上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Institute of Media, Information, and Network

Transfer & Meta Reinforcement Learning

Nuowen Kan
阚诺文

Department of Electronic Engineering
Shanghai Jiao Tong University

June. 2020

Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Challenges for (robotic) RL in real-world

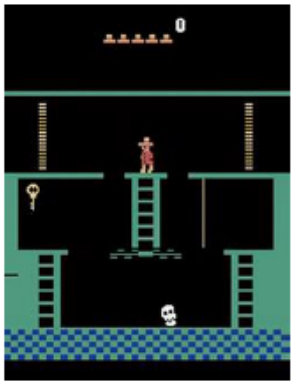
- *Learning behavioral skills* is the purview of reinforcement learning
  - *Active behavioral skills*, not passive recognition tasks

- RL can master a wide range of (single) tasks in simulated environments, but can struggle to generalize to open-world settings (sim2real problem)
  - Can only learn skills based on what is present in the data (prior)
  - Require broad generalization

this is easy (mostly)

Why?

this is impossible

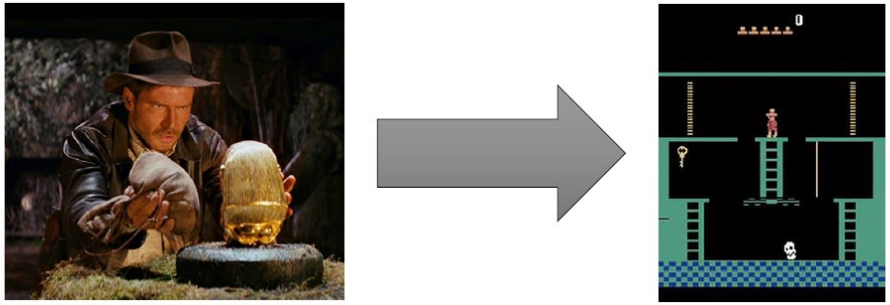


- ✓ Getting key = reward
- ✓ Opening door = reward
- ✓ Getting killed by skull = bad

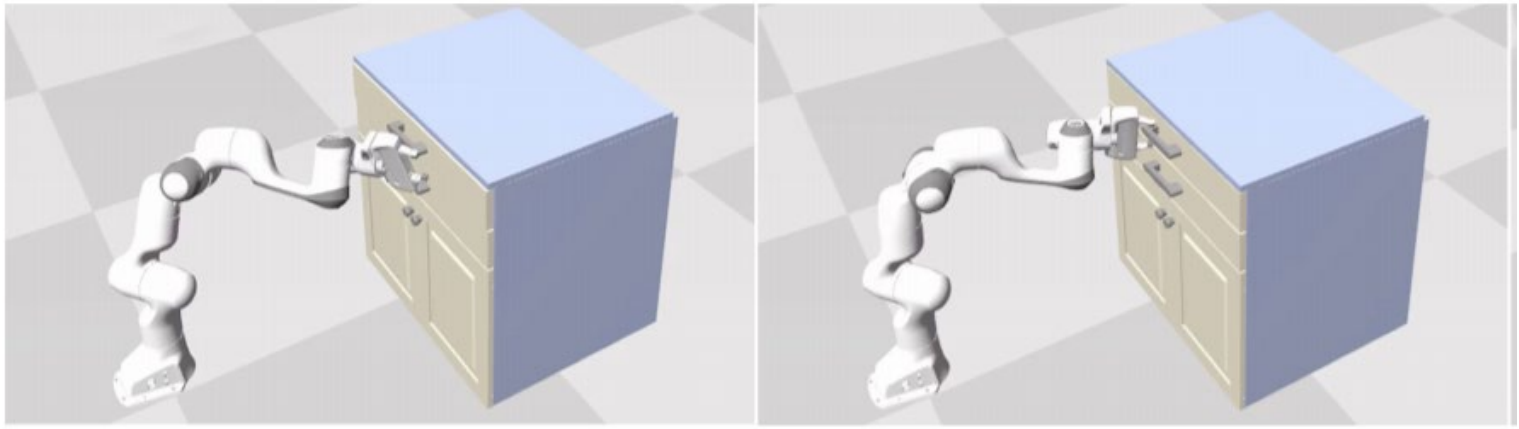
Prior knowledge

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!**



Require broad generalization



Sim to real

Sim2real

Improving RL in real-world

Three ingredients to improve RL (robotic learning system) in real-world settings:

1. The ability to learn from diverse prior data *(offline RL)*
2. The ability to rapidly master new tasks and environments *(meta-learning, transfer learning)*
3. The ability to do all this autonomously

Contents at a glance

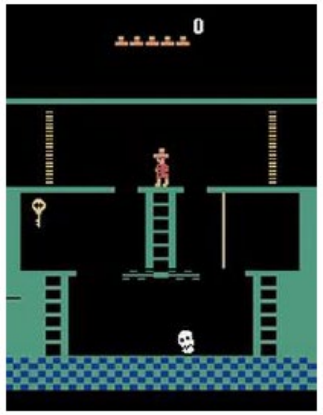
- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Transfer learning terminology

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

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

Transfer learning terminology

in RL, task= MDP!

Transfer setting (total 9)

- $< S, A, R, P >, < S, A, R, P >, < S, A, R, P >, < S, A, R, P >$ (2)
- $< S, A, R, P >, < S, A, R, P >, < S, A, R, P >, < S, A, R, P >, < S, A, R, P >$ (3)
- $< S, A, R, P >, < S, A, R, P >, < S, A, R, P >, < S, A, R, P >$ (3)
- $< S, A, R, P >$ (1)

✓ How is the knowledge stored?

a) **Q-function:** tells us which actions or states are good

b) **Policy:** tells us which actions are potentially useful some actions are never useful!

c) **Models:** what are the laws of physics that govern the world?

d) **Features/hidden states:** provide us with a good representation

- Don't underestimate this!

Question:

How can we frame transfer learning problems?

No single solution!

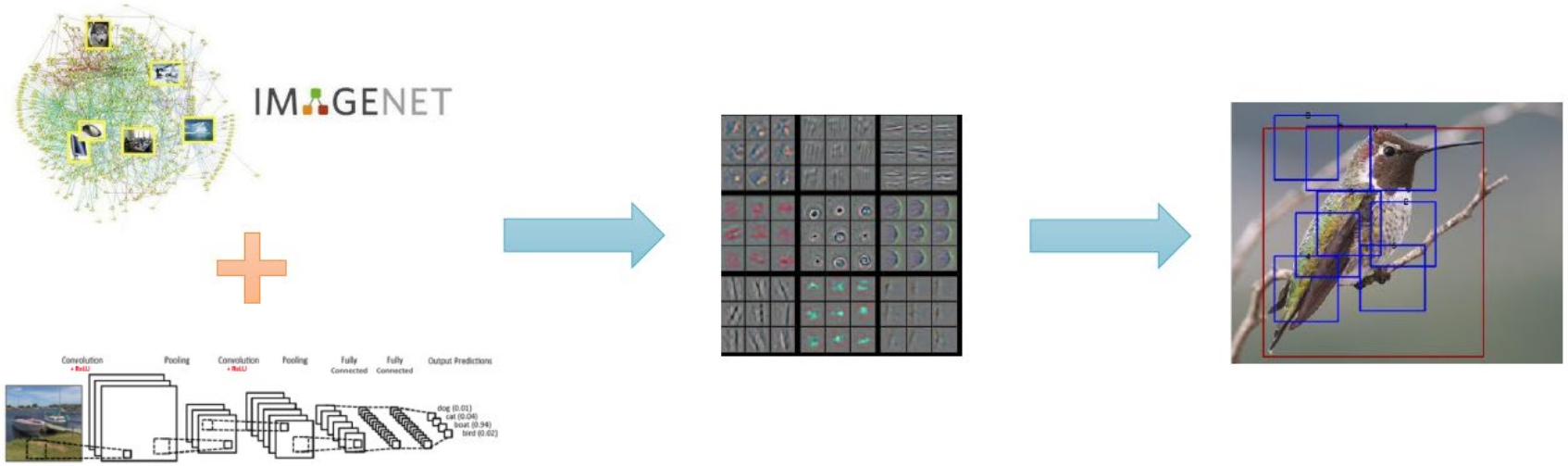
Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Forward transfer: train on one task, transfer to a new task

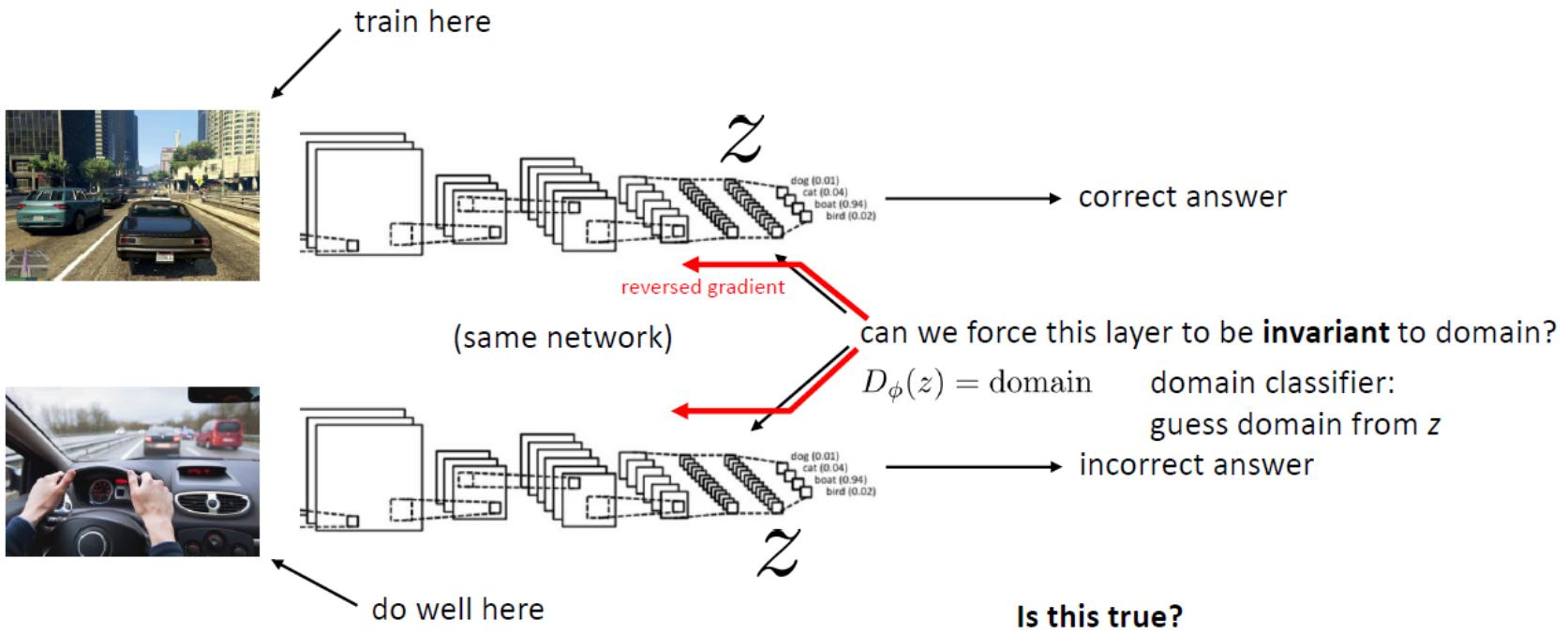
Pretraining + Finetuning

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



Domain adaptation for supervised learning

Domain adaptation in computer vision



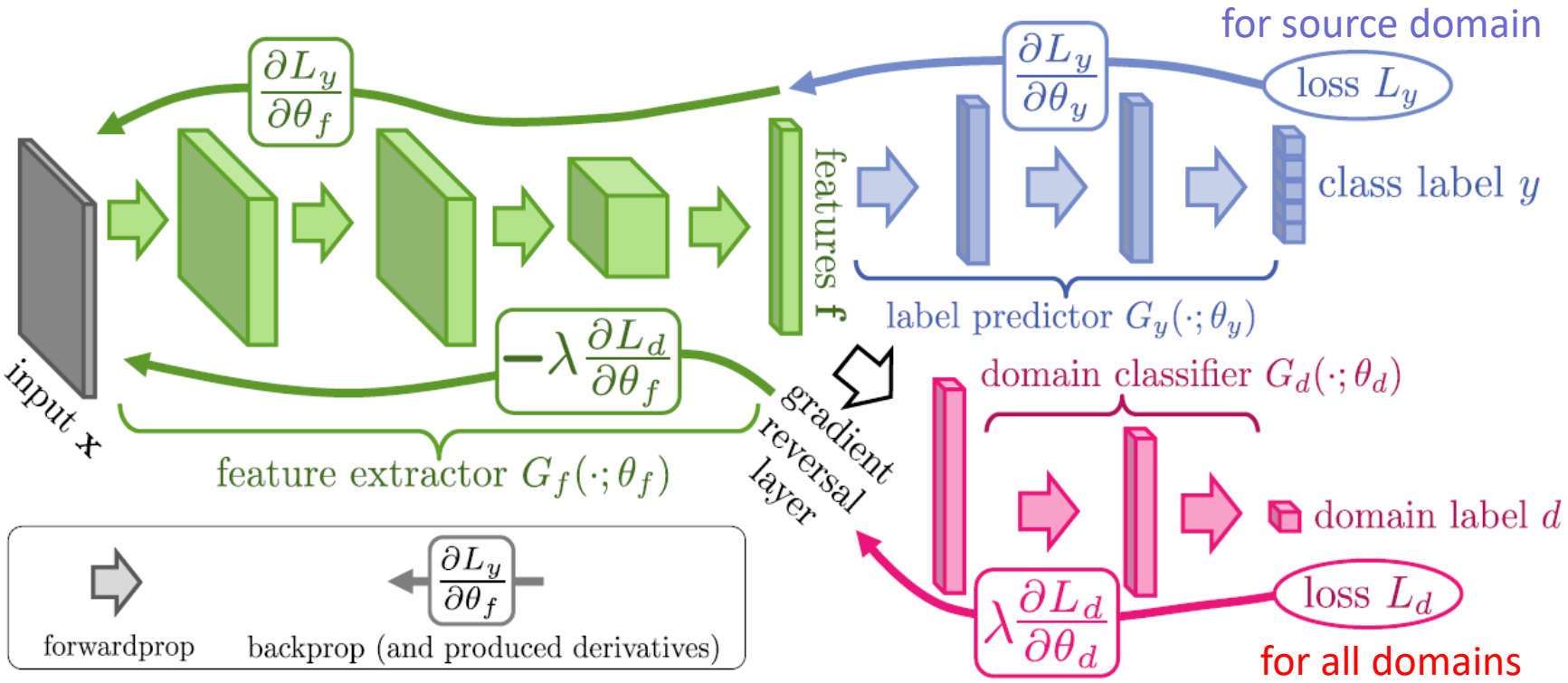
Invariance assumption: everything that is **different** between domains is **irrelevant**

