Transfer and Meta Learning

- **Last time:**
  - Imitation Learning
  - Inverse Reinforcement Learning
- **Today:**

(Lecture content from Sergey Levine's CS 285 at Berkeley)

Transfer and Meta Learning

- **Last time:**
  - Imitation Learning
  - Inverse Reinforcement Learning
- **Today:**
  - How do we **transfer** knowledge from one domain to another? (e.g. simulated to real-world)

(Lecture content from Sergey Levine's CS 285 at Berkeley)

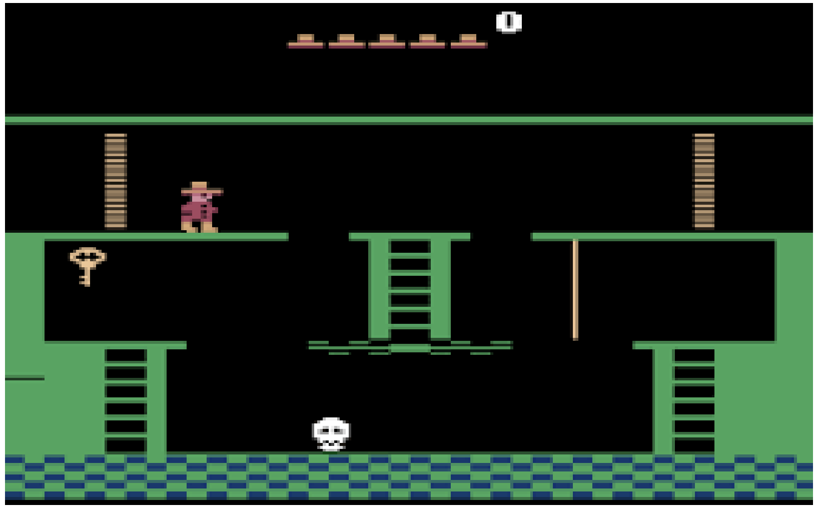
Transfer and Meta Learning

- **Last time:**
  - Imitation Learning
  - Inverse Reinforcement Learning
- **Today:**
  - How do we **transfer** knowledge from one domain to another? (e.g. simulated to real-world)
  - How do we learn how to learn? (**Meta** learning)

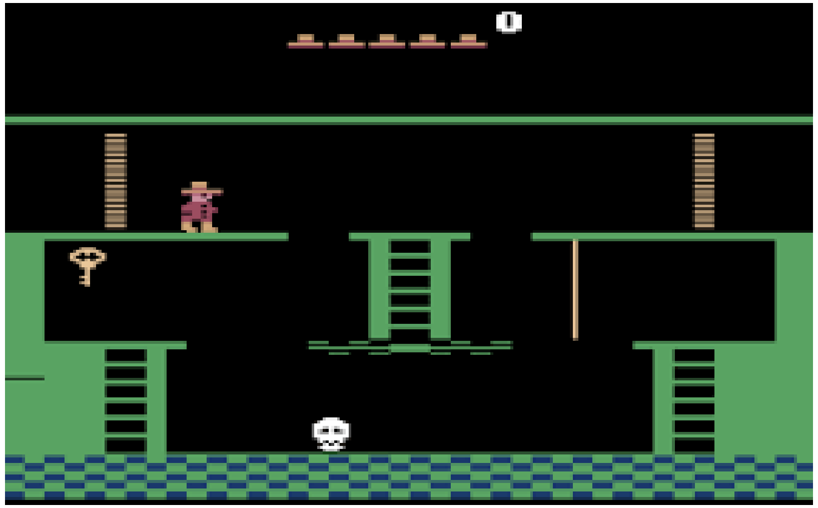
(Lecture content from Sergey Levine's CS 285 at Berkeley)

Transfer Learning and Montezuma's Revenge

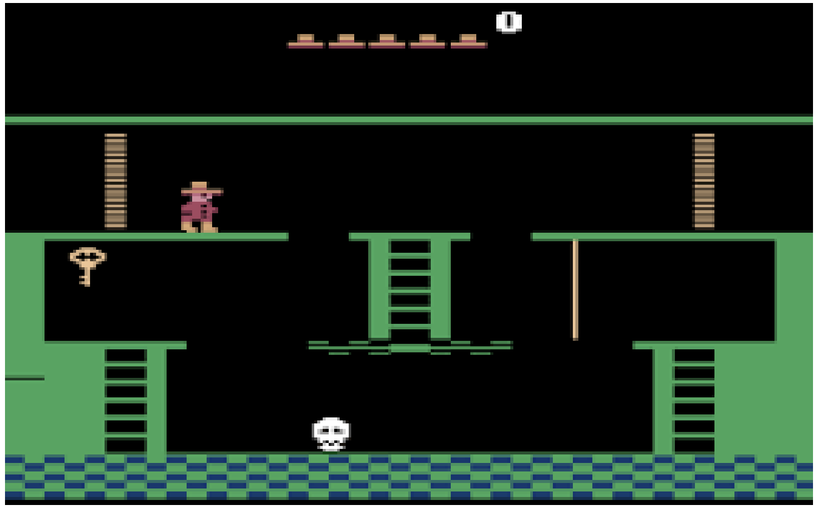
Transfer Learning and Montezuma's Revenge



Transfer Learning and Montezuma's Revenge



Transfer Learning and Montezuma's Revenge



Could an RL agent be better at Montezuma's revenge after watching Indiana Jones?

Transfer Learning

Transfer Learning

Transfer Learning: Use experience from one set of tasks for faster learning and better performance on a new task

Transfer Learning

Transfer Learning: Use experience from one set of tasks for faster learning and better performance on a new task

In RL, task=MDP

Transfer Learning

Transfer Learning: Use experience from one set of tasks for faster learning and better performance on a new task

In RL, task=MDP

Source domain $\rightarrow$ target domain

Transfer Learning

Transfer Learning: Use experience from one set of tasks for faster learning and better performance on a new task

In RL, task=MDP

Source domain $\rightarrow$ target domain

- "shot" = number of attempts in the target domain
- "0-shot" = run policy in target domain
- "1-shot" = try task once
- "few shot"

Transfer Learning

How should prior knowledge be stored?

