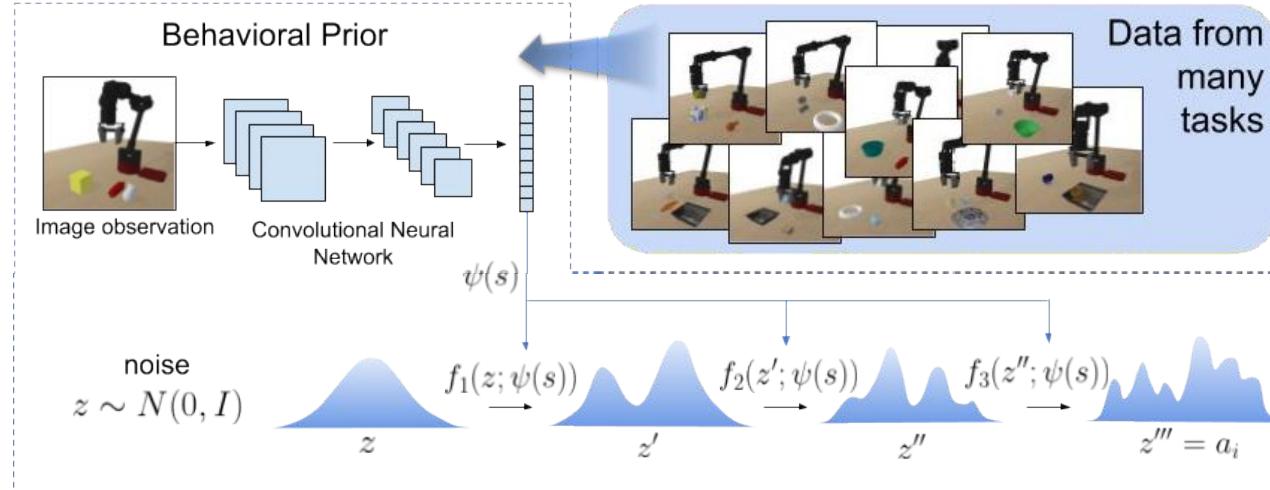
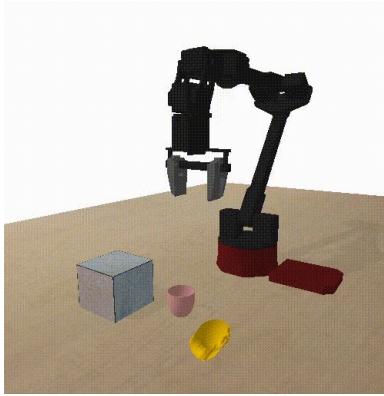


Gupta, Yu, Zhao, Kumar, Rovinsky, Xu, Devlin, Levine. **Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention.** 2021.

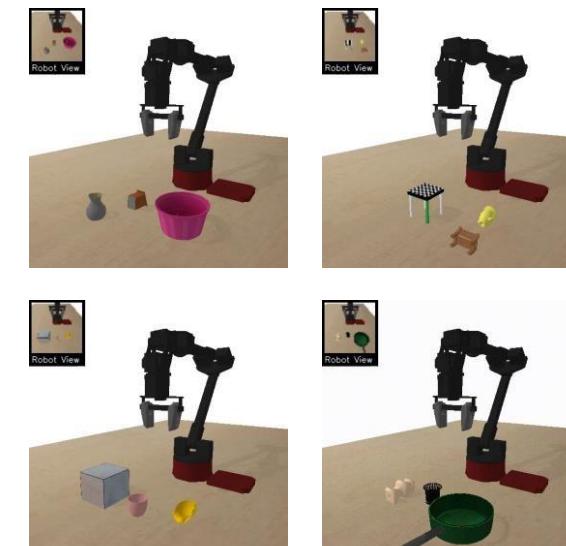
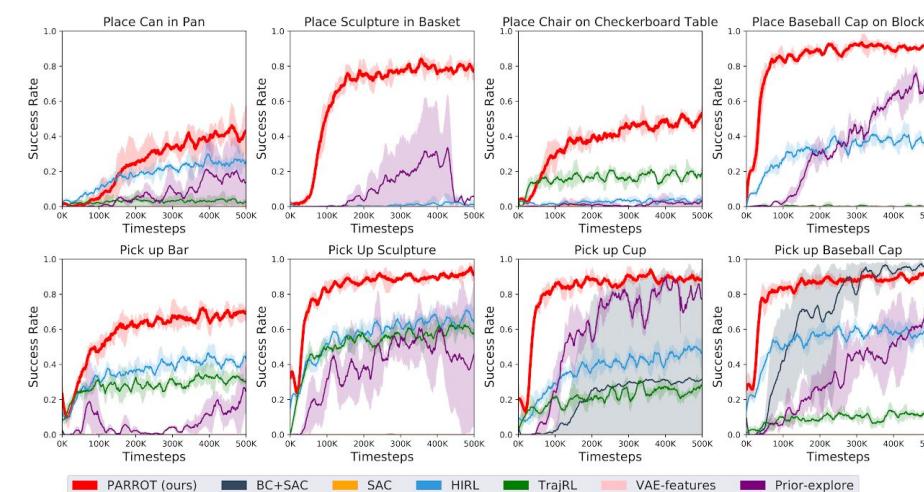
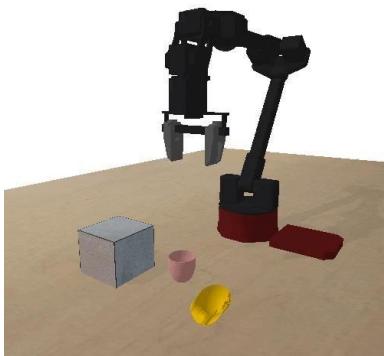
33