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 for supervised learning

From the perspective of adversarial learning



- Adversarial training ensures that the feature distributions over the two domains are made similar (*as indistinguishable as possible for the domain classier*)

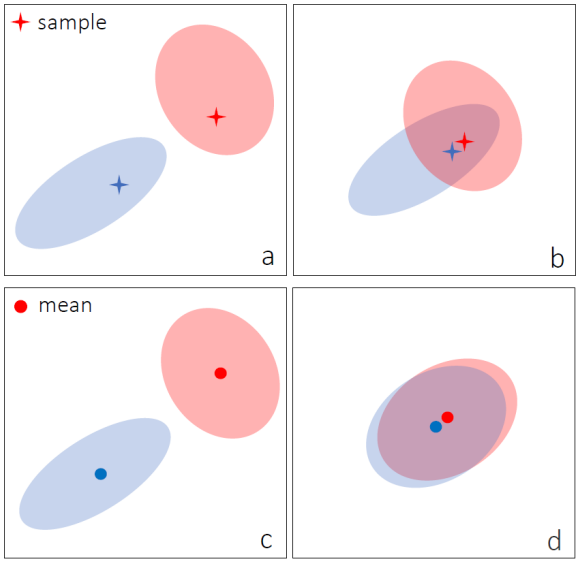
  - Ganin, Ustinova, Ajakan, Germain, Larochelle, Laviolette, Marchand, Lempitsky. **Domain-Adversarial Training of Neural Networks**. 2015.

Domain adaptation for supervised learning

From the perspective of Bayesian

Consider the uncertainty

Sample distance v.s. distribution distance



Definition 2.1 (Domain Invariance) *Let* $\mathbf{x}_i$ *be a given sample from domain* D_i *in the domain space* $\mathfrak{D}$*, and* $\mathbf{x}_\zeta = g_\zeta(\mathbf{x}_i)$ *be a transformation of* $\mathbf{x}_i$ *in another domain* D_ζ *from the same domain space, where* $\zeta \sim q(\zeta)$*.* $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ *denotes the output distribution of input* $\mathbf{x}$ *with model* $\boldsymbol{\theta}$*. Model* $\boldsymbol{\theta}$ *is domain-invariant in* $\mathfrak{D}$ *if*

$$p_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x}_i) = p_{\boldsymbol{\theta}}(\mathbf{y}_\zeta|\mathbf{x}_\zeta), \qquad \forall \zeta \sim q(\zeta). \tag{3}$$

$$p_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x}_i) = \mathbb{E}_{q_\zeta}[p_{\boldsymbol{\theta}}(\mathbf{y}_\zeta|\mathbf{x}_\zeta)]. \tag{4}$$

$$\begin{aligned} &\mathbb{D}_{\mathrm{KL}}\left[p_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x}_i)||\mathbb{E}_{q_\zeta}[p_{\boldsymbol{\theta}}(\mathbf{y}_\zeta|\mathbf{x}_\zeta)]\right] \\ &\quad \leq \mathbb{E}_{q_\zeta}\left[\mathbb{D}_{\mathrm{KL}}\left[p_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x}_i)||p_{\boldsymbol{\theta}}(\mathbf{y}_\zeta|\mathbf{x}_\zeta)\right]\right], \end{aligned} \tag{5}$$

- Zehao Xiao, Jiayi Shen, Xiantong Zhen, Ling Shao, Cees G. M. Snoek, "**A Bit More Bayesian: Domain-Invariant Learning with Uncertainty.** " ICML, 2021.

Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Domain adaptation for RL

How do we apply these ideas in RL?

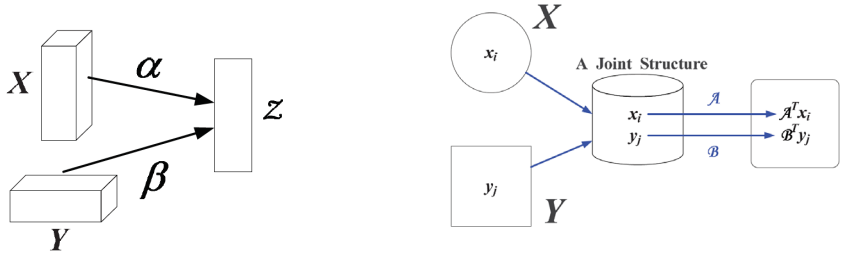
- ☐ What issues are we likely to face?

  - ✓ **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

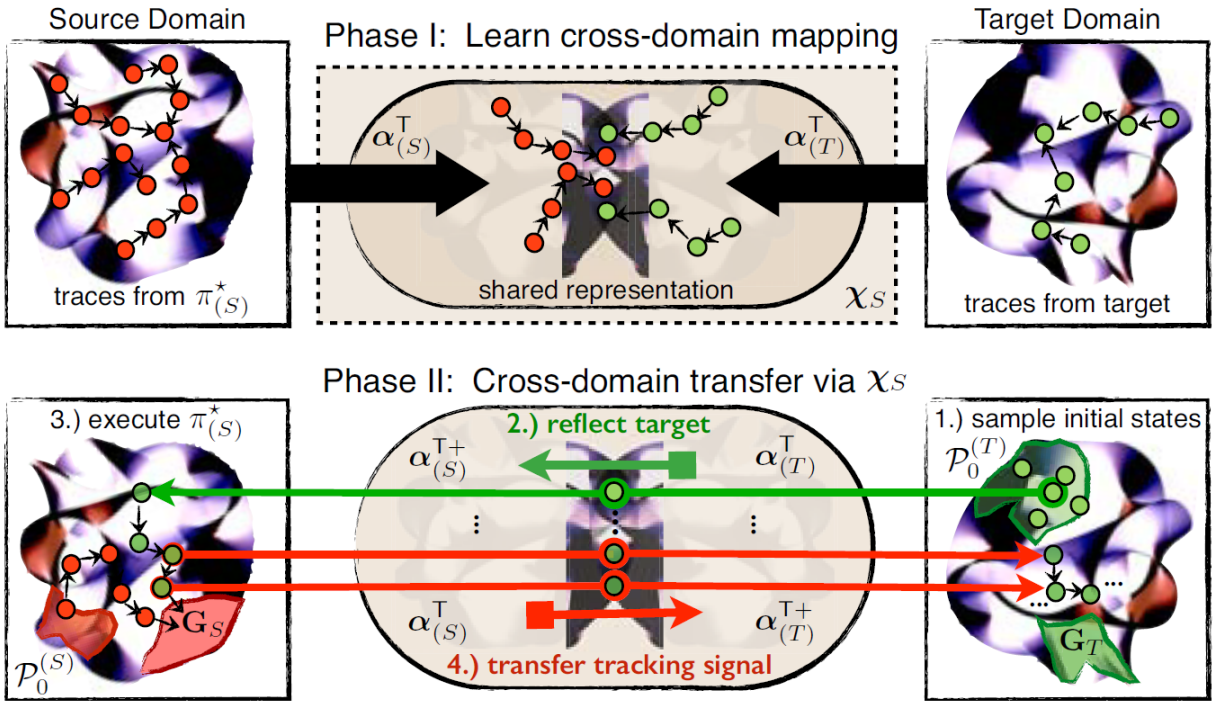

  - ✓ **Finetuning issues:** if pretraining & finetuning, the finetuning process may still need to explore, but optimal policy during finetuning may be deterministic!

Unsupervised cross-domain transfer via manifold alignment



Unsupervised Manifold Alignment

Learn such mapping function α and β to project the data instances to a new lower dimensional space by matching the local geometry and preserving the neighborhood relationship with each set.

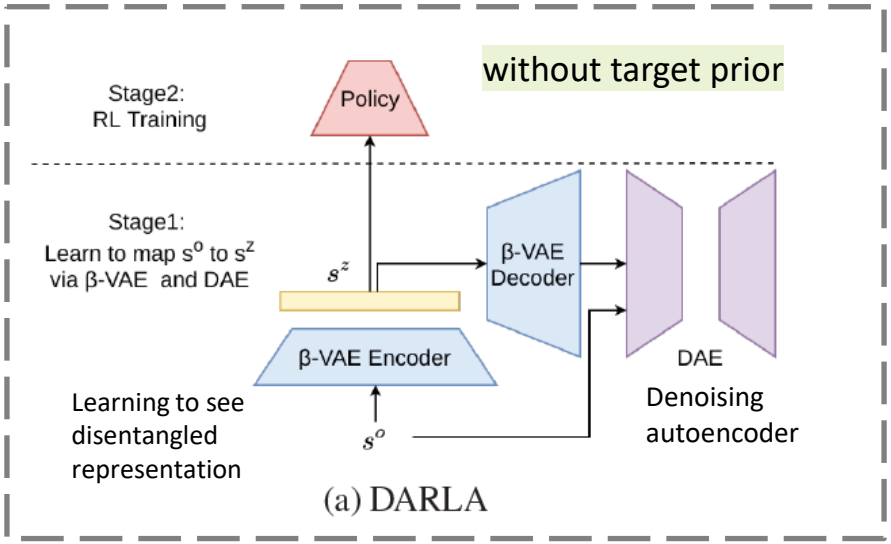


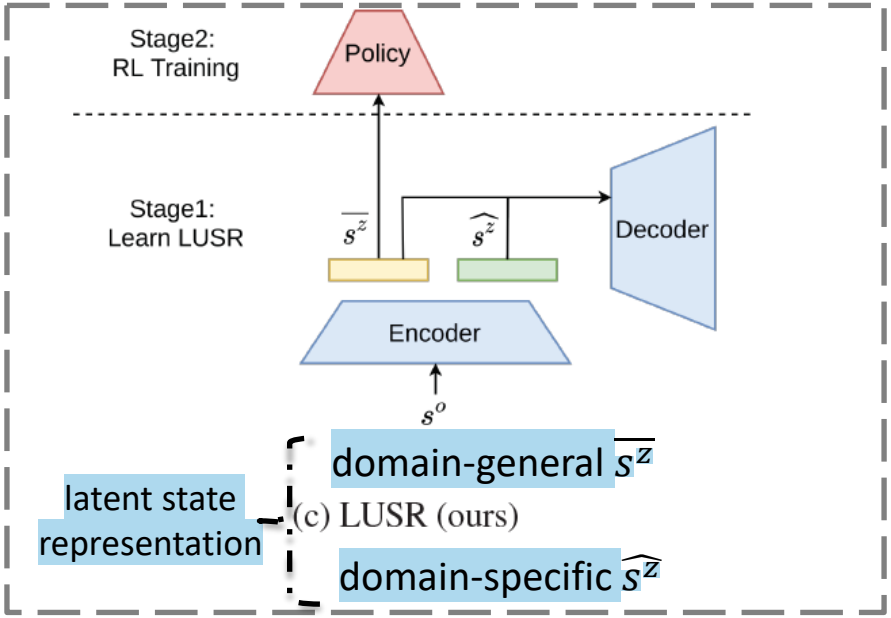
Learning an inter-state mapping, also suitable for transferring cost function

Policy transfer and improvement, initializing the initial policy in target domain for an effective training.

- Ammar, Haitham Bou, et al. "Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment." Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

Domain adaptation for RL: recent papers

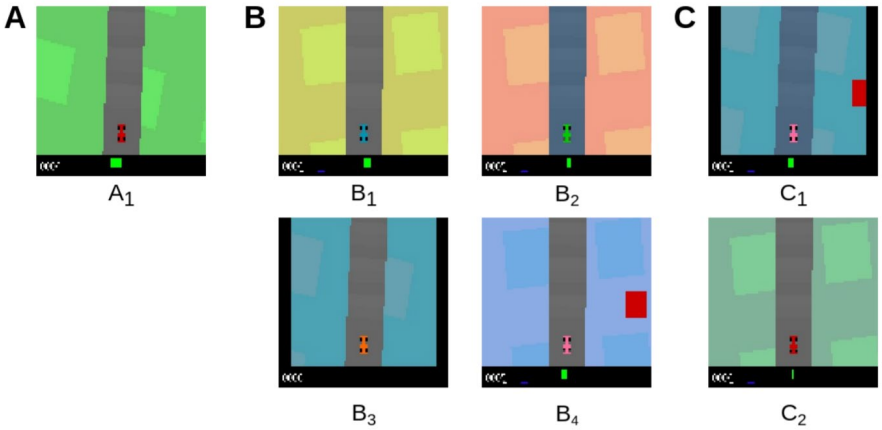




- Assumption of transfer:

$$T_S \approx T_T,\ R_S \approx R_T$$
$$A_S = A_T, \text{ but } S_S \neq S_T$$

$$S_S^o \neq S_T^o$$
$$S_S^z = (\widehat{S_S^z}, \overline{S_S^z}); \quad S_T^z = (\widehat{S_T^z}, \overline{S_T^z})$$
$$\overline{S_S^z} = \overline{S_T^z}; \quad \widehat{S_S^z} \neq \widehat{S_T^z}$$
$$T_S^o \neq T_T^o; \quad R_S^o \neq R_T^o$$
$$T_S^z = T(\overline{S_S^z}) = T(\overline{S_T^z}) = T_T^z$$
$$R_S^z = R(\overline{S_S^z}) = R(\overline{S_T^z}) = R_T^z$$