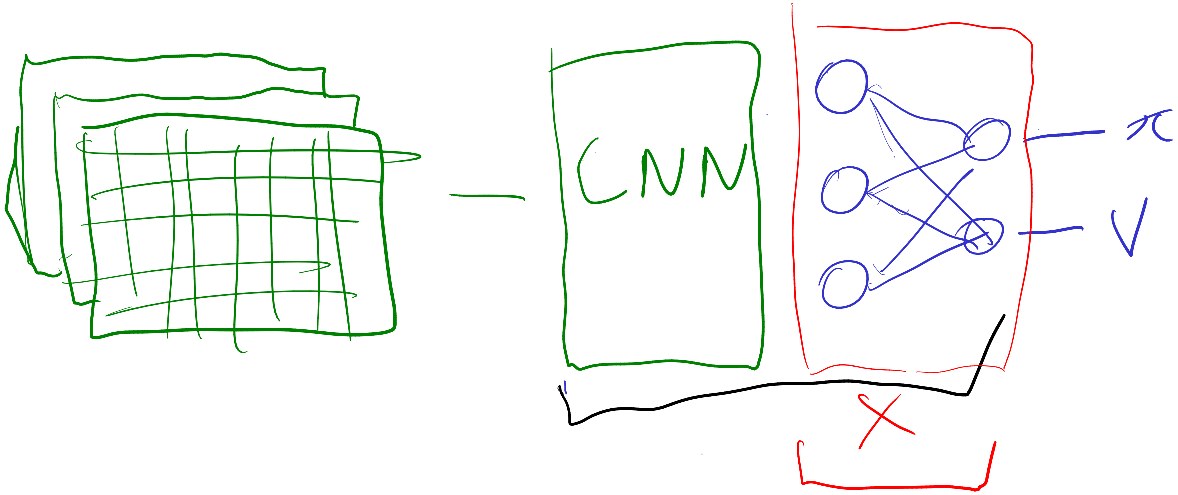
- Q-function
- Policy
- Model
- Features/hidden states

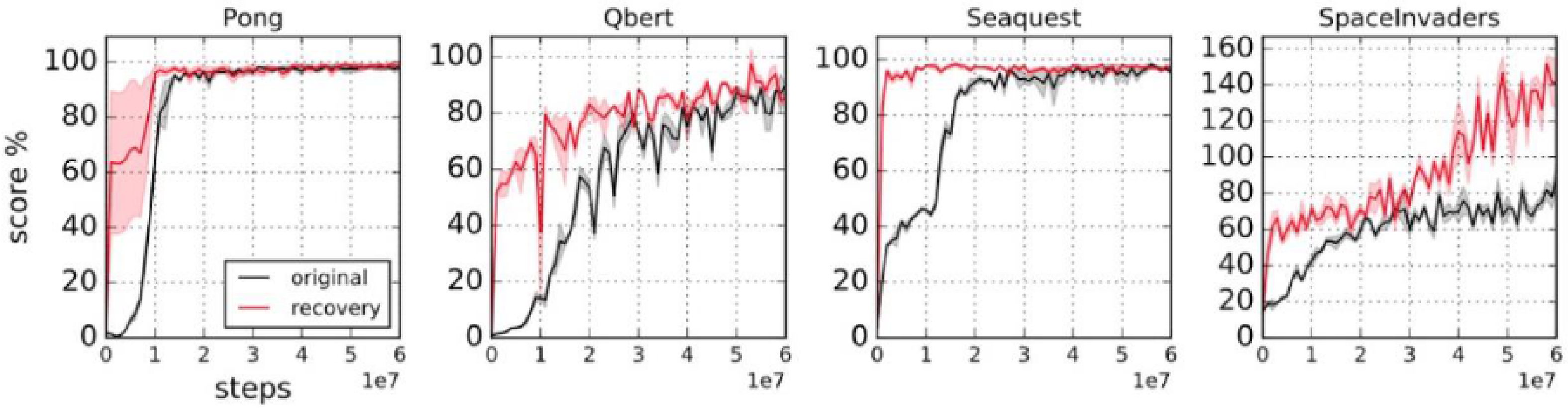
Transfer Learning

How should prior knowledge be stored?

- Q-function
- Policy
- Model
- Features/hidden states

Representation Bottleneck





Transfer Learning

How should prior knowledge be stored?

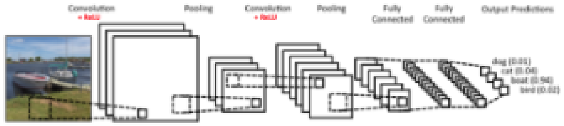
- Q-function
- Policy
- Model
- Features/hidden states

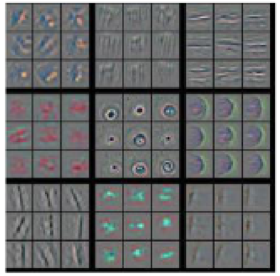
Transfer Learning

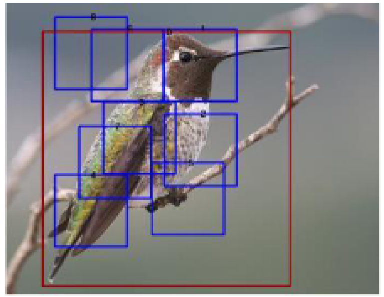
How should prior knowledge be stored?