How to bootstrap exploration from experience?

exploring from scratch



exploring from behavioral prior



This all seems really hard, what's the point?



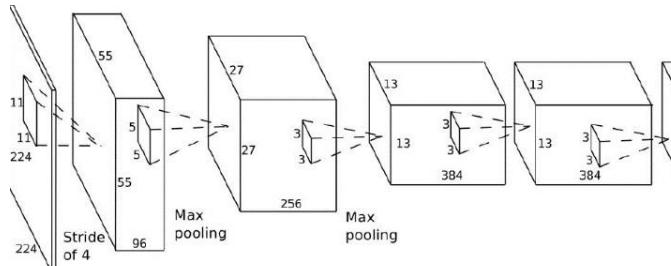
Why is this interesting?

- To see interesting emergent behavior, we must train our systems in environments that actually require interesting emergent behavior!
- RL in the real world may be difficult, but it is also rewarding
- It's exciting to see what solutions intelligent agents come up with
- Most exciting if they come up with something we don't expect
- This requires the world they inhabit to admit novel solutions
- This means that world must be complicated enough!

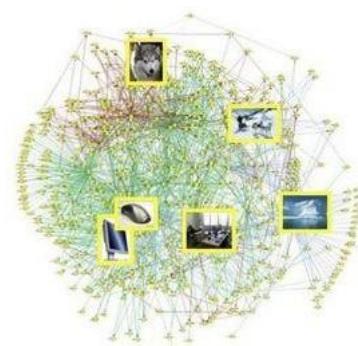
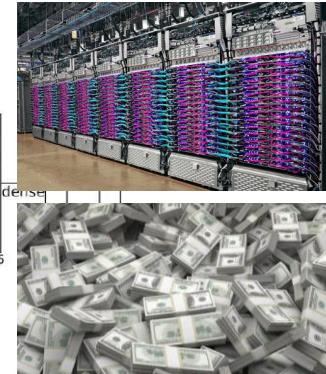
Reinforcement Learning as “Universal” Learning

Large-scale machine learning

Why does deep learning work?