Domain adaptation for RL: recent papers

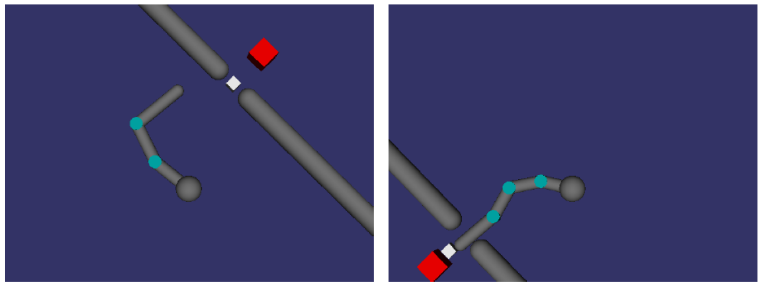
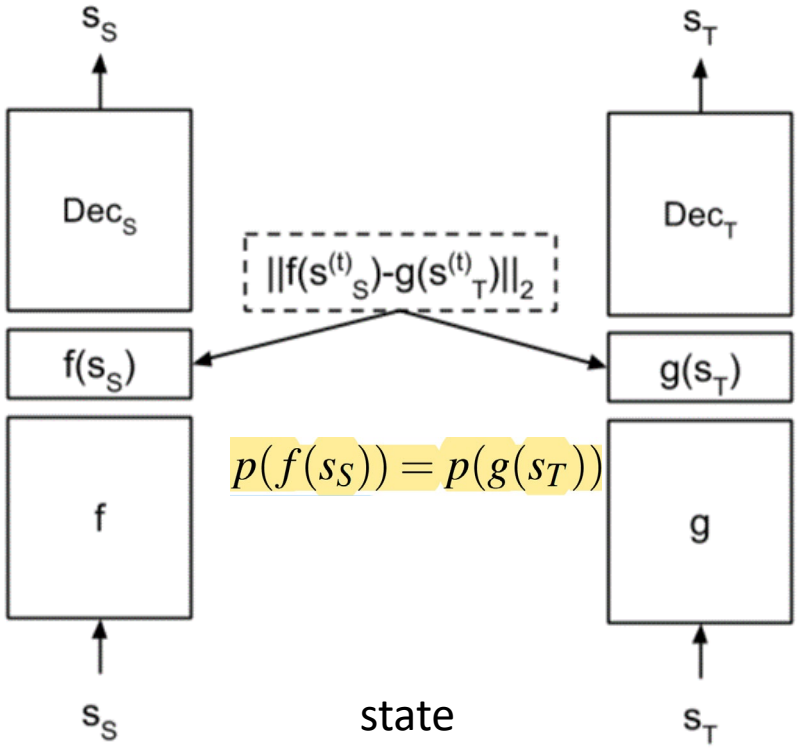


Figure 2: The 3 and 4 link robots performing the button pressing task, which

Different domains



To approximate the requirement that $p(f(s_{Sp,r})) = p(g(s_{Tp,r}))$, we assume a pairing P of states in the proxy domains as described in 3.3. The pairing P is a list of pairs of states (s_{Sp}, s_{Tp}) which are corresponding across domains. As f and g are parametrized as neural networks, we can optimize them using the similarity loss metric introduced by Chopra et al. (2005):

$$\mathscr{L}_{\text{sim}}(s_{Sp}, s_{Tp}; \theta_f, \theta_g) = ||f(s_{Sp,r}; \theta_f) - g(s_{Tp,r}; \theta_g)||_2.$$

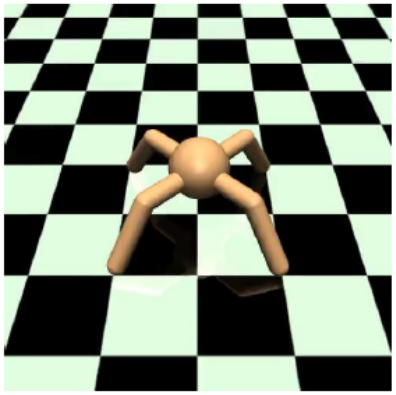
Suggested readings

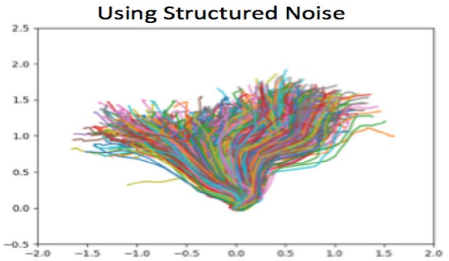
- Higgins, I.; Pal, A.; Rusu, A. A.; Matthey, L.; Burgess, C. P.; Pritzel, A.; Botvinick, M.; Blundell, C.; and Lerchner, A. 2017. **Darla: Improving zero-shot transfer in reinforcement learning**. *arXiv preprint arXiv:1707.08475*.
- Laskin, M.; Srinivas, A.; and Abbeel, P. 2020. **Curl: Contrastive unsupervised representations for reinforcement learning.** In *International Conference on Machine Learning*, 5639–5650. PMLR.
- Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, Jeffrey L. Krichmar; **Domain Adaptation In Reinforcement Learning Via Latent Unified State Representation.** AAAI. 2020.
- A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "**Learning invariant feature spaces to transfer skills with reinforcement learning**," *ICLR*, 2017.
- 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,** ICML, 2016**.**
- Eysenbach et al., **Off Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers**, ICLR, 2021

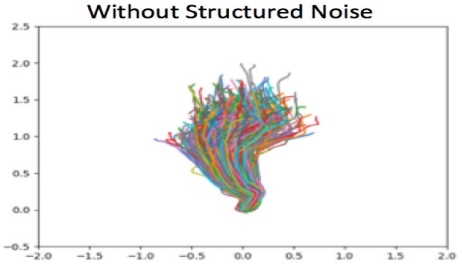
Domain adaptation for RL

What if we can also finetune for RL?

- ☐ What issues are we likely to face?
  - ✓ **RL tasks are generally much less diverse**
    - ✓ Features are less general
    - ✓ Policies & value functions become overly specialized
  - ✓ **Optimal policies in fully observed MDPs are deterministic**
    - ✓ Loss of exploration at convergence
    - ✓ Low entropy policies adapt very slowly to new settings

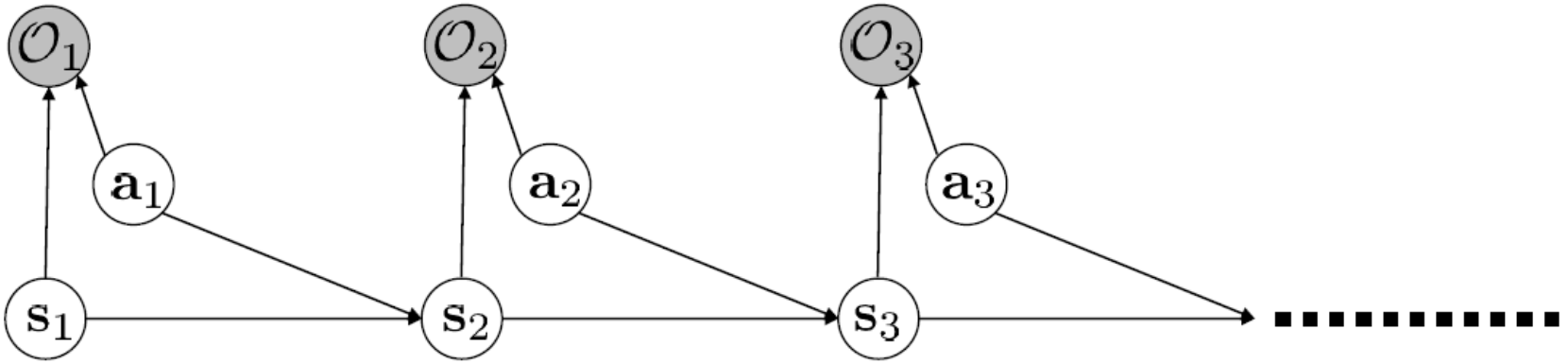






Finetuning with maximum-entropy policies

How can we increase diversity and entropy?



$\pi(\mathbf{a}|\mathbf{s}) = \exp(Q_\phi(\mathbf{s},\mathbf{a}) - V(\mathbf{s}))$ optimizes $\sum_t E_{\pi(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)] + E_{\pi(\mathbf{s}_t)}[\mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))]$

policy entropy

Act **as randomly as possible** while collecting high rewards!

- Haarnoja, Tang, et al. "Reinforcement Learning with Deep Energy Based Policies"

Suggested readings

- Haarnoja , Tang, et al. (2017). **Reinforcement Learning with DeepEnergy 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

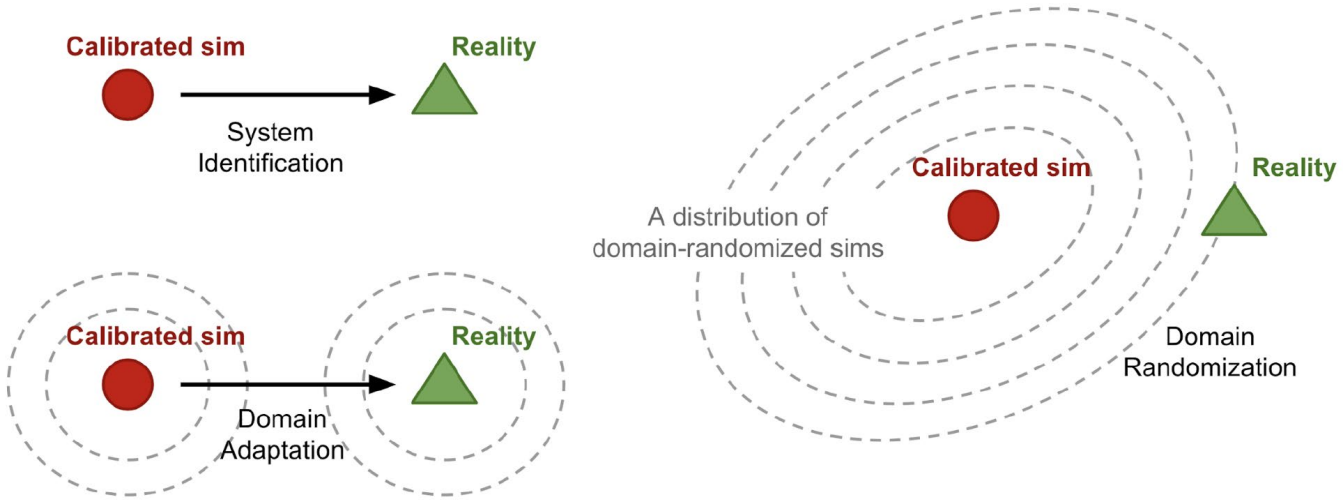
Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Domain randomization for RL

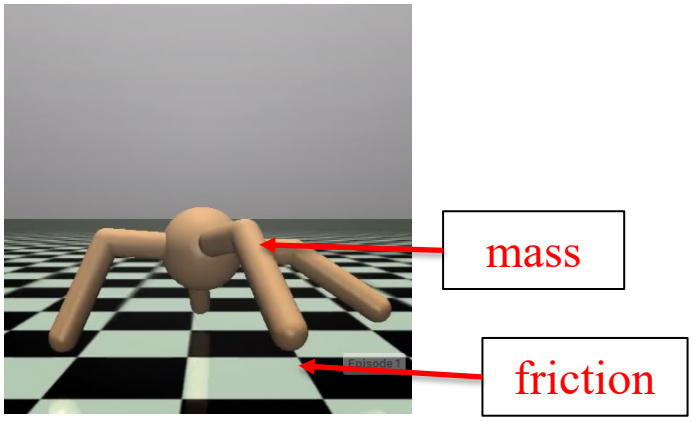
What if we can manipulate the source domain?

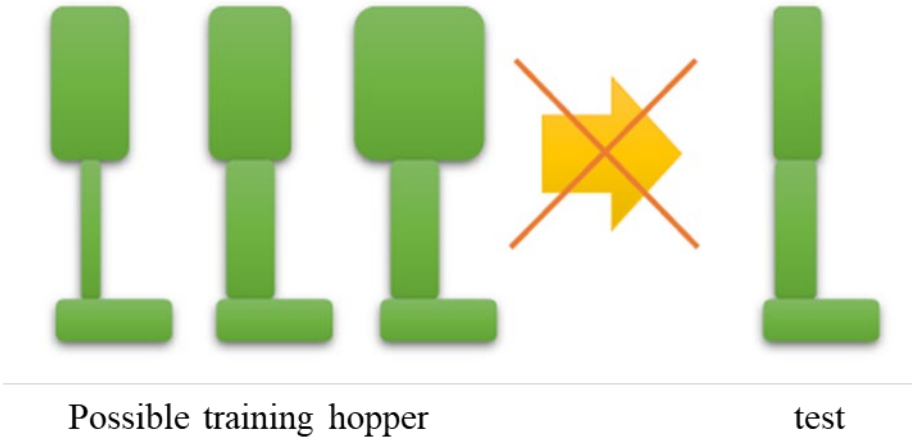
- So far: source domain (e.g., empty room) and target domain (e.g., corridor) are fixed
- What if we can **design** the source domain, and we have a **difficult** target domain?
  - Often the case for simulation to real world transfer



Domain randomization for RL: recent papers

- **Model mismatch**





- Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, Hongkai Xiong, "**Monotonic Robust Policy Optimization with Model Discrepancy**", ICML, 2021.