- Q-function
- Policy
- Model
- Features/hidden states

Pretraining + Finetuning







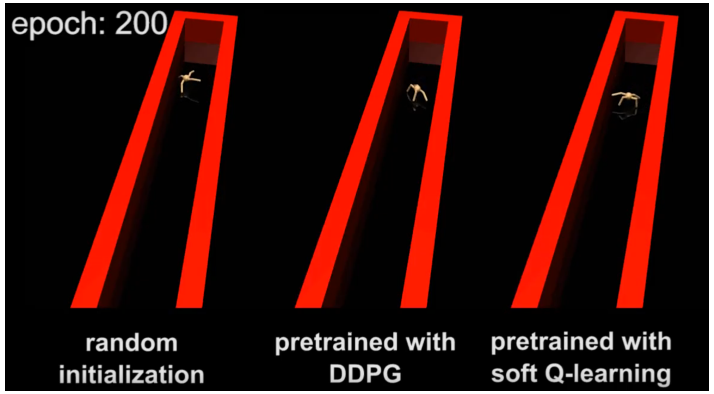
Pretraining + Finetuning

Pretraining + Finetuning

Pretrain: reward speed in any direction

Pretraining + Finetuning

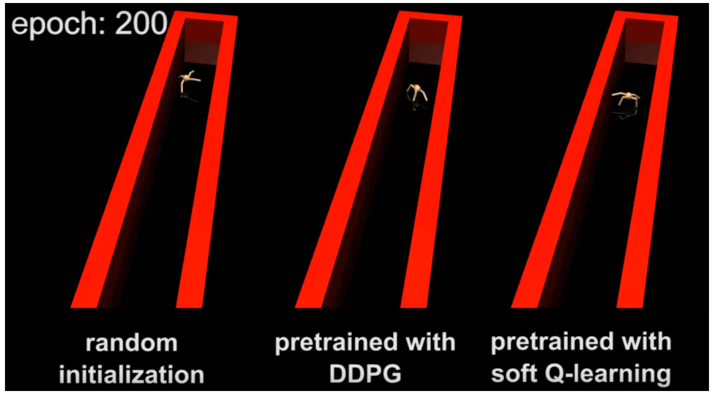
Pretrain: reward speed in any direction



Fine Tune: reward speed in specific direction

Pretraining + Finetuning

Pretrain: reward speed in any direction



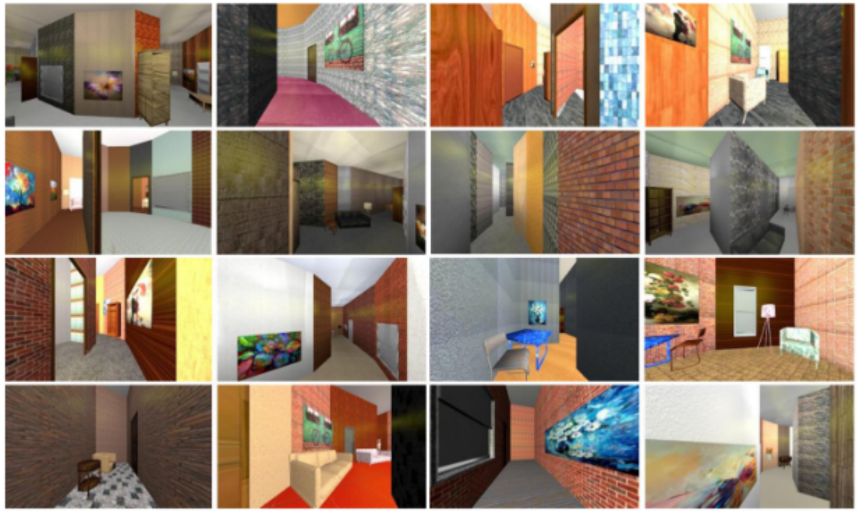
Fine Tune: reward speed in specific direction