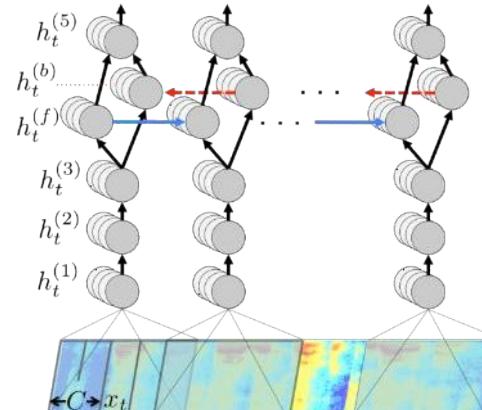
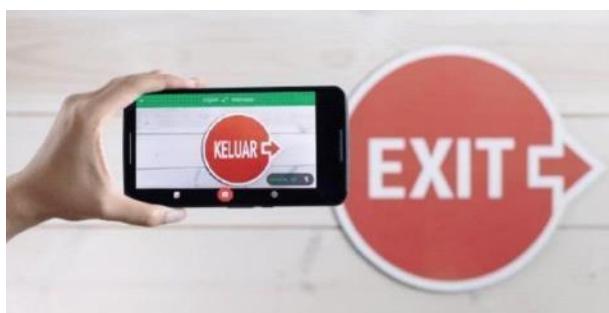
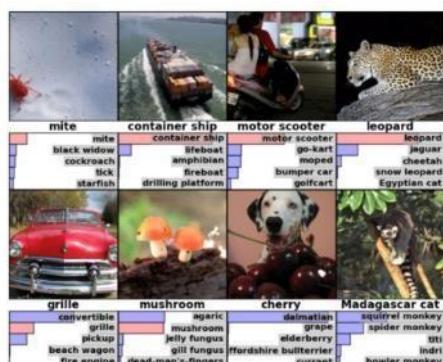
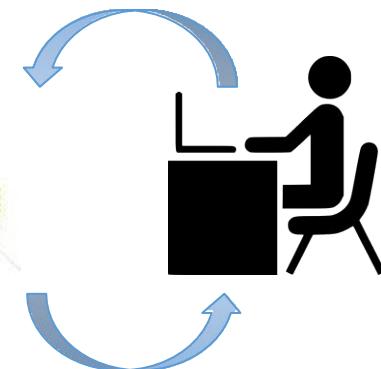


big models

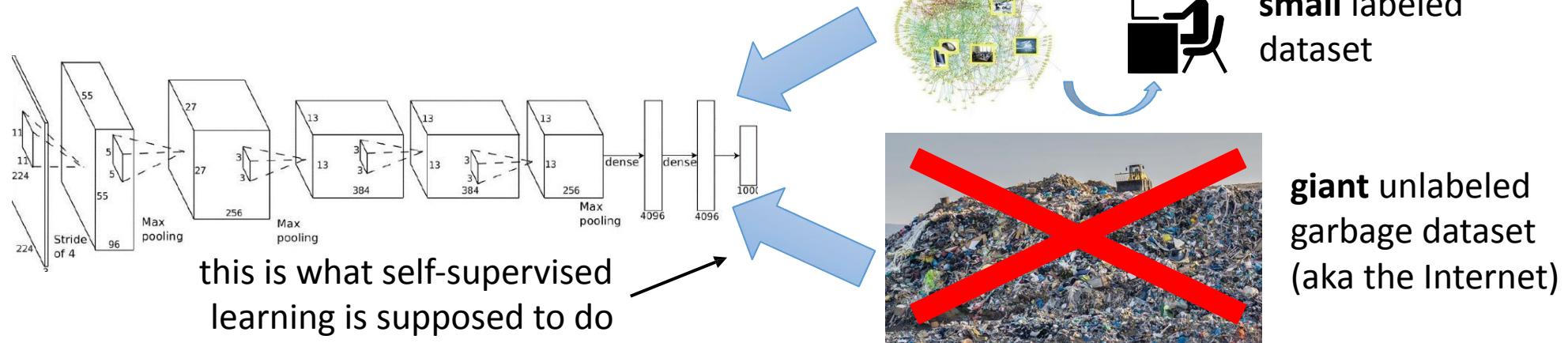


big datasets

labeled



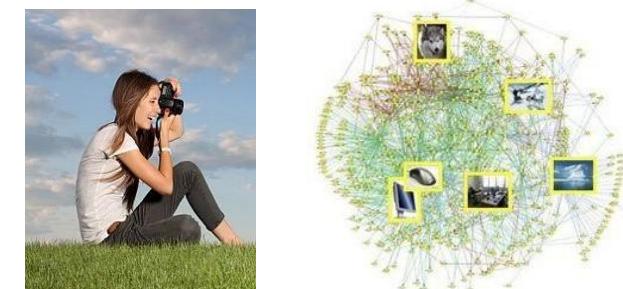
Reducing the supervision burden



But then where does the knowledge come from?

“Classic” unsupervised learning: $p_\theta(\mathbf{x})$

(this is what, for example, large language models do)



Aside: perhaps this is why “prompting” large language models is such an art!

Stepping back a bit...



Why do we need **machine learning** anyway?

A postulate

Machine learning for one reason and one reason only – that's to produce adaptable and complex decisions.



Decision: how do I move my joints?



Decision: how do I steer the car?



What is the decision? The image label?

Usually not!

What happens with that label **afterwards**?