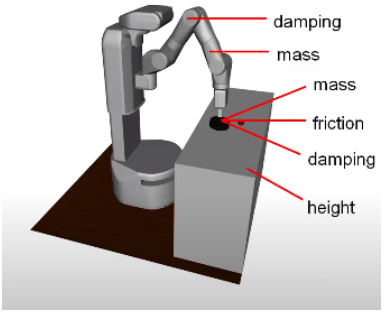
Domain randomization for RL: recent papers

Monotonic Worst-Case Improvement

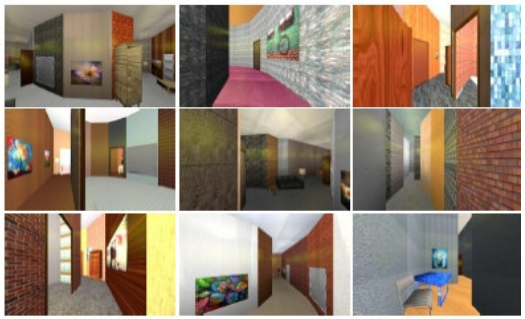
- **Manipulate the source domain in simulator**
  - **Domain randomization**

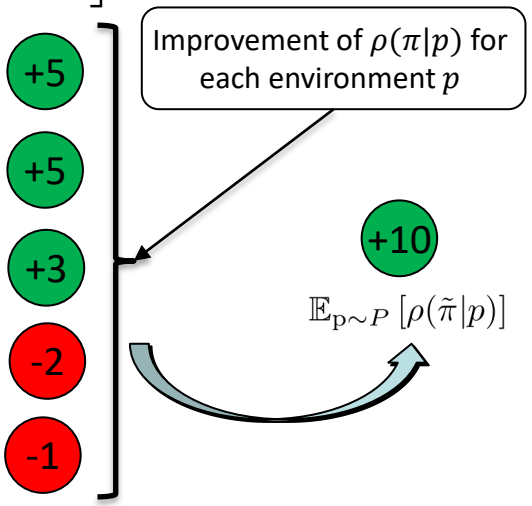
$$\mathbb{E}_{\mathrm{p}\sim P}\left[\rho(\tilde{\pi}|p)\right] \qquad \rho(\pi|p) = \mathbb{E}_{a_t\sim P(\cdot|s_t),s_{t+1}\sim\mathcal{T}(\cdot|s_t,a_t,p)}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right]$$

    - Physical coefficient randomization



    - Visual randomization





  - **Drawbacks: Average formulation leads to suboptimal policy**
  - **Improve generalization by reducing variance of $\rho(\pi|p)$**
    - $\boldsymbol{Var}(\rho(\pi|p))$ **is bounded by the worst-case performance on environment** p_w**:**

$$|\rho(\pi|p) - \mathbb{E}_{p\sim P}\left[\rho(\pi|p)\right]| \leq |\rho(\pi|p_w) - \mathbb{E}_{p\sim P}\left[\rho(\pi|p)\right]|$$

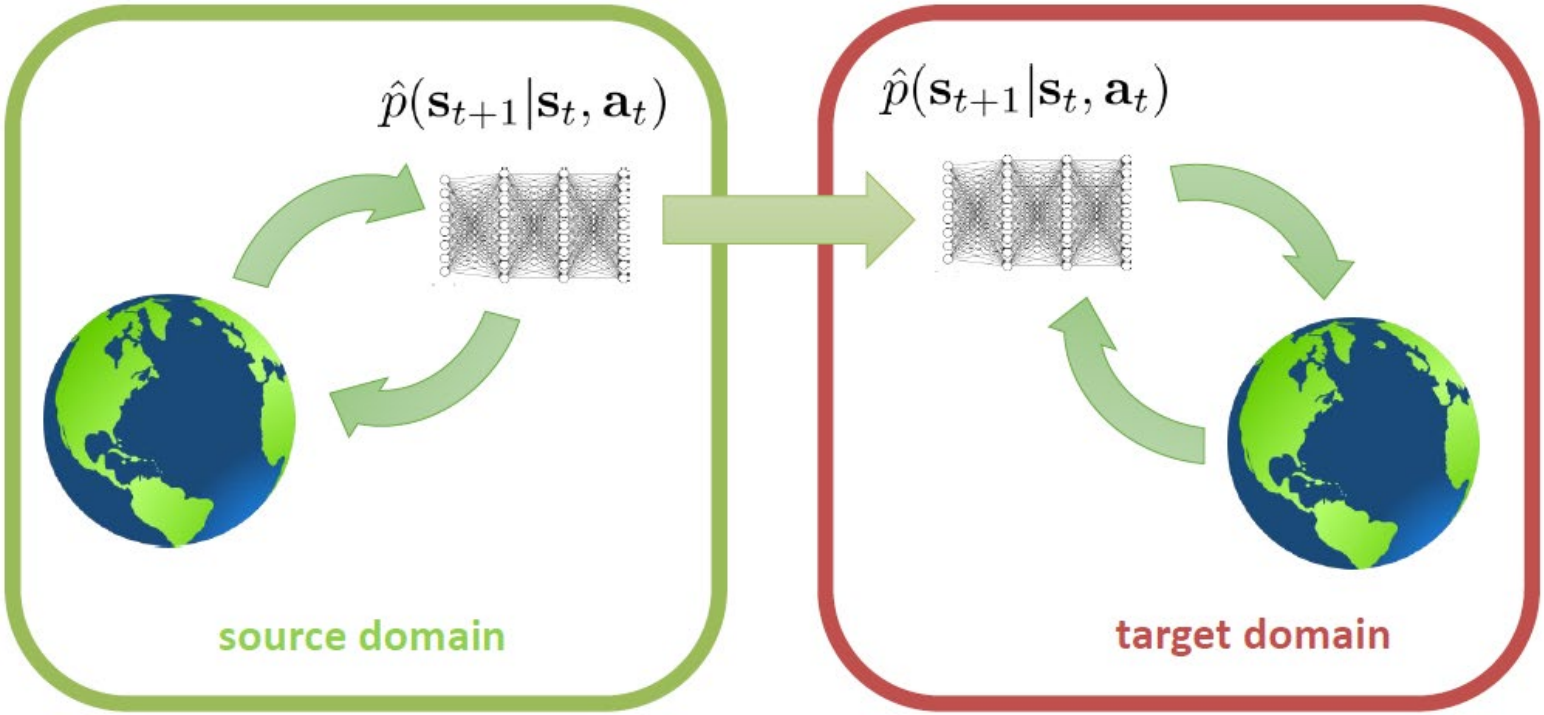
Suggested readings

- 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.**
- James et al. (2017). **Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task.**

Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Transferring models



Transferring models

- **System identification** is to build a mathematical model for a physical system; in the context of RL, the mathematical model is the simulator. To make the simulator more realistic, careful calibration is necessary.

- Unfortunately, calibration is expensive. Furthermore, many physical parameters of the same machine might vary significantly due to temperature, humidity, positioning or its wear-and-tear in time.

- Might identify the dynamics by latent variables, see “skills discovery”.

Transferring value functions

- ***Transferring value functions***: not straightforward to transfer by itself, since the value function entangles the dynamics and reward, but possible with a decomposition.
  - For more details, see: Barreto et al., Successor Features for Transfer in Reinforcement Learning

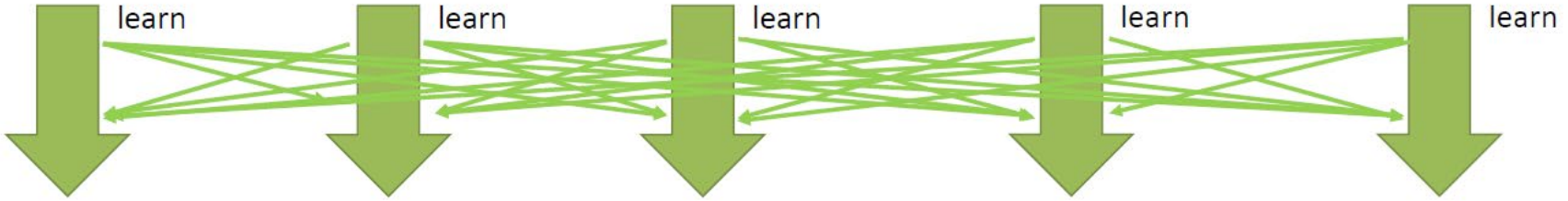
Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Multi-Task Transfer

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



Multi-task learning can:

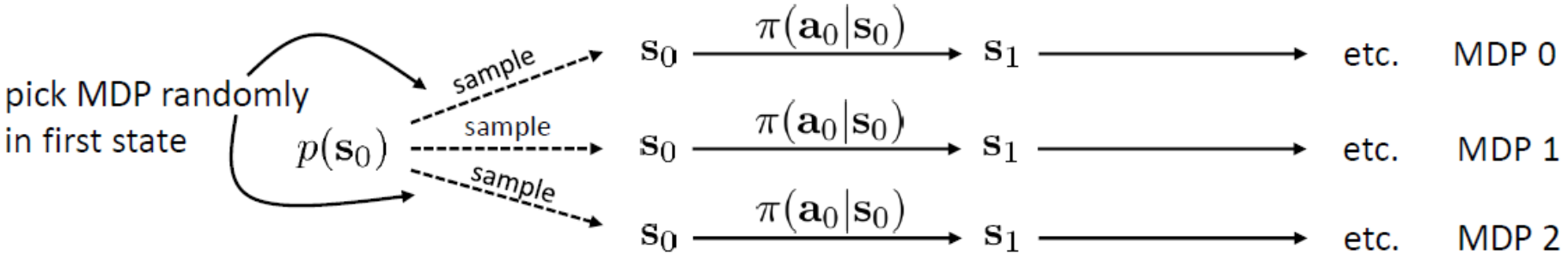
- Accelerate learning of all tasks that are learned together
- Provide better pre-training for down-stream tasks

Similar to meta learning, we will discuss later…

Multi-Task Transfer

Can we solve multiple tasks at once?

- ✓ Multi-task RL corresponds to single-task RL in a **joint MDP**



Multi-Task Transfer

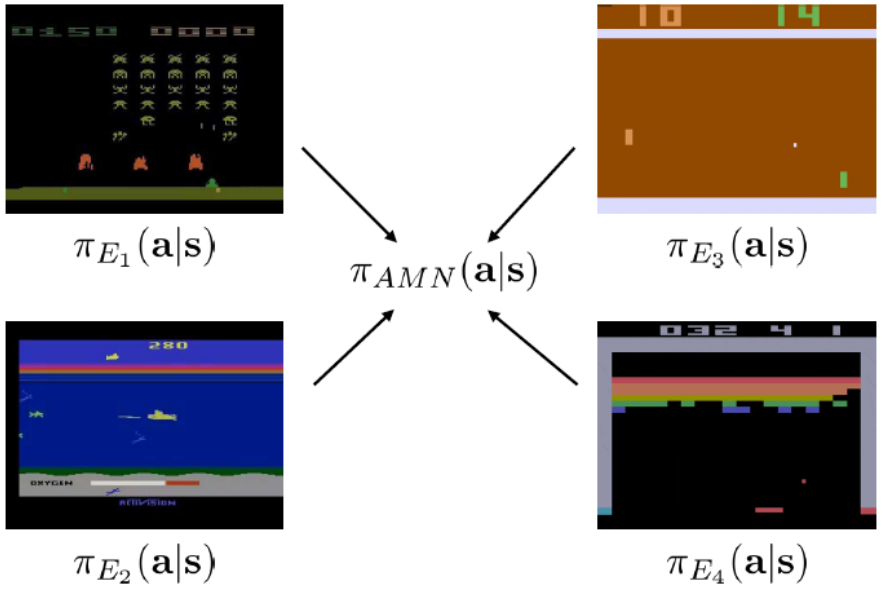
What is difficult about this?

a. **Gradient interference:** becoming better on one task can make you worse on another

b. **Winner-take-all problem:** imagine one task starts getting good –algorithm is likely to prioritize that task (to increase average expected reward) at the expensive of others

In practice, this kind of multi-task RL is very challenging

Actor-mimic and policy distillation



Cross entropy loss:

$$\mathcal{L} = \sum_{\mathbf{a}} \pi_{E_i}(\mathbf{a}|\mathbf{s}) \log \pi_{AMN}(\mathbf{a}|\mathbf{s})$$

See paper Parisotto et al. “Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning” for more details

Multitask transfer for RL: recent papers

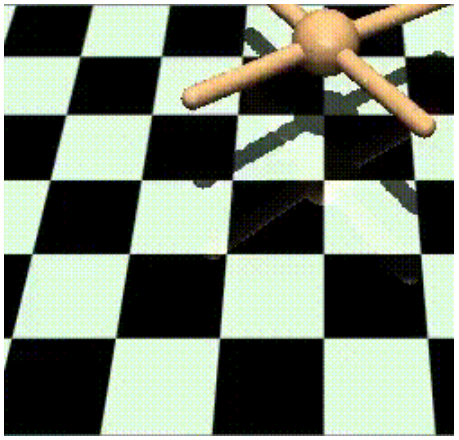
Unsupervised training for multi-task

- Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. **Dynamics-aware unsupervised discovery of skills**. ICLR, 2020.
- Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. **Diversity is all you need: Learning skills without a reward function**. arXiv preprint arXiv:1802.06070, 2018.