Both reward + entropy

$$\pi(a|s) \propto_a \exp(Q(s,a))$$

https://sites.google.com/view/softqlearning/home

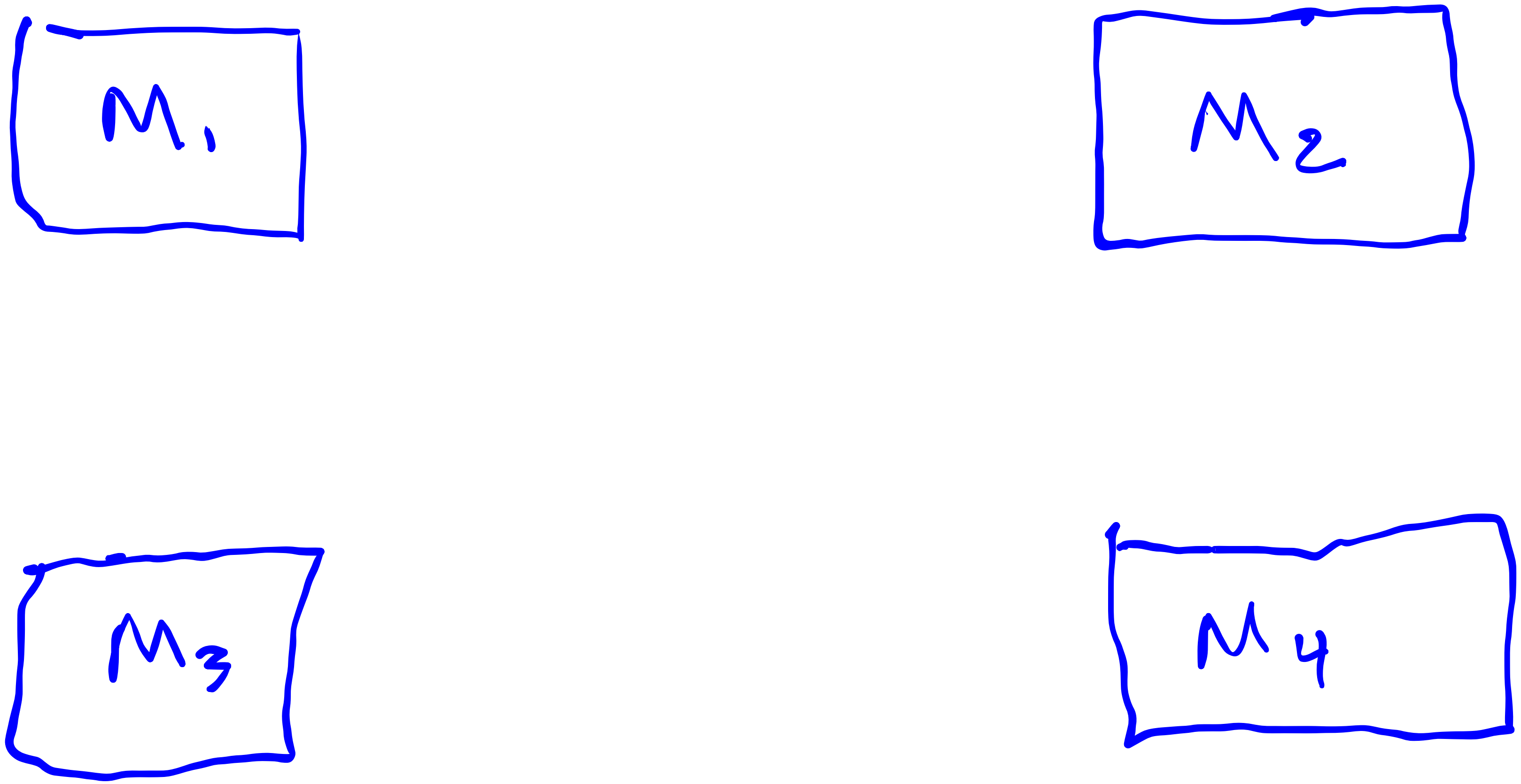
CAD2RL



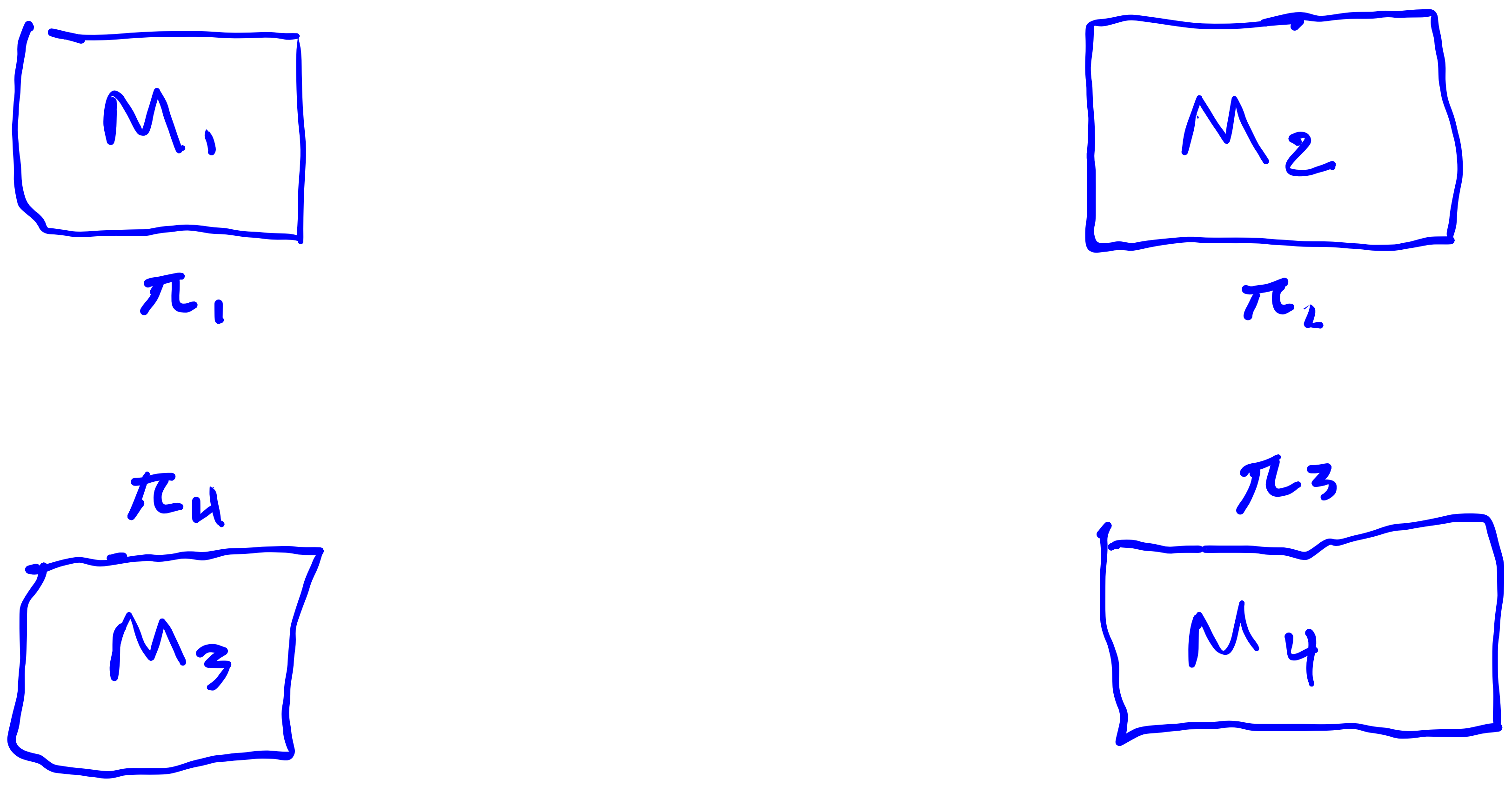
Key: Diversity

Actor Mimic

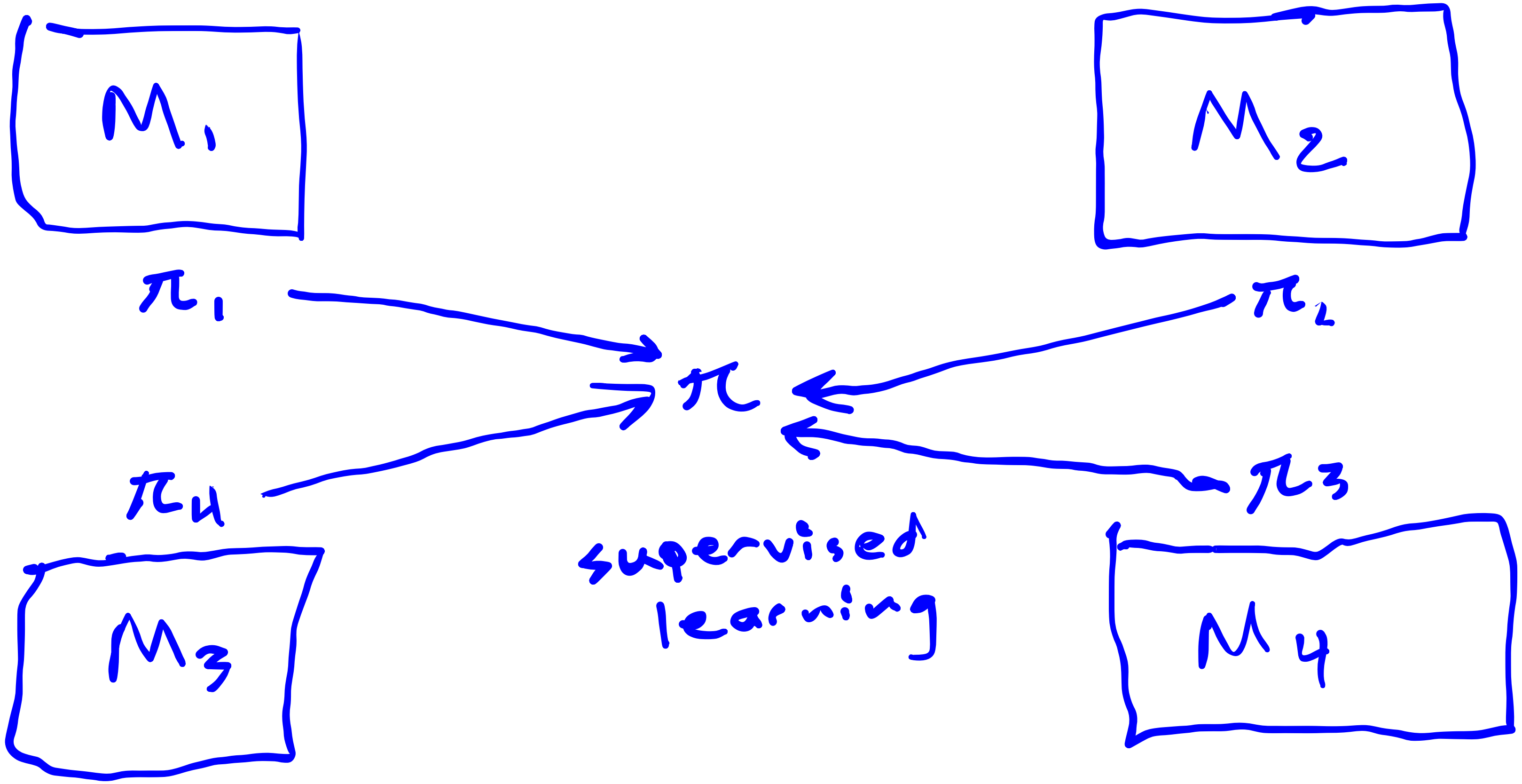
Actor Mimic



Actor Mimic



Actor Mimic



Transfer Learning

How should prior knowledge be stored?

- Q-function
- Policy
- Model
- Features/hidden states

Transfer Learning

How should prior knowledge be stored?

- Q-function
- Policy
- Model
- Features/hidden states

Successor Features

Successor Features

All domains have same S, A, T, γ

Difference: R

Successor Features

All domains have same S, A, T, γ

Difference: R

Let $R(s, a) = w^\top \phi(s, a)$ where ϕ is a feature vector.

Successor Features

All domains have same S, A, T, γ

Difference: R

Let $R(s,a) = w^\top \phi(s,a)$ where ϕ is a feature vector.

$$Q^\pi(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s,a) \mid s_0 = s, a_0 = a\right]$$

Successor Features

All domains have same S, A, T, γ

Difference: R

Let $R(s,a) = w^\top \phi(s,a)$ where ϕ is a feature vector.

$$
\begin{aligned}
Q^\pi(s,a) &= E\left[\sum_{t=0}^{\infty} \gamma^t R(s,a) \mid s_0 = s, a_0 = a\right] \\
&= E\left[\sum_{t=0}^{\infty} \gamma^t w^\top \phi(s,a) \mid s_0 = s, a_0 = a\right]
\end{aligned}
$$

Successor Features

All domains have same S, A, T, γ

Difference: R

Let $R(s,a) = w^\top \phi(s,a)$ where ϕ is a feature vector.

$$
\begin{aligned}
Q^\pi(s,a) &= E\left[\sum_{t=0}^{\infty} \gamma^t R(s,a) \mid s_0 = s, a_0 = a\right] \\
&= E\left[\sum_{t=0}^{\infty} \gamma^t w^\top \phi(s,a) \mid s_0 = s, a_0 = a\right] \\
&= w^\top E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right]
\end{aligned}
$$

Successor Features

All domains have same S, A, T, γ

Difference: R

Let $R(s,a) = w^\top \phi(s,a)$ where ϕ is a feature vector.

$$
\begin{aligned}
Q^\pi(s,a) &= E\left[\sum_{t=0}^{\infty} \gamma^t R(s,a) \mid s_0 = s, a_0 = a\right] \\
&= E\left[\sum_{t=0}^{\infty} \gamma^t w^\top \phi(s,a) \mid s_0 = s, a_0 = a\right] \\
&= w^\top E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right]
\end{aligned}
$$

Successor Feature:

$$
\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right]
$$

Successor Features

All domains have same S, A, T, γ

Difference: R

Let $R(s,a) = w^\top \phi(s,a)$ where ϕ is a feature vector.