Is it used to tag a user's photo?

Detect an endangered animal in a camera trap?

Aside: why do we need **brains** anyway?



Daniel Wolpert

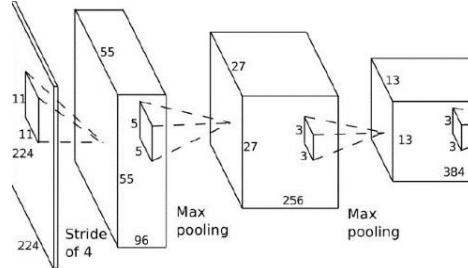
(knows quite a lot about brains)



"We have a brain for one reason and one reason only -- that's to produce **adaptable and complex movements**. Movement is the only way we have affecting the world around us... I believe that to understand movement is to understand the whole brain."

Reinforcement learning as a way to use “cheap” (previously collected) data

- this tells us what the task is (e.g., reward function)



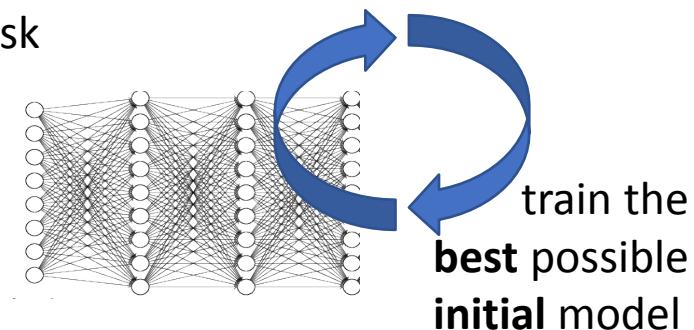
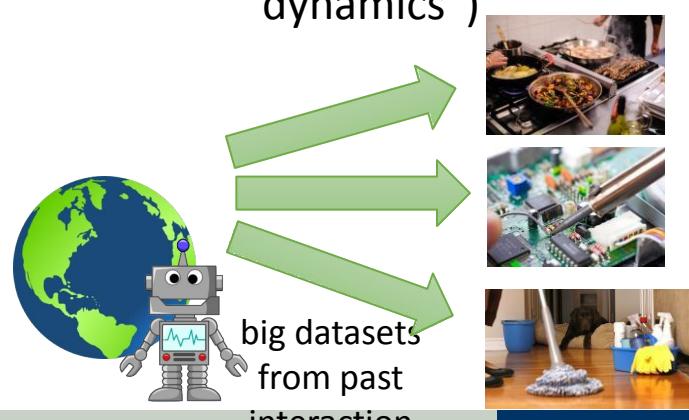
this tells us **how the world works** (i.e., “dynamics”)



limited amount of supervision

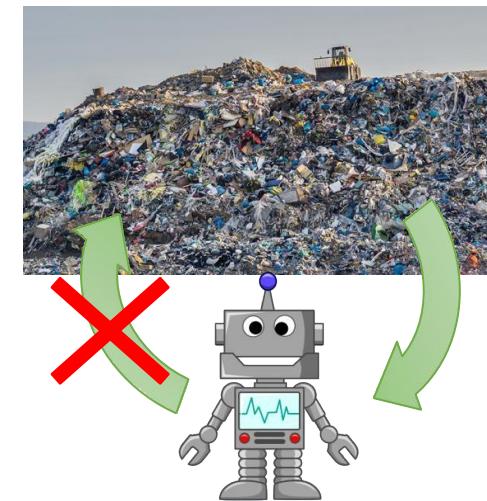
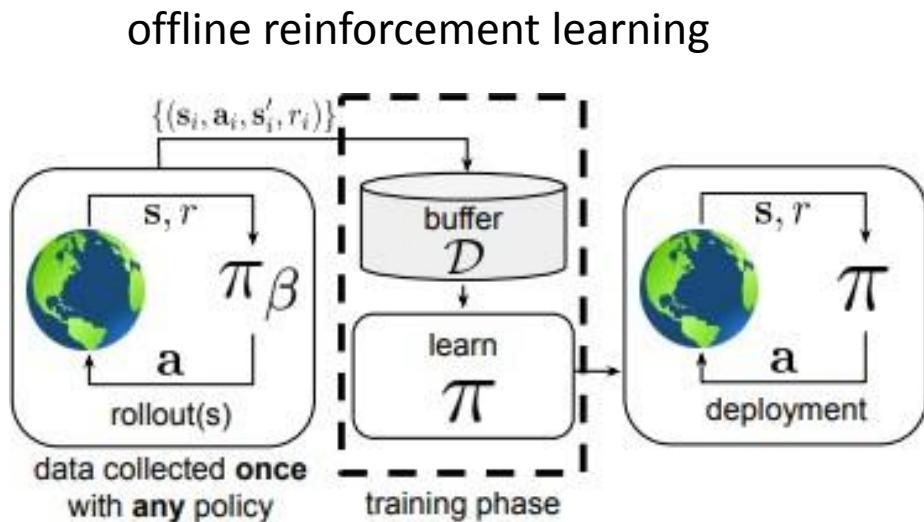