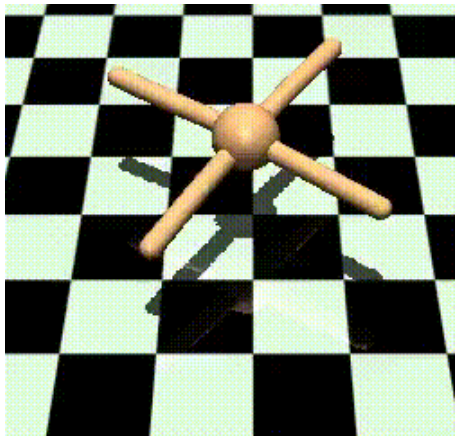
Different leaned skills in Mujoco envs

Ants

111DOF

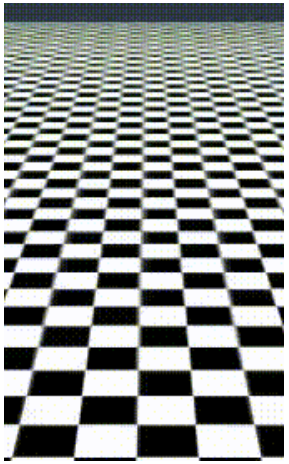


Skill-1



Skill-2

Humanoid

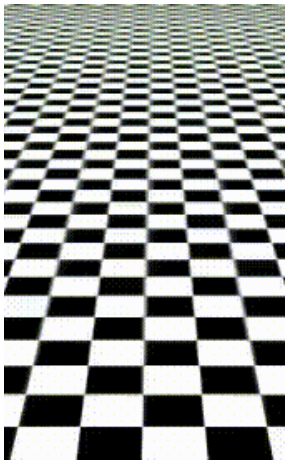


Skill-1

Action sequence

↓

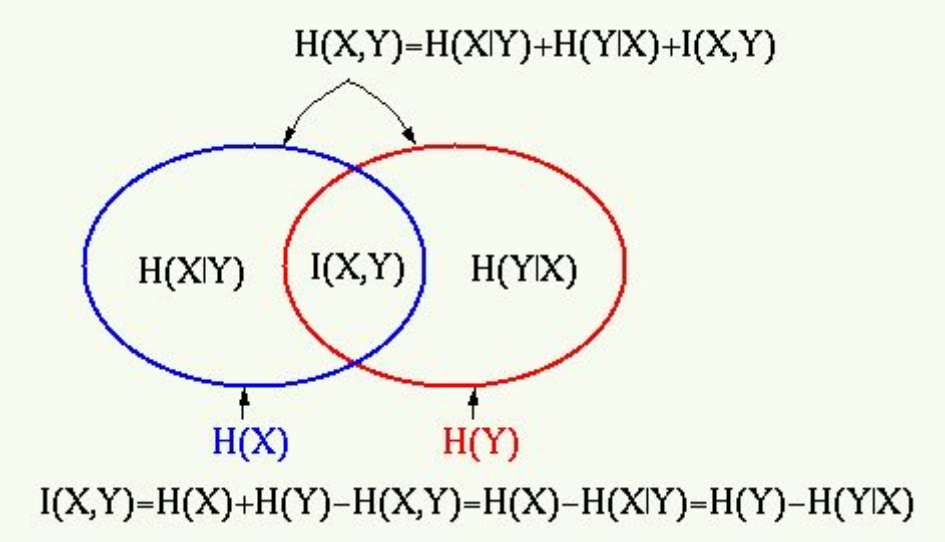
Primitives



Skill-2

Mutual information

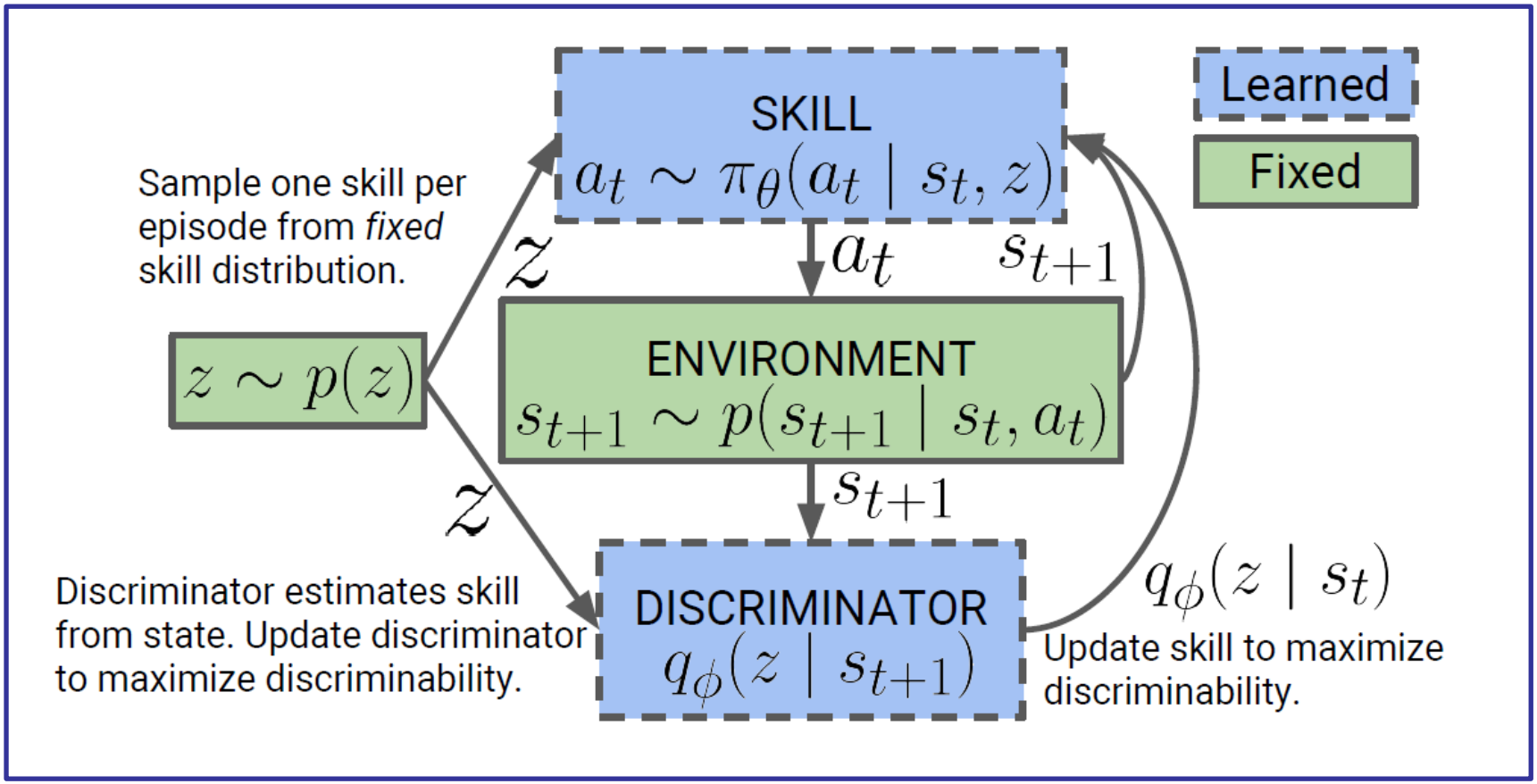
$$\mathrm{I}(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right),$$



Learning skills without a reward function

- $z \sim p(Z)$: skill, *latent variable*
- S : state
- A: action

- $I(;)$: **mutual information**
- $H[\cdot]$: entropy



Learning skills without a reward function

- **What kinds of skills are best:**
  1. Distinguishable
  2. As diverse as possible

skill should control which states the agent visits

states, not actions, are used to distinguish skills

a. $$\begin{aligned}\mathcal{F}(\theta) &\triangleq I(S;Z) + \mathcal{H}[A \mid S] - I(A;Z \mid S) \\ &= (\mathcal{H}[Z] - \mathcal{H}[Z \mid S]) + \mathcal{H}[A \mid S] - (\mathcal{H}[A \mid S] - \mathcal{H}[A \mid S, Z]) \\ &= \mathcal{H}[Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[A \mid S, Z]\end{aligned}$$

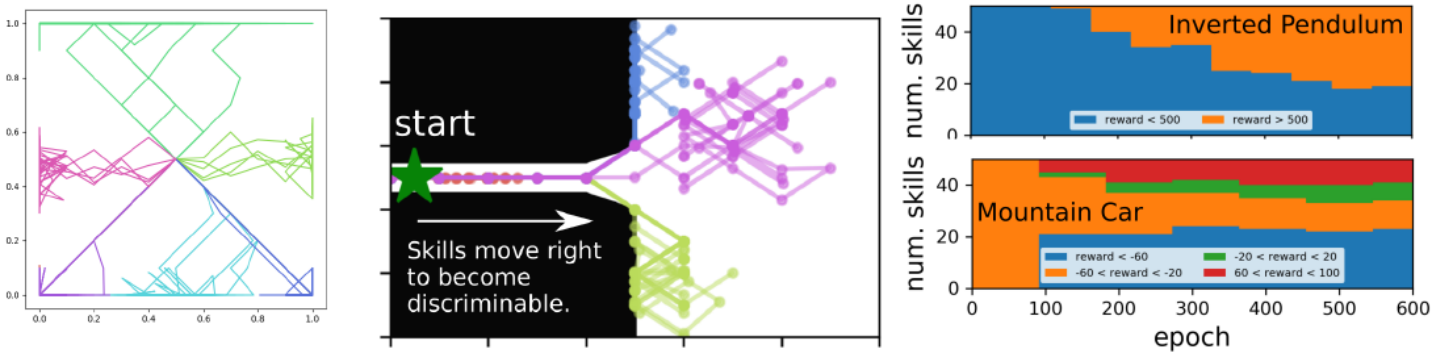
each skill should act as randomly as possible

b. $$\begin{aligned}\mathcal{F}(\theta) &= \mathcal{H}[A \mid S, Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[Z] \\ &= \mathcal{H}[A \mid S, Z] + \mathbb{E}_{z\sim p(z), s\sim\pi(z)}[\log p(z \mid s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)] \\ &\geq \mathcal{H}[A \mid S, Z] + \mathbb{E}_{z\sim p(z), s\sim\pi(z)}[\log q_\phi(z \mid s) - \log p(z)] \triangleq \mathcal{G}(\theta, \phi)\end{aligned}$$

c. $$r_z(s,a) \triangleq \log q_\phi(z \mid s) - \log p(z)$$

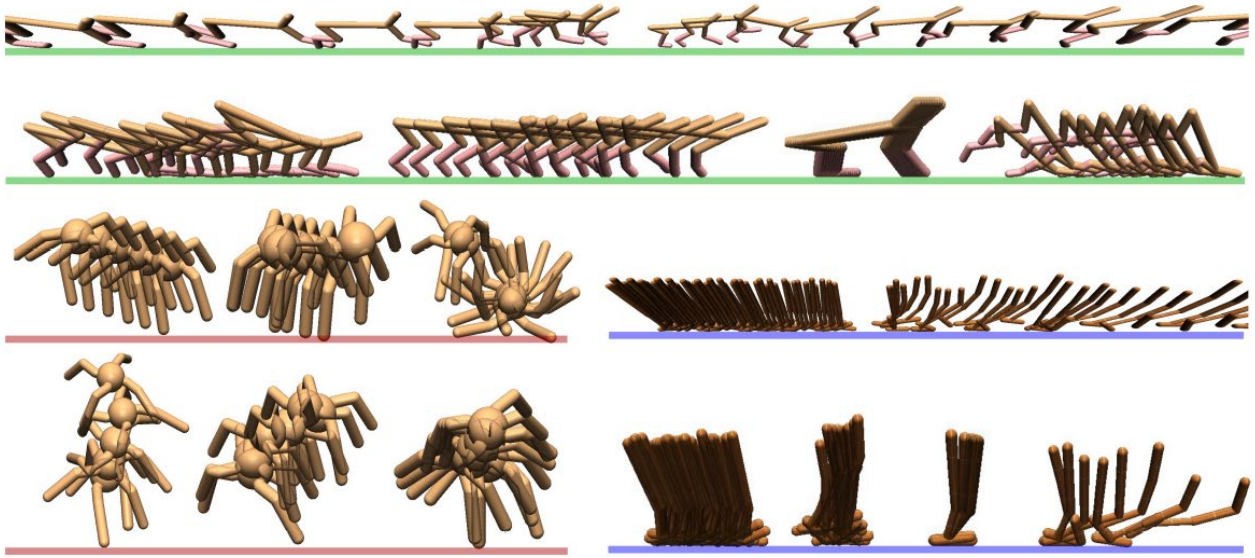
Learning with SAC $\pi_\theta(a|s,z)$

Analysis of Learned Skills



(a) 2D Navigation (b) Overlapping Skills (c) Training Dynamics

Figure 2: *(Left)* DIAYN skills in a simple navigation environment; *(Center)* skills can overlap if they eventually become distinguishable; *(Right)* diversity of the rewards increases throughout training.



Harnessing Learned skills

1. Accelerating learning

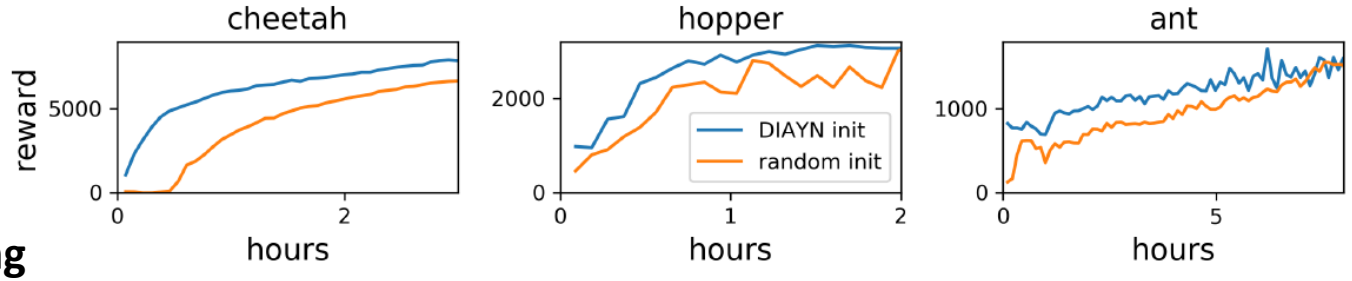


Figure 5: **Policy Initialization**: Using a DIAYN skill to initialize weights in a policy accelerates learning, suggesting that pretraining with DIAYN may be especially useful in resource constrained settings. Results are averages across 5 random seeds.