$$\begin{aligned} Q^\pi(s,a) &= E\left[\sum_{t=0}^{\infty} \gamma^t R(s,a) \mid s_0 = s, a_0 = a\right] \\ &= E\left[\sum_{t=0}^{\infty} \gamma^t w^\top \phi(s,a) \mid s_0 = s, a_0 = a\right] \\ &= w^\top E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \end{aligned}$$

Successor Feature:

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right]$$

$$Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right]$$

$$Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty}\gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \qquad Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \qquad Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

$$Q'^\pi = w'^\top \psi^\pi$$

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right]$$

$$Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

$$Q'^\pi = w'^\top \psi^\pi$$

Important: Does this yield optimal policy for R'?

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \qquad Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

$$Q'^\pi = w'^\top \psi^\pi$$

Important: Does this yield optimal policy for R'?

No!

$$Q^\pi(s,a) = R(s,a) + \gamma E[Q^\pi(s', \pi(s'))]$$

$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} E[Q^*(s', a')]$$

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \qquad Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

$$Q'^\pi = w'^\top \psi^\pi$$

Important: Does this yield optimal policy for R'?

No!

$$Q^\pi(s,a) = R(s,a) + \gamma E[Q^\pi(s', \pi(s'))]$$

$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} E[Q^*(s',a')]$$

How to use this in practice:

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \qquad Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

$$Q'^\pi = w'^\top \psi^\pi$$

Important: Does this yield optimal policy for R'?

No!

$$Q^\pi(s,a) = R(s,a) + \gamma E[Q^\pi(s', \pi(s'))]$$
$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} E[Q^*(s', a')]$$

How to use this in practice:

- Keep a family of good policies and associated successor features with a variety of weights.

Using successor features

$$\psi^\pi(s,a) \equiv E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s,a) \mid s_0 = s, a_0 = a\right] \qquad Q^\pi(s,a) = w^\top \psi^\pi(s,a)$$

Given ψ^π, one can easily calculate Q'^π for a new reward function $R' = w'^\top \phi$.

$$Q'^\pi = w'^\top \psi^\pi$$

Important: Does this yield optimal policy for R'?

No!