Large, Cheap dataset of previously collected data



train the **best** possible **initial** model

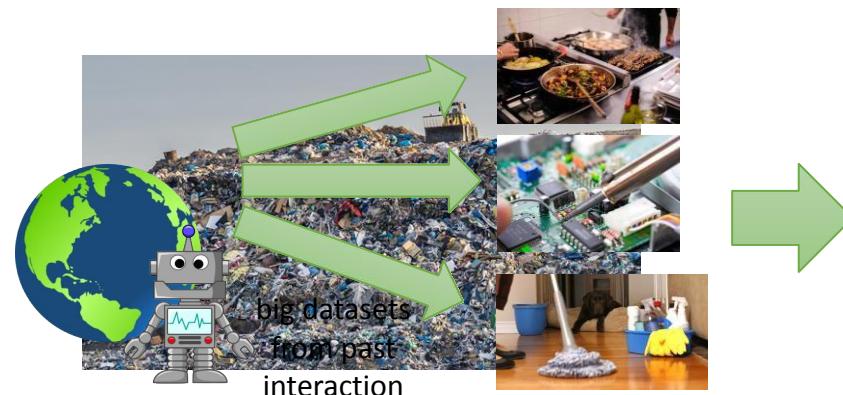
The RL + data problem



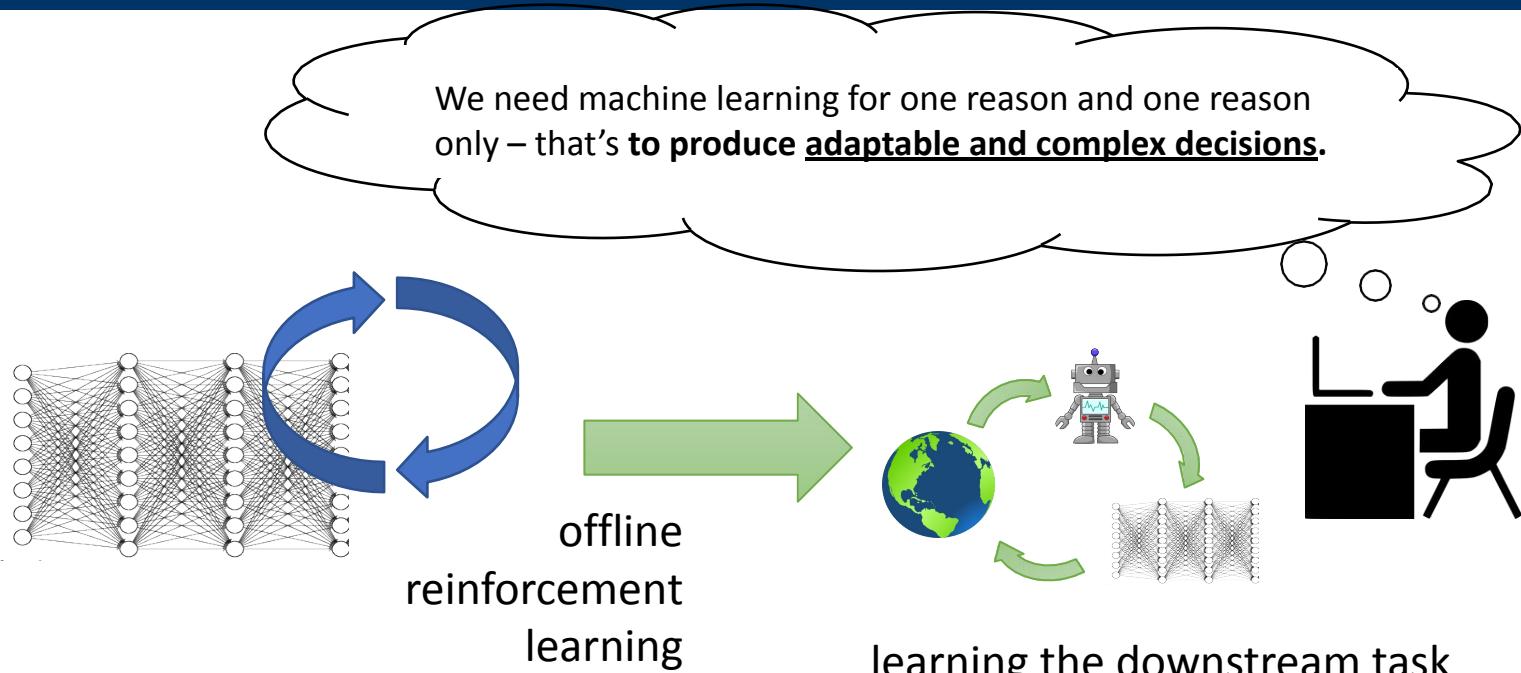
with naïve RL, this is a **costly** interactive process if done in the real world!