2. Imitation learning

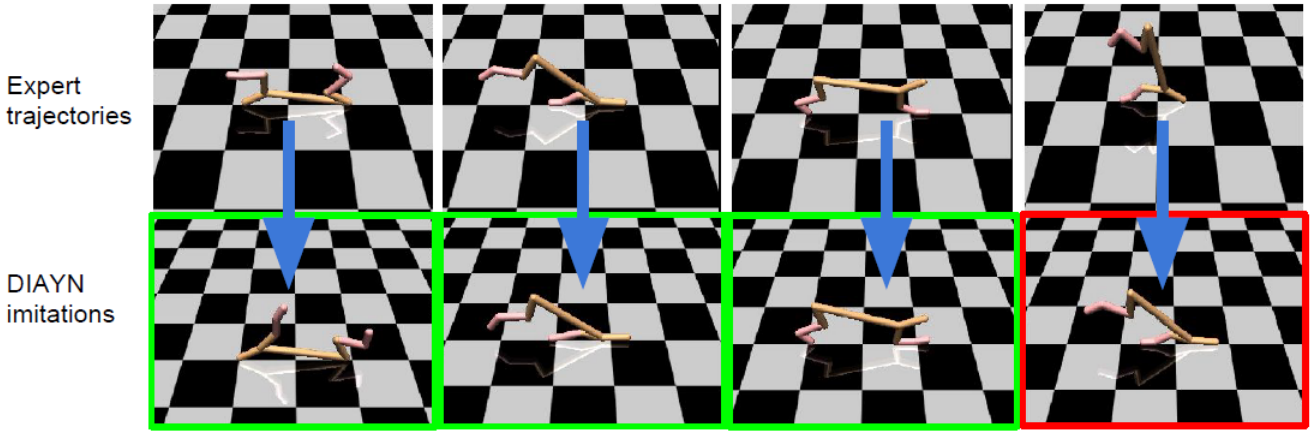


Figure 9: **Imitating an expert**: DIAYN imitates an expert standing upright, flipping, and faceplanting, but fails to imitate a handstand.

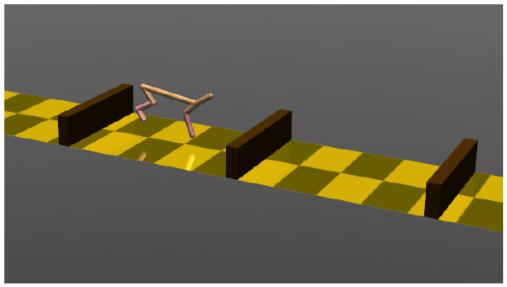
$$\hat{z} = \arg\max_{z} \Pi_{s_t \in \tau^*} q_\phi(z \mid s_t)$$

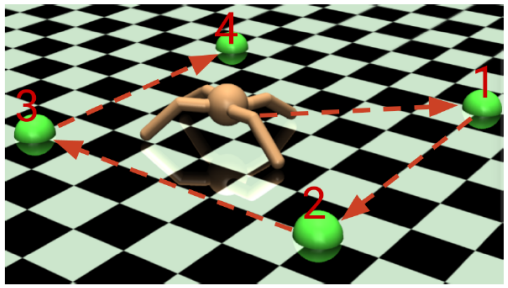
Use for hierarchical RL

3. Hierarchical RL Hierarchical RL should decompose a complex task into motion primitives.

In practice, algorithms for hierarchical RL can encounter many problems:

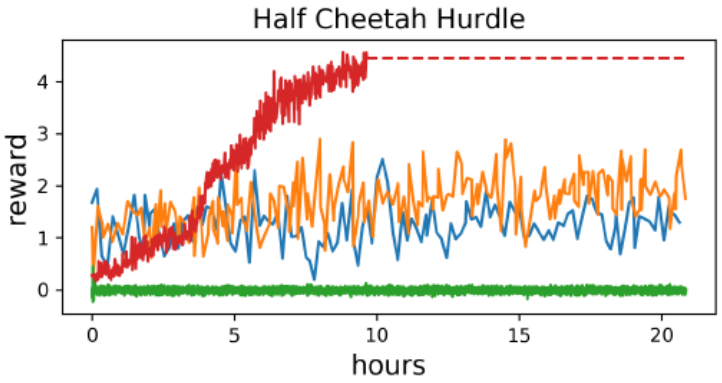
1. each motion primitiv
2. the hierarchical polic
3. all motion primitives





Cheetah Hurdle Ant Navigation

- we learn a meta-controller w … ıe next k steps



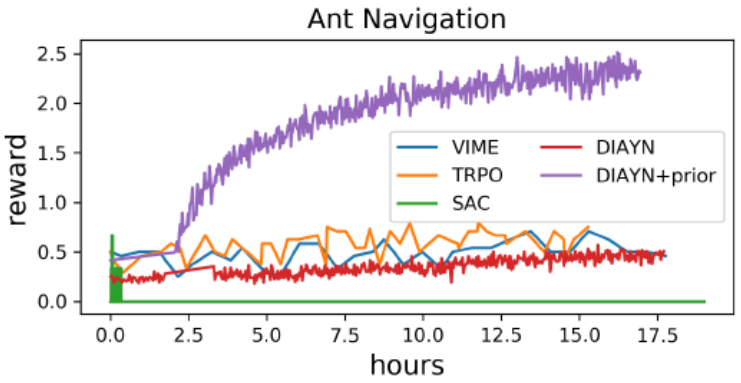


Figure 7: **DIAYN for Hierarchical RL**: By learning a meta-controller to compose skills learned by DIAYN, cheetah quickly learns to jump over hurdles and ant solves a sparse-reward navigation task.

Dynamics-aware unsupervised discovery of skills

$$\mathcal{I}(s'; z \mid s) = \mathcal{H}(z \mid s) - \mathcal{H}(z \mid s', s)$$
$$= \mathcal{H}(s' \mid s) - \mathcal{H}(s' \mid s, z)$$

encoding a diverse set of skills in the latent space Z, while making the transitions for a given $z \in Z$ predictable.

- $z \sim p(Z)$: skill, latent variable
- S : state
- $I(;)$: mutual information
- $H[\cdot]$: entropy

$$\mathcal{I}(s'; z \mid s) = \int p(z, s, s') \log \frac{p(s' \mid s, z)}{p(s' \mid s)} ds' ds dz$$

$$\mathcal{I}(s'; z \mid s) = \mathbb{E}_{z,s,s' \sim p}\left[\log \frac{p(s' \mid s, z)}{p(s' \mid s)}\right]$$
$$= \mathbb{E}_{z,s,s' \sim p}\left[\log \frac{q_\phi(s' \mid s, z)}{p(s' \mid s)}\right] + \mathbb{E}_{s,z \sim p}\left[\mathcal{D}_{KL}(p(s' \mid s, z) \,||\, q_\phi(s' \mid s, z))\right]$$
$$\geq \mathbb{E}_{z,s,s' \sim p}\left[\log \frac{q_\phi(s' \mid s, z)}{p(s' \mid s)}\right]$$ Variational lower bound

1. Tighten bound

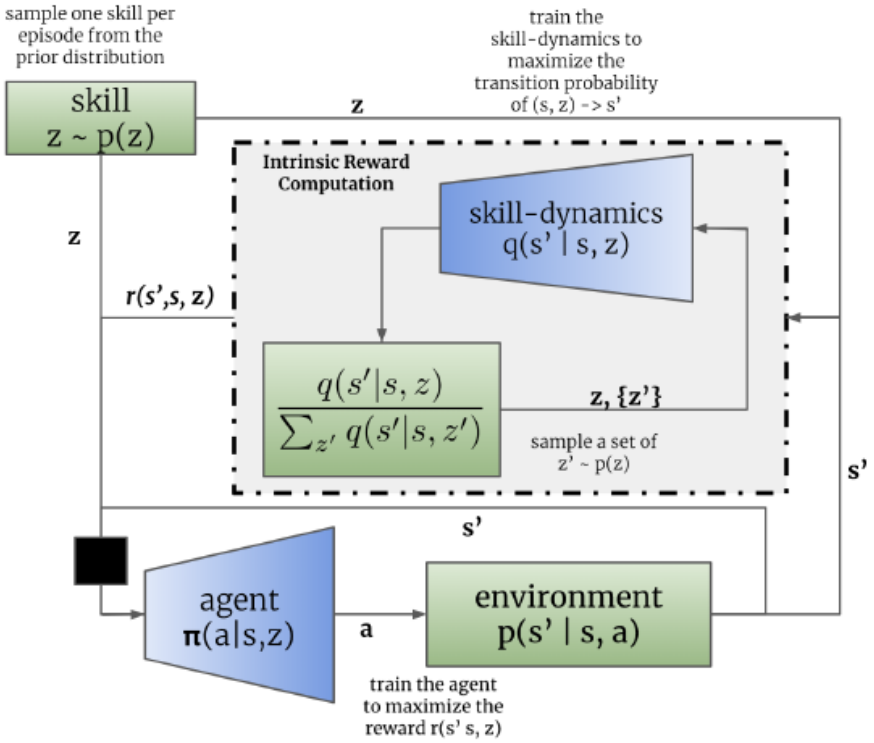
$$\nabla_\phi \mathbb{E}_{s,z}[\mathcal{D}_{KL}(p(s' \mid s, z) \,||\, q_\phi(s' \mid s, z))] = \nabla_\phi \mathbb{E}_{z,s,s'}\left[\log \frac{p(s' \mid s, z)}{q_\phi(s' \mid s, z)}\right]$$
$$= -\mathbb{E}_{z,s,s'}\left[\nabla_\phi \log q_\phi(s' \mid s, z)\right]$$

2. Maximize approximate lower bound

$$r_z(s, a, s') = \log \frac{q_\phi(s' \mid s, z)}{\sum_{i=1}^{L} q_\phi(s' \mid s, z_i)} + \log L, \quad z_i \sim p(z).$$

The approximation is motivated as follows: $p(s' \mid s) = \int p(s' \mid s, z) p(z|s) dz \approx \int q_\phi(s' \mid s, z) p(z) dz \approx \frac{1}{L} \sum_{i=1}^{L} q_\phi(s' \mid s, z_i)$ for $z_i \sim p(z)$, where L denotes the number of samples from the prior. We are using the marginal of variational approximation q_ϕ over the prior $p(z)$ to approximate the marginal distribution of transitions. We discuss this approximation in Appendix C. Note,

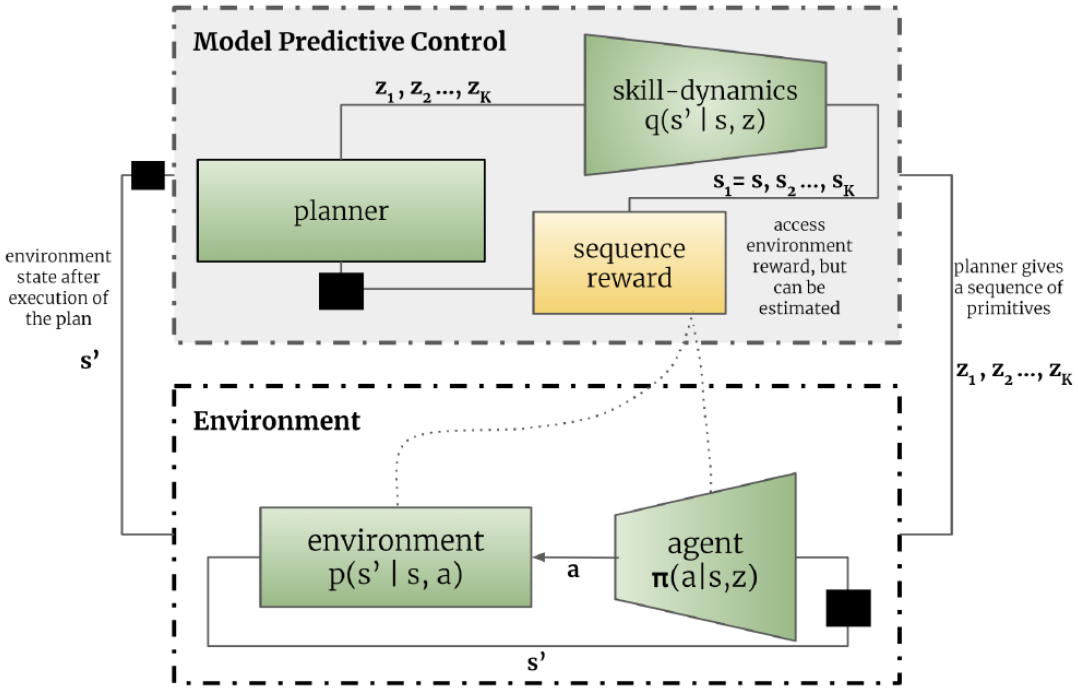
Dynamics-aware discovery of skills



Algorithm 1: Dynamics-Aware Discovery of Skills (DADS)

Initialize π, q_ϕ;
while *not converged* **do**
  Sample a skill $z \sim p(z)$ every episode;
  Collect new M on-policy samples;
  Update q_ϕ using K_1 steps of gradient descent on M transitions;
  Compute $r_z(s, a, s')$ for M transitions;
  Update π using any RL algorithm;
end

Model-based planning using skills



Algorithm 2: Latent Space Planner

$s \leftarrow s_0$;
Initialize parameters $\mu_1, \ldots \mu_{H_P}$;
for $i \leftarrow 1$ **to** H_E / H_Z **do**
  for $j \leftarrow 1$ **to** R **do**
    $\{z_i, \ldots z_{i+H_P-1}\}_{k=1}^{K} \sim \mathcal{N}_i, \ldots \mathcal{N}_{i+H_P-1}$;
    Compute r_{env} for $\{z_i, \ldots z_{i+H_P-1}\}_{k=1}^{K}$;
    Update $\mu_i, \ldots, \mu_{i+H_P-1}$;
  end
  Sample z_i from $\mathcal{N}_i$;
  Execute $\pi(a|s, z_i)$ for H_Z steps;
  Initialize μ_{i+H_P};
end

Resembling HRL, the controller plans skills instead of actions.