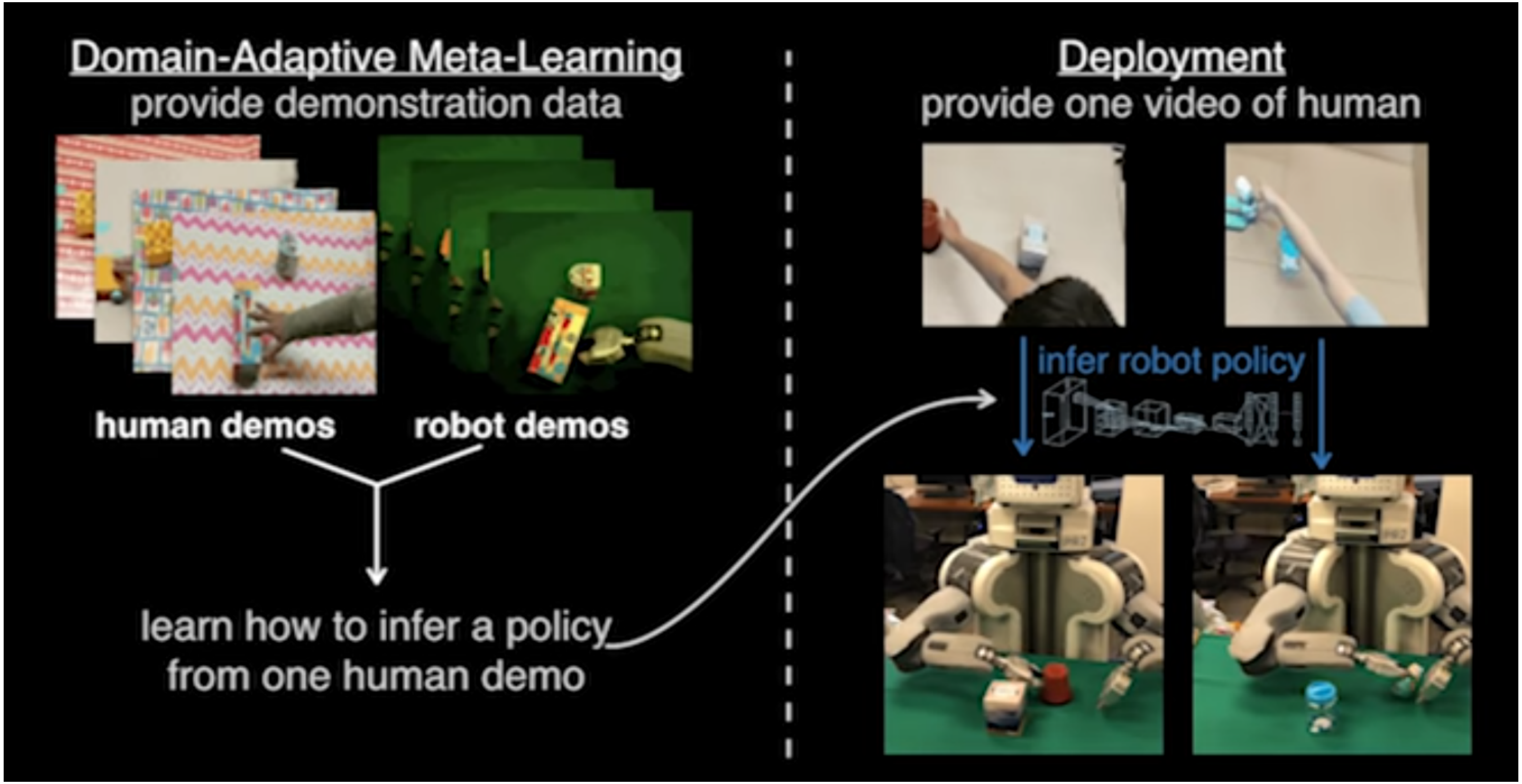
$$Q^\pi(s,a) = R(s,a) + \gamma E[Q^\pi(s', \pi(s'))]$$

$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} E[Q^*(s', a')]$$

How to use this in practice:

- Keep a family of good policies and associated successor features with a variety of weights.
- In target domain, start with best policy from this set and finetune/plan online

Meta Learning: Motivation



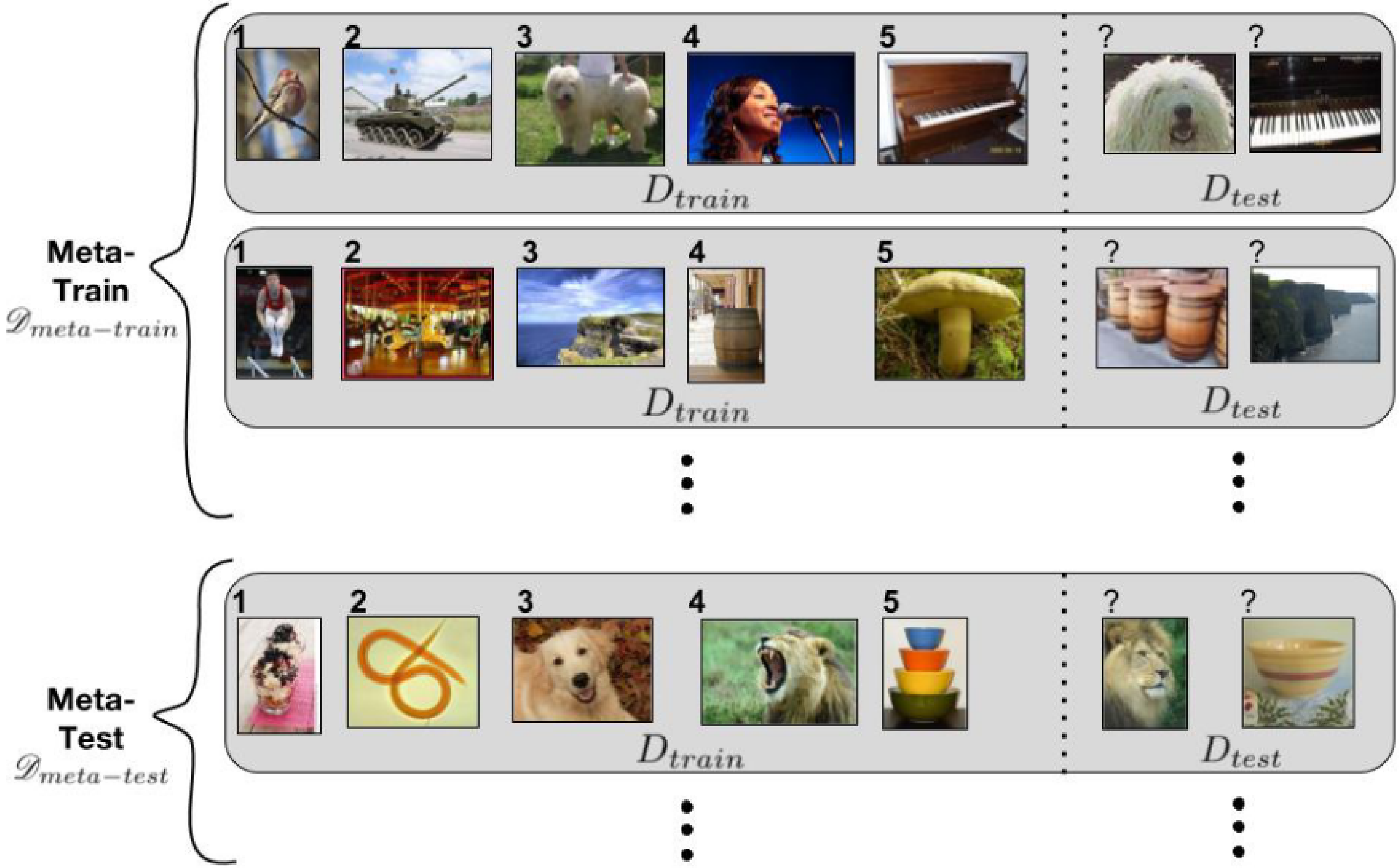
https://www.youtube.com/watch?v=1eYqV_vGlJY