but self-supervised learning is about using **cheap** data that we already have lying around (in the garbage heap)!

The recipe



large dataset of diverse
(but possibly low-quality)
behavior



there are a few different
choices here:

- human-defined skills
- goal-conditioned RL
- self-supervised skill discovery

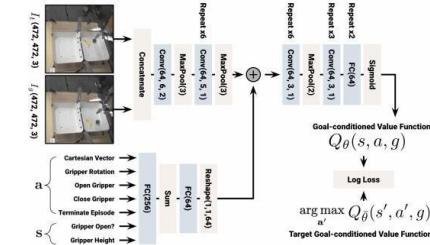
learning the downstream task

Can we learn from offline data without well-defined tasks?

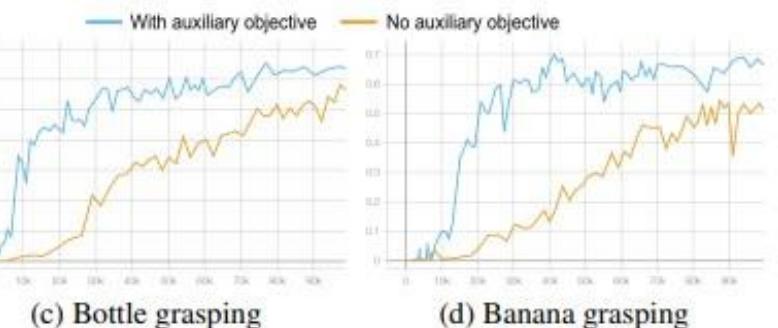


- No reward function at all, task is defined entirely using a **goal image**
- Uses a conservative offline RL method designed for goal-reaching, based on CQL
- Works very well as an **unsupervised pretraining objective**

1. Train **goal-conditioned Q**-function with offline RL



2. Finetune with a **task reward** and limited data



Can offline RL train large language models?

What we want:



Can you teach me about behavior cloning?



Of course! I'd be happy to explain behavior cloning. To start, could you tell me if you've ever come across the terms "artificial intelligence" or "machine learning" before?



Yes, I've heard those terms, but I'm not exactly sure what they mean.



No problem at all! Let's take it step by step. Have you ever used a computer or a smartphone?

What we get:



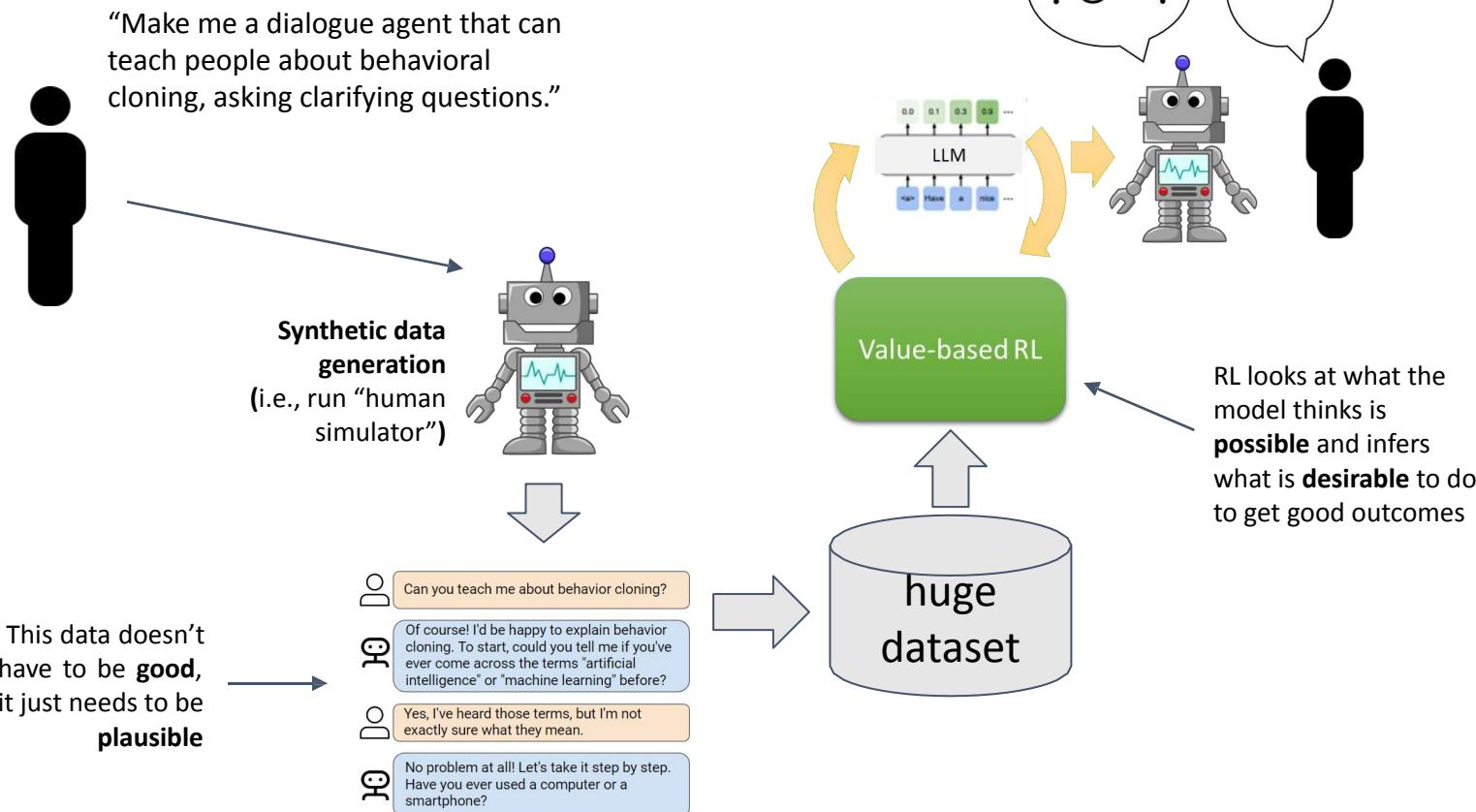
J Can you explain the AI concept of behavior cloning to me? But first, I want you to ask me questions to figure out what my background in AI is, then tailor your explanation to match my level.



Certainly! Let's start by understanding your background in AI to provide an explanation that matches your level of knowledge. Please answer the following questions:

1. Have you ever studied or worked with artificial intelligence or machine learning concepts before, even at a basic level?
2. Are you familiar with terms like "neural networks," "training data," or "supervised learning"?
3. Have you ever heard of or worked with specific AI tools or frameworks like TensorFlow, PyTorch, or scikit-learn?
4. Are you interested in a high-level overview or a more detailed explanation of behavior cloning?
5. Do you have any specific examples or applications in mind where you'd like to see behavior cloning?

Model-based offline RL for multi-turn dialogue



RL + LLM is great for tasks that are easy for LLMs to **simulate** but hard for them to **perform optimally**.

With RL, instead of emulating humans, LLMs can learn to achieve desired outcomes using their understanding of how humans behave.

Example: model-based RL for multi-turn dialogue



GPT Agent Dialogue:

H: Hi, I've been hearing about something called "behavior cloning." I don't really understand what it is. Can you explain it to me?

AI: Certainly! Let's start by understanding your background. Have you ever used a computer or worked with artificial intelligence or machine learning before? Are you familiar with terms like "neural networks," "training data," or "learning from experience"? If so, you might be interested in a high-level overview or a more detailed explanation.

H: I have some basic knowledge of artificial intelligence and machine learning. I'm an expert. I am familiar with neural networks and supervised learning, and I have experience with training data.