Experiments results

Ant navigation

Skill Variance analysis

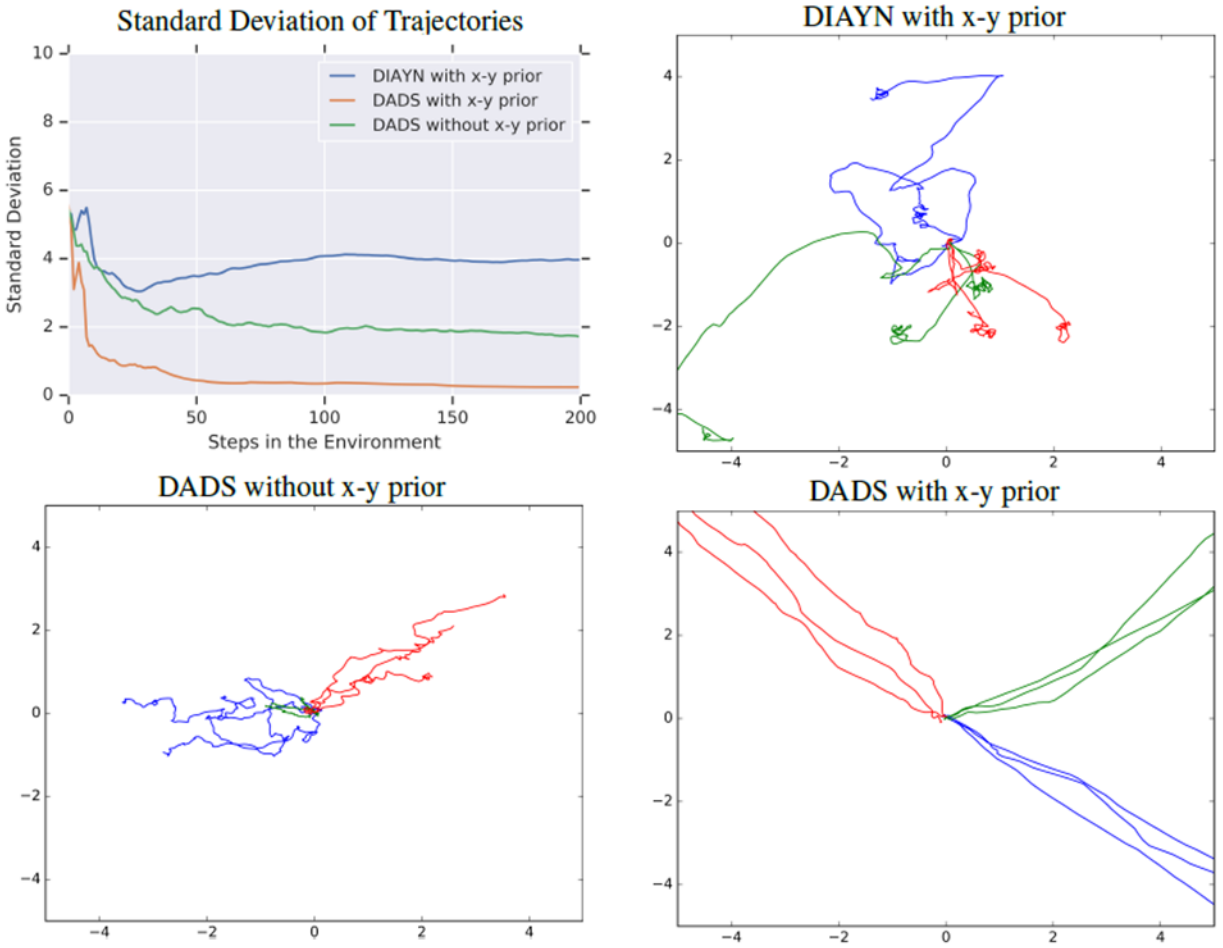


Figure 6: (Top-Left) Standard deviation of Ant's position as a function of steps in the environment, averaged over multiple skills and normalized by the norm of the position. (Top-Right to Bottom-Left Clockwise) X-Y traces of skills learned using DIAYN with x-y prior, DADS with x-y prior and DADS without x-y prior, where the same color represents trajectories resulting from the execution of the same skill z in the environment. High variance skills from DIAYN offer limited utility for hierarchical control.

Experiments results

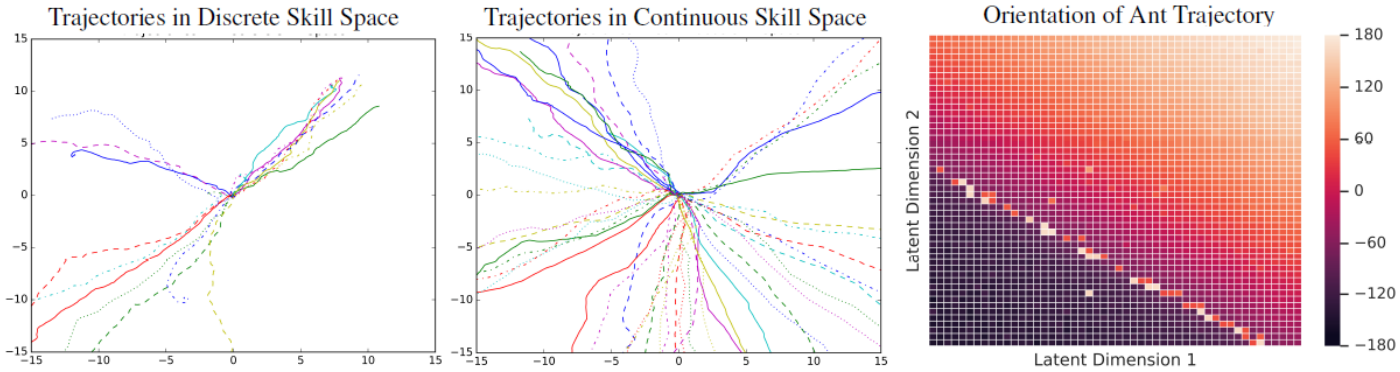


Figure 5: (Left, Centre) X-Y traces of Ant skills and (Right) Heatmap to visualize the learned continuous skill space. Traces demonstrate that the continuous space offers far greater diversity of skills, while the heatmap demonstrates that the learned space is smooth, as the orientation of the X-Y trace varies smoothly as a function of the skill.

Ant navigation

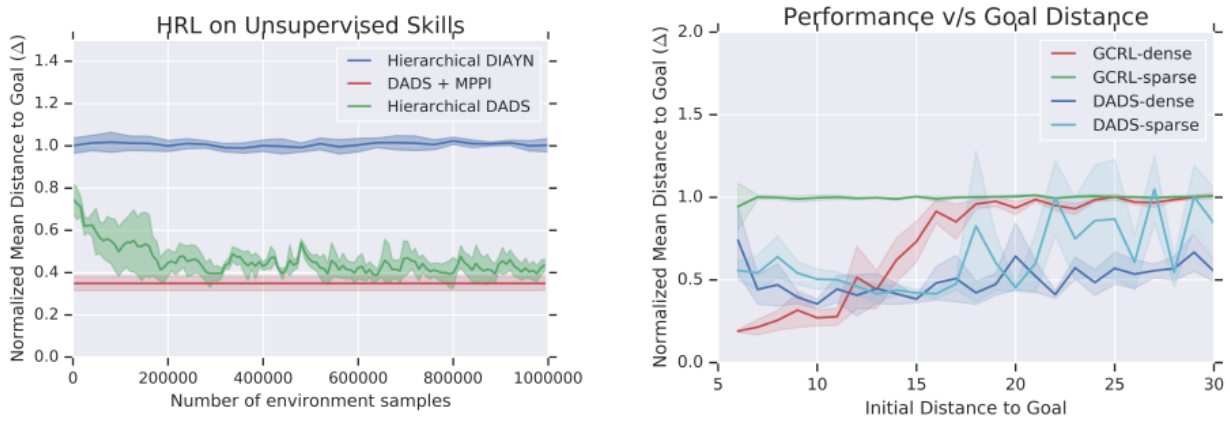


Figure 8: (Left) A RL-trained meta-controller is unable to compose primitive learned by DIAYN to navigate Ant to a goal, while it succeeds to do so using the primitives learned by DADS. (Right) Goal-Conditioned RL (GCRL-dense/sparse) does not generalize outside its training distribution, while MPPI controller on learned skills (DADS-dense/sparse) experiences significantly smaller degrade in performance.

Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Meta reinforcement learning

- **Forward transfer:** source domain to target domain
  - Diversity is good! The more varied the training, the more likely transfer is to succeed

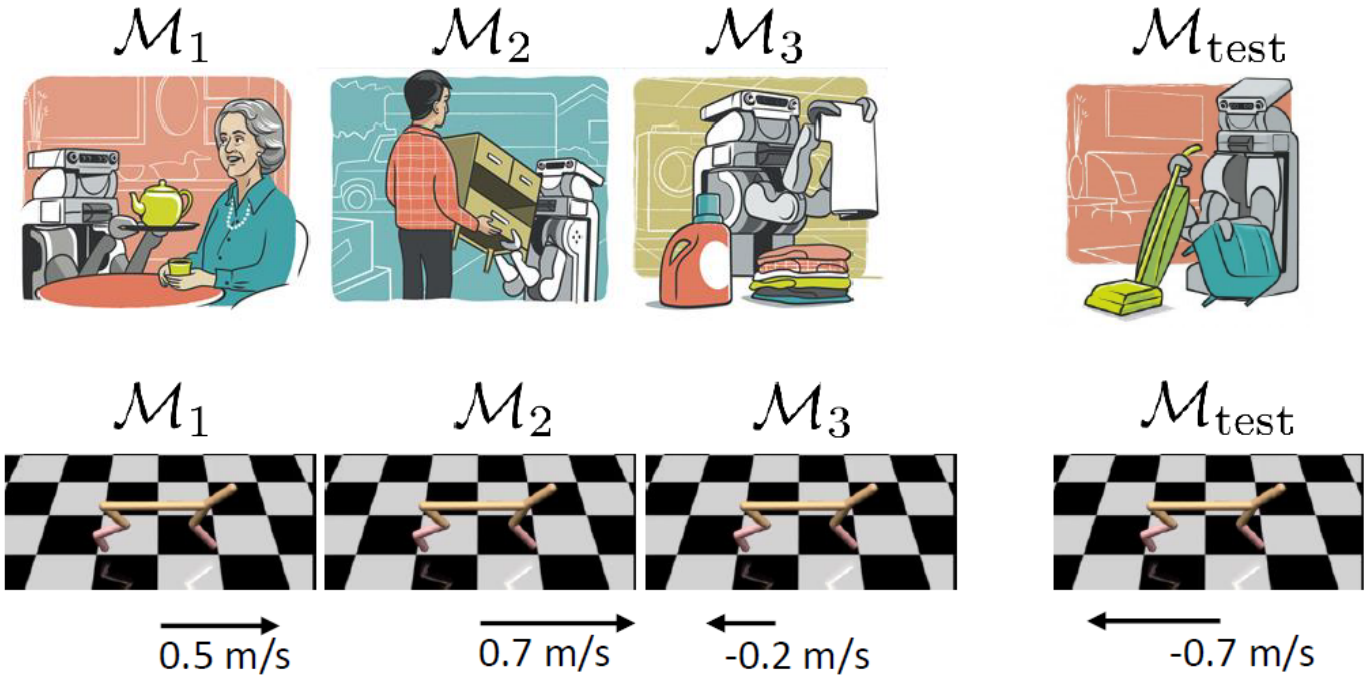
- **Multi-task learning:** even more variety
  - No longer training on the same kind of task
  - But more variety = more likely to succeed at transfer

- **How do we represent transfer knowledge**?
  - Model (as in model-based RL): rules of physics are conserved across tasks
  - Policies – requires finetuning, but closer to what we want to accomplish
  - What about *learning methods*?

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

What is meta reinforcement learning



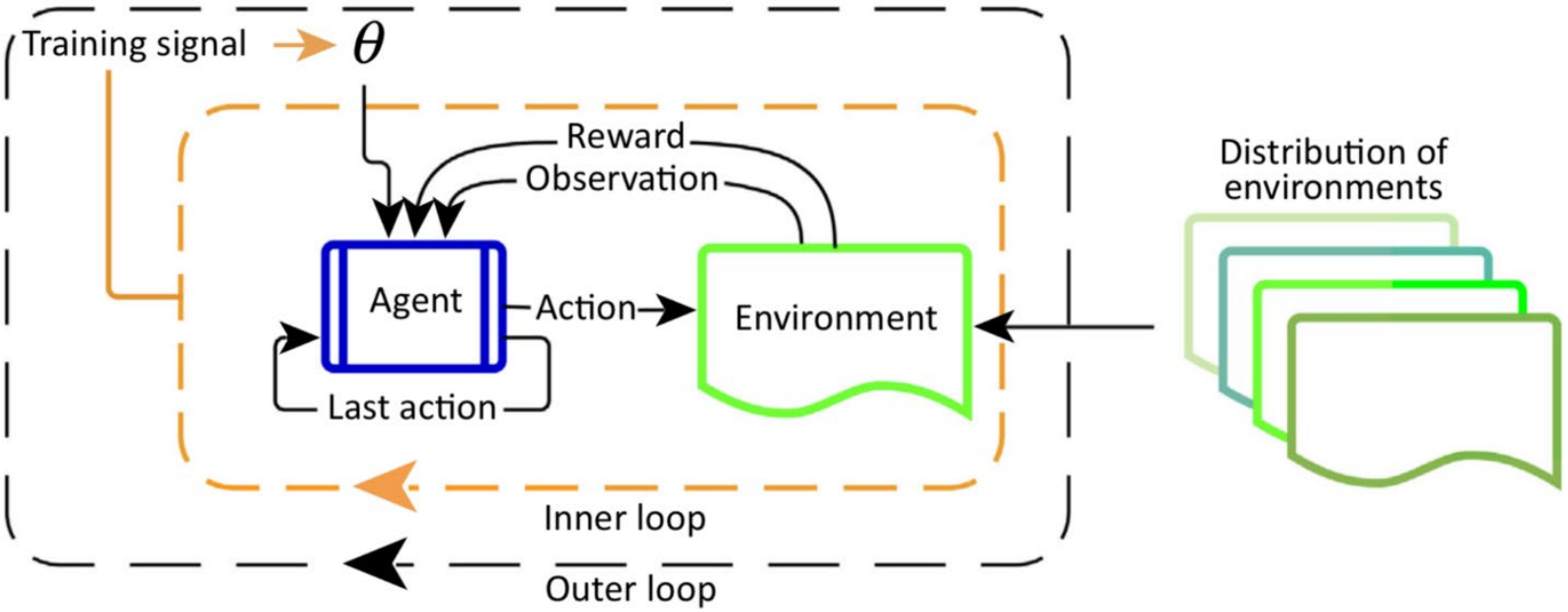
- *Meta-RL is meta-learning on reinforcement learning tasks. After trained over a distribution of tasks, the agent can quickly adapt to a new task by developing a new RL algorithm with its internal activity dynamics.*
- *The adaptation process, essentially a mini learning session, happens at test with limited exposure to the new configurations.*

Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Formulation of Meta Reinforcement Learning

We have a distribution of tasks, each formulation as an MDP, $M_i \in \mathcal{M}$.
An MDP is determined by a 4-tuple, $M_i =< \mathcal{S}, \mathcal{A}, P_i, R_i >$



1. The outer loop samples environments and adjusts θ that determine the agent' behavior;
2. In the inner loop, the agent interacts with the environment and optimizes for the maximal reward.