Meta Learning

Meta Learning

Machine
Learning
Data Set

Meta Learning



Meta Reinforcement Learning

Meta Reinforcement Learning

RL

Meta Reinforcement Learning

RL

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \, \mathrm{E}_{\pi_\theta}[R(\tau)]$$

Meta Reinforcement Learning

RL

$$\theta^* = \arg\max_{\theta} \mathrm{E}_{\pi_\theta}[R(\tau)]$$

$$= f_{\mathrm{RL}}(M)$$

Meta Reinforcement Learning

RL

$$\theta^* = \arg\max_{\theta} \mathrm{E}_{\pi_\theta}[R(\tau)]$$

$$= f_{\mathrm{RL}}(M)$$

Meta RL

Meta Reinforcement Learning

RL

$$\theta^* = \arg\max_{\theta} \mathrm{E}_{\pi_\theta}[R(\tau)]$$

$$= f_{\mathrm{RL}}(M)$$

Meta RL

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} \mathrm{E}_{\pi_{\phi_i}}[R_i(\tau)]$$

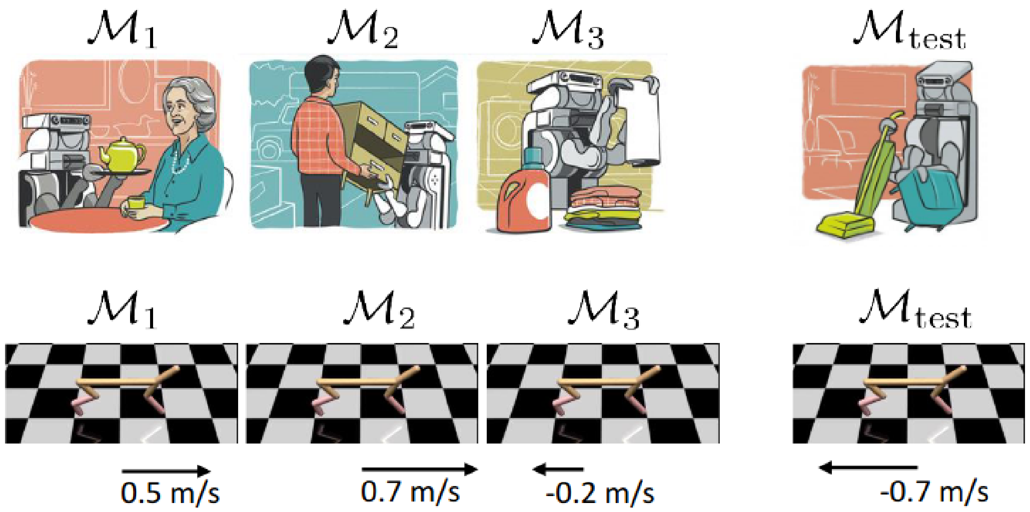


Image: Sergey Levine CS285 slides

Meta Reinforcement Learning

RL

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathrm{E}_{\pi_\theta}[R(\tau)]$$

$$= f_{\mathrm{RL}}(M)$$

Meta RL

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \mathrm{E}_{\pi_{\phi_i}}[R_i(\tau)]$$

$$\text{where } \phi_i = f_\theta(M_i)$$

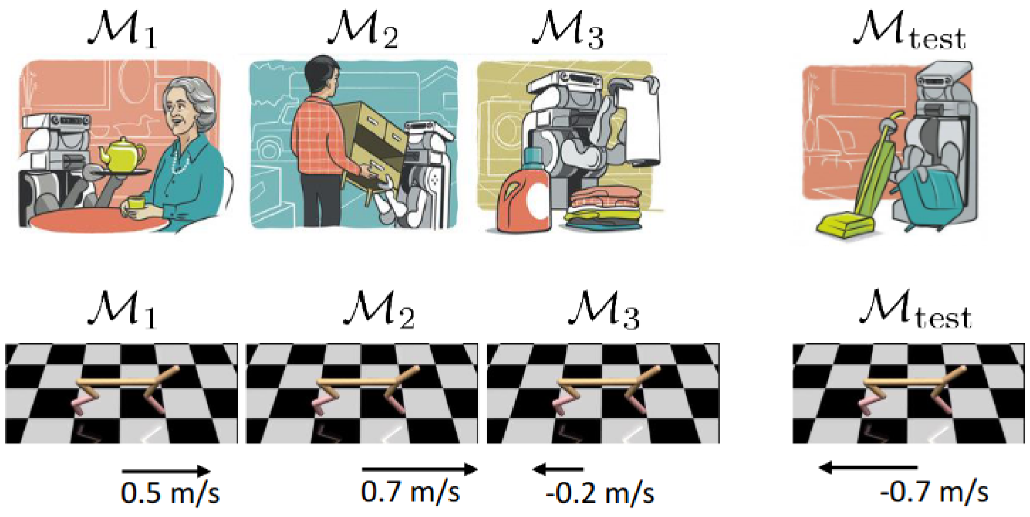


Image: Sergey Levine CS285 slides

Meta Reinforcement Learning

RL

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \, \mathrm{E}_{\pi_\theta}[R(\tau)]$$

$$= f_{\mathrm{RL}}(M)$$

Meta RL

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \mathrm{E}_{\pi_{\phi_i}}[R_i(\tau)]$$