AI: Great, thank you for that information. Behavior cloning is a type of reinforcement learning, which is a subcategory of machine learning. In reinforcement learning, an agent learns to make optimal decisions by interacting with its environment and receiving feedback in the form of rewards or punishments. Behavior cloning, on the other hand, allows a computer to imitate or "clone" the behavior of an already established agent, such as a human expert or a pre-existing system. Does that make sense so far?

H: Yes, that's starting to make sense. What do you mean by "expert"?

AI: An expert, in this context, refers to a human or an advanced AI model that has learned how to solve a particular task or problem. The AI model then uses this knowledge to learn how to perform the task itself. Does that clarify things?

H: Yes, thank you.

IE-RL Agent Dialogue:

H: Hi there! I've heard about this thing called "behavior cloning" in AI, but I don't really understand what it is. Can you explain it to me?

AI: Of course! I'd be happy to explain behavior cloning. To start, could you tell me if you've ever come across the terms "artificial intelligence" or "machine learning" before?

H: Yes, I've heard those terms, but I'm not exactly sure what they mean.

AI: No problem at all! Let's take it step by step. Have you ever used a computer or a smartphone?

H: Yes, I use my computer and phone all the time.

AI: Great! So, imagine that a computer or a smartphone could learn to do something by watching you do it. That's a bit like what behavior cloning is.

H: Oh, interesting. Can you explain more about how that works?

AI: Sure. Behavior cloning is a type of machine learning algorithm where a computer learns to perform a task by imitating a human's behavior. It does this by closely analyzing and mimicking the actions of a human expert, using a large dataset of examples. This approach is often used in tasks that don't have a clear set of rules or can't be easily solved through traditional programming methods. Does that make sense?

H: Yes, I think so. So how does the computer learn to mimic the behavior?

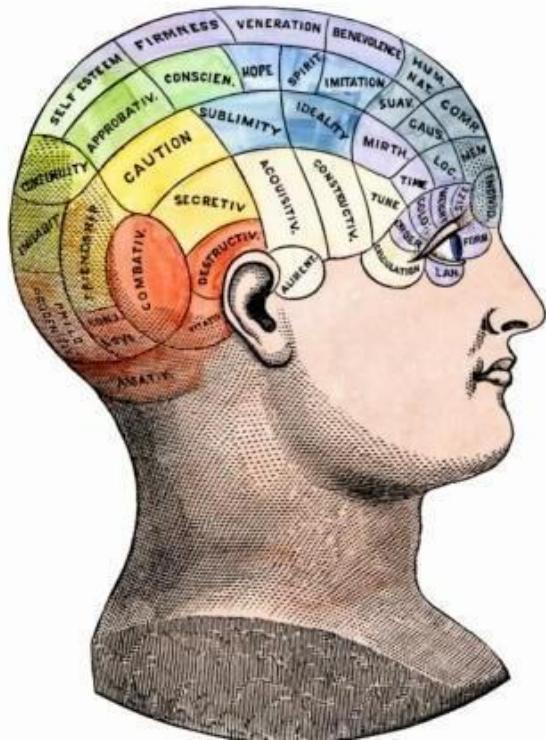
AI: That's a great question. The computer is fed a large amount of data that shows different examples of the task being performed by a human expert. The computer then uses this data to identify patterns and develop a model of the behavior. This model allows the computer to predict the correct actions to take in a given situation. Over time, the computer can continue to refine its model and improve its performance. Does that help clarify things?

H: That does. Thank you!

Back to the Bigger Picture



Learning as the basis of intelligence



- Reinforcement learning = can reason about decision making
- Deep models = allows RL algorithms to learn and represent complex input-output mappings

Deep models are what allow reinforcement learning algorithms to solve complex problems end to end!

How Much Information Does the Machine Need to Predict?

Y LeCun

- "Pure" Reinforcement Learning (cherry)
 - ▶ The machine predicts a scalar reward given once in a while.
 - ▶ **A few bits for some samples**
- Supervised Learning (icing)
 - ▶ The machine predicts a category or a few numbers for each input
 - ▶ Predicting human-supplied data
 - ▶ **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
 - ▶ The machine predicts any part of its input for any observed part.
 - ▶ Predicts future frames in videos
 - ▶ **Millions of bits per sample**
- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)



These slides have been adapted from

- Chelsea Finn, [CS224R: Deep Reinforcement Learning, Stanford](#)
- Sergey Levine, [CS285: Deep Reinforcement Learning, Berkeley](#)