- Matthew Botvinick, et al. "Reinforcement Learning, Fast and Slow" Cell Review, Volume 23, Issue 5, P408-422, May 01, 2019.

On the origin of Meta-RL

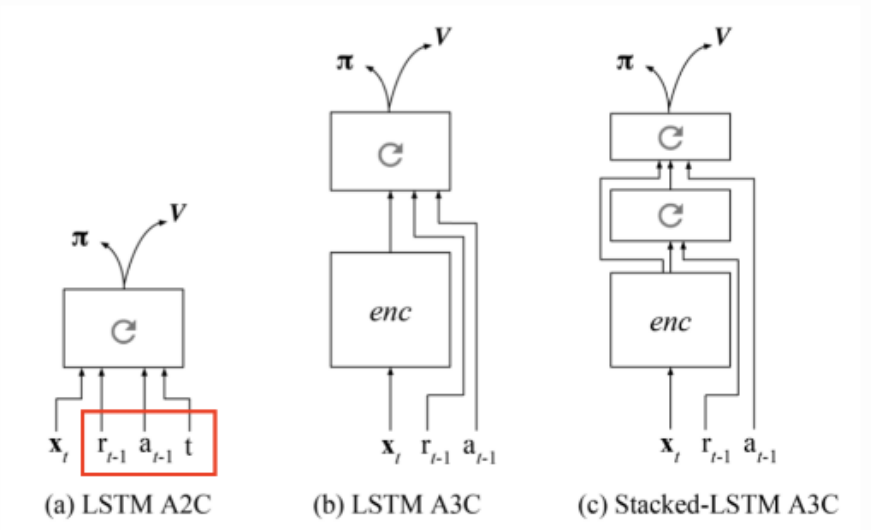
- **Main differences from RL**

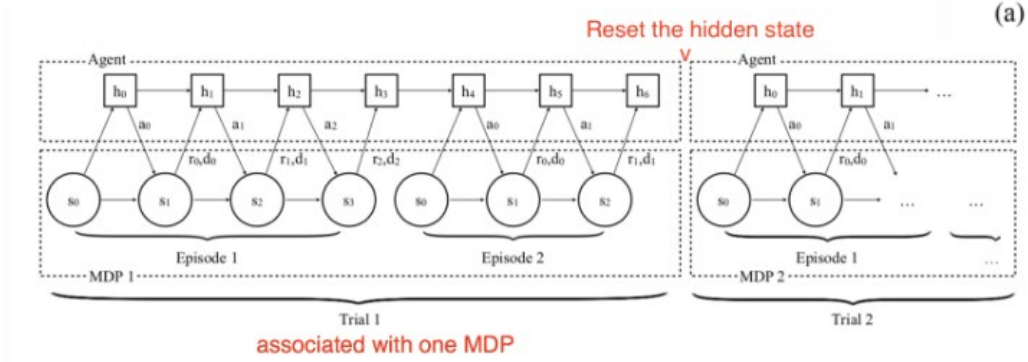
1. In RL, $\pi_\theta(s_t) \rightarrow$ a distribution over $\mathcal{A}$
2. In Meta-RL: $\pi_\theta(s_{t-1}, a_{t-1}, r_{t-1}, s_t) \rightarrow$ a distribution over $\mathcal{A}$

Feed histories into the model so that the policy can adjust its strategy according to the dynamics/MDP.

- **The training procedure works as follows:**

1. Sample a new MDP, $M_i \sim \mathcal{M}$
2. Reset the hidden state of the model;
3. Collect multiple trajectories and update the model weights
4. Repeat from step 1.





- Jane X Wang, et al. "Learning to reinforcement learn." arXiv preprint arXiv:1611.05763 (2016).
- Yan Duan, et al. "RL ^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." ICLR 2017.

Key Components of Meta-RL

- **A model with Memory**

  A recurrent neural network maintains a hidden state. Thus, it could acquire and memorize the knowledge about the current task by updating the hidden state during rollouts.

- **Meta-learning Algorithm**

  How we can update the model weights to optimize for the purpose of solving an unseen task fast at test time.

- **A Distribution of MDPs**

  While the agent is exposed to a variety of environments and tasks during training, it has to learn how to adapt to different MDPs.

Contents at a glance

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition

Model-Agnostic Meta-Learning

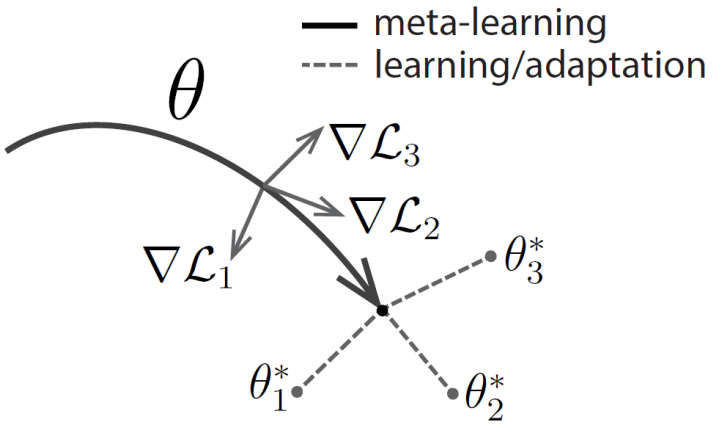
Optimizing model weights

Problem setup:

Given a task $\mathcal{T}$

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}$$

- $\mathcal{L}$: loss function
- x: observation, a: output
- $q(x)$: initial distribution of observations
- $q(x_{t+1}|x_t, a_t)$: transition function
- f_θ: model with parameters θ



One or more gradient update for a new task $\mathcal{T}_i$

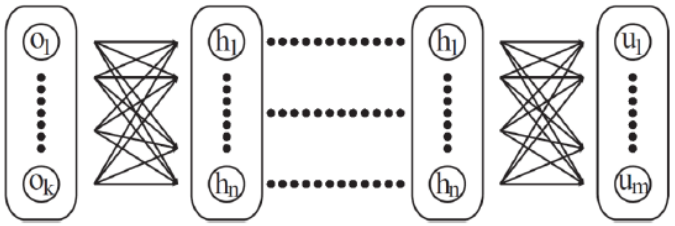
$$\theta_i' = \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

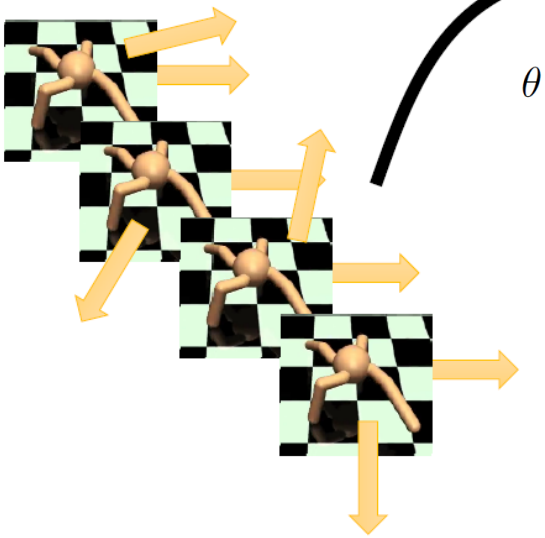
- Goal: To achieve a good generalization across a variety of tasks,

Chelsea Finn, Pieter Abbeel, and Sergey Levine. "**Model-agnostic meta-learning for fast adaptation of deep networks**." ICML 2017.

Model-Agnostic Meta-Learning

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$



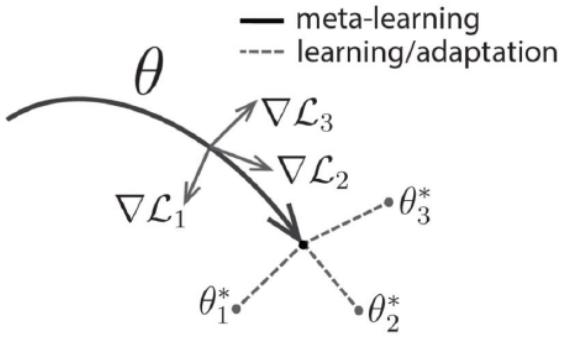


θ

~~$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$~~

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

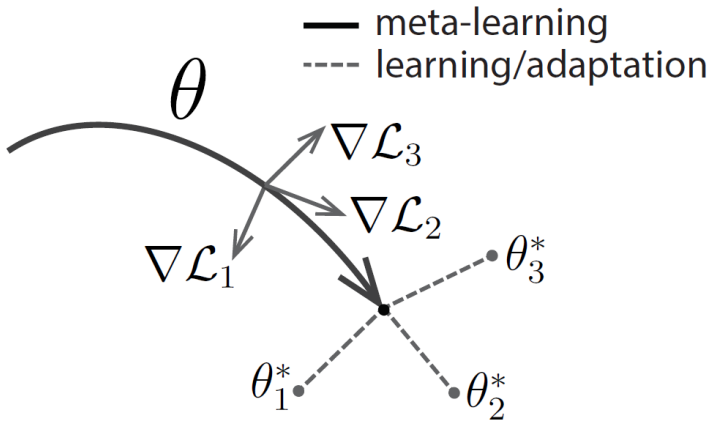
θ'



Model-Agnostic Meta-Learning

✓ The meta-optimization step above relies on **second derivatives.**

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}^{(0)}(f_\theta)$$



$$\theta^* = \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}^{(1)}(f_{\theta_i'}) = \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}^{(1)}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}^{(0)}(f_\theta)})$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}^{(1)}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}^{(0)}(f_\theta)})$$

Model-Agnostic Meta-Learning

✓ The meta-optimization step above relies on **second derivatives.**

Let's consider the case of performing k inner gradient steps, $k \geq 1$. Starting with the initial model parameter θ_{meta}:

$$
\begin{aligned}
\theta_0 &= \theta_{\text{meta}} \\
\theta_1 &= \theta_0 - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_0) \\
\theta_2 &= \theta_1 - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_1) \\
&\dots \\
\theta_k &= \theta_{k-1} - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_{k-1})
\end{aligned}
$$

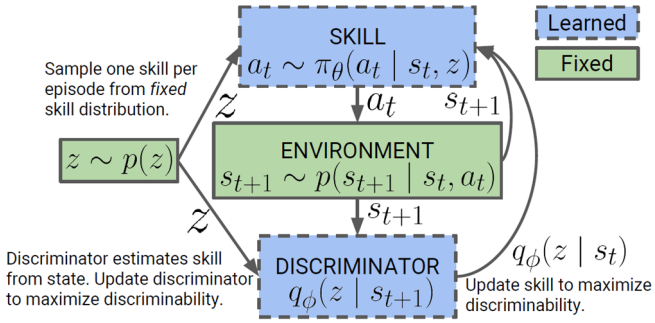
$$
\begin{aligned}
\theta_{\text{meta}} &\leftarrow \theta_{\text{meta}} - \beta g_{\text{MAML}} \\
g_{\text{MAML}} &= \nabla_\theta \mathcal{L}^{(1)}(\theta_k) \\
&= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot (\nabla_{\theta_{k-1}} \theta_k) \dots (\nabla_{\theta_0} \theta_1) \cdot (\nabla_\theta \theta_0) \\
&= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \Big(\prod_{i=1}^{k} \nabla_{\theta_{i-1}} \theta_i \Big) \cdot I \\
&= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \prod_{i=1}^{k} \nabla_{\theta_{i-1}} (\theta_{i-1} - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_{i-1})) \\
&= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \prod_{i=1}^{k} (I - \alpha \nabla_{\theta_{i-1}} (\nabla_\theta \mathcal{L}^{(0)}(\theta_{i-1})))
\end{aligned}
$$

$$
g_{\text{MAML}} = \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \prod_{i=1}^{k} (I - \alpha \nabla_{\theta_{i-1}} (\nabla_\theta \mathcal{L}^{(0)}(\theta_{i-1})))
$$

Training Task Acquisition

- Task Generation by Domain Randomization
- Learning with random rewards:
  - ✓ Sample random weights
  - ✓ Learn a discriminator function to encourage diversity-driven exploration

$$
\begin{aligned}
\mathcal{F}(\theta) &= I(S;Z) + H[A \mid S] - I(A;Z \mid S) \\
&= (H(Z) - H(Z \mid S)) + H[A \mid S] - (H[A \mid S] - H[A \mid S, Z]) \\
&= H[A \mid S, Z] - H(Z \mid S) + H(Z) \\
&= H[A \mid S, Z] + \mathbb{E}_{z \sim p(z), s \sim \rho(s)}[\log p(z \mid s)] - \mathbb{E}_{z \sim p(z)}[\log p(z)] \\
&\geq H[A \mid S, Z] + \mathbb{E}_{z \sim p(z), s \sim \rho(s)}[\log D_\phi(z \mid s) - \log p(z)]
\end{aligned}
$$



- Abhishek Gupta, et al."Unsupervised meta-learning for Reinforcement Learning" arXiv preprint arXiv:1806.04640 (2018).

Suggested readings

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,

Recap

- Preliminary
- Transfer Learning for RL
  1. Forward transfer
     a. Domain adaptation for supervised learning
     b. Domain adaptation for RL
     c. Domain randomization
  2. Transferring models and value functions
     a. Model-based RL & System identification
     b. Transfer value function
  3. Multi-task transfer
- Meta Reinforcement Learning
  1. Origin & formulation of Meta RL
  2. Meta-learning algorithm
  3. Task acquisition