$$\text{where } \phi_i = f_\theta(M_i)$$

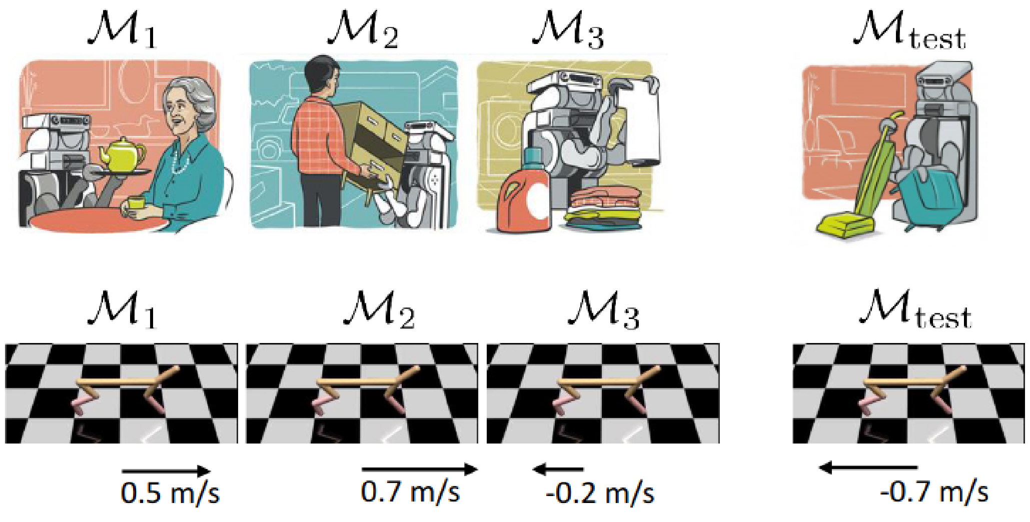


Image: Sergey Levine CS285 slides

Important: Exploration can speed up Meta RL

Meta Reinforcement Learning

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

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

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

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

Model Agnostic Meta Learning (MAML) for RL

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

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

Model Agnostic Meta Learning (MAML) for RL

$$f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

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

Model Agnostic Meta Learning (MAML) for RL

$$f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

Meta Policy Gradient

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

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

Model Agnostic Meta Learning (MAML) for RL

$$f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

Meta Policy Gradient

$$\theta \leftarrow \theta + \beta \sum_i \nabla_\theta J_i[\theta + \alpha \nabla_\theta J_i(\theta)]$$

Meta Reinforcement Learning

Approach 2: Gradient-Based Meta-RL (MAML)

RL: Policy Gradient

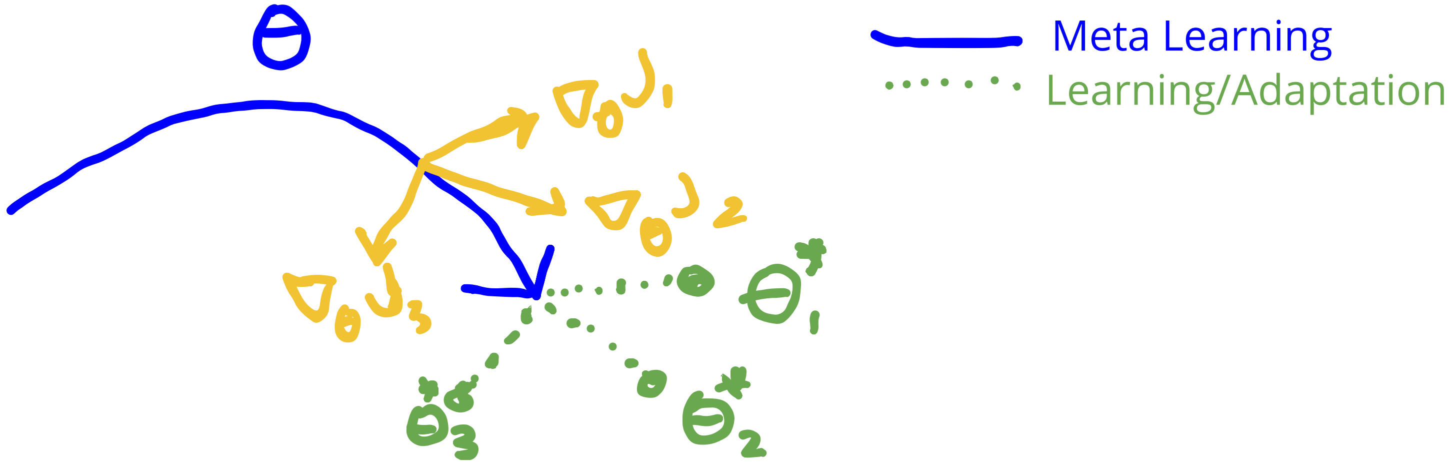
$$\theta^{k+1} \leftarrow \theta^k + \alpha \nabla_\theta J(\theta^k)$$

Model Agnostic Meta Learning (MAML) for RL

$$f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

Meta Policy Gradient

$$\theta \leftarrow \theta + \beta \sum_i \nabla_\theta J_i[\theta + \alpha \nabla_\theta J_i(\theta)]$$



Recap

Recap

- In **Transfer Learning**, the goal is to use training from one or more **source domains** to one or more **target domains**.

Recap

- In **Transfer Learning**, the goal is to use training from one or more **source domains** to one or more **target domains**.
- Various methods exist to transfer via knowledge stored in the **policy**, **model**, **value function**, or **other features**.

Recap

- In **Transfer Learning**, the goal is to use training from one or more **source domains** to one or more **target domains**.
- Various methods exist to transfer via knowledge stored in the **policy**, **model**, **value function**, or **other features**.
- In **Meta Learning**, the goal is to learn how to master a new environment quickly.

Recap

- In **Transfer Learning**, the goal is to use training from one or more **source domains** to one or more **target domains**.
- Various methods exist to transfer via knowledge stored in the **policy**, **model**, **value function**, or **other features**.
- In **Meta Learning**, the goal is to learn how to master a new environment quickly.
- A meta learning problem can be posed as a **POMDP**.

Recap

- In **Transfer Learning**, the goal is to use training from one or more **source domains** to one or more **target domains**.
- Various methods exist to transfer via knowledge stored in the **policy**, **model**, **value function**, or **other features**.
- In **Meta Learning**, the goal is to learn how to master a new environment quickly.
- A meta learning problem can be posed as a **POMDP**.
- In model agnostic meta learning (**MAML**), the policy is parameterized so that **one gradient step** in the new environment will produce a good policy.