Meta Reinforcement Learning

SiDNN 01.03.2022

Jingyu Liu

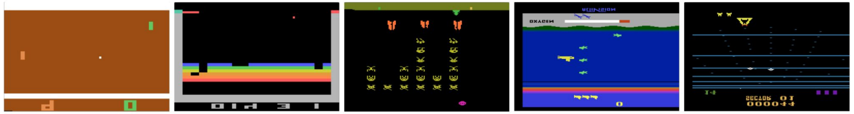
Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

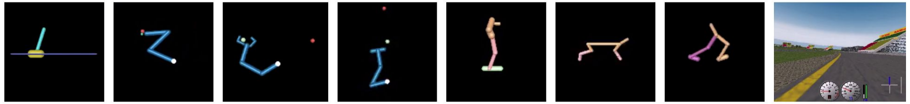
Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

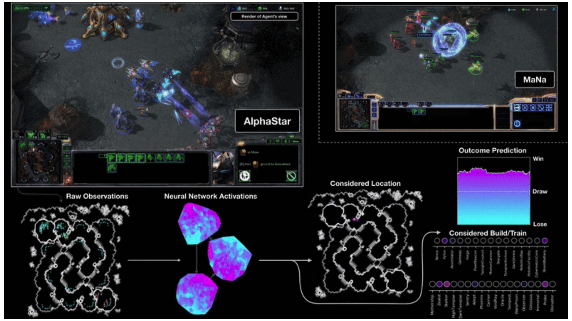
Why do we care about meta-rl?



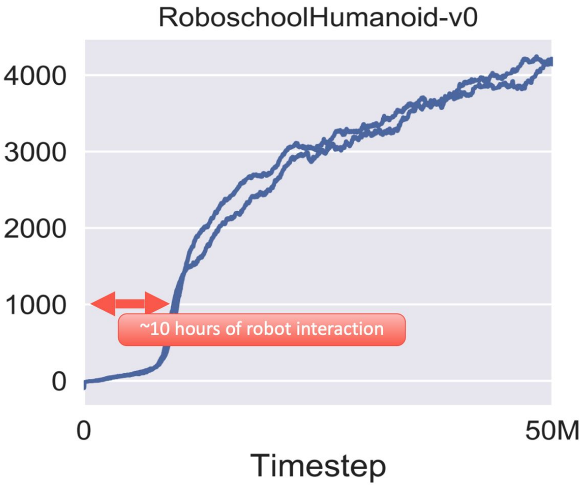
Mnih et al. **Playing Atari with Deep Reinforcement Learning** [2013]



Lillicrap et al. **Continuous Control with Deep RL** [2015]



Deepmind. **Grandmaster level in StarCraft II using multi-agent reinforcement learning** [2019]



graph: Schulman et al. '17

- Humans can learn new skills very **quickly**, efficiently **adapting** to new environments and tasks.
- Can we design algorithms that **learn to reinforcement learn**?

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
    - Black-box adaptation (based on recurrent policies)
    - Optimization-based methods
    - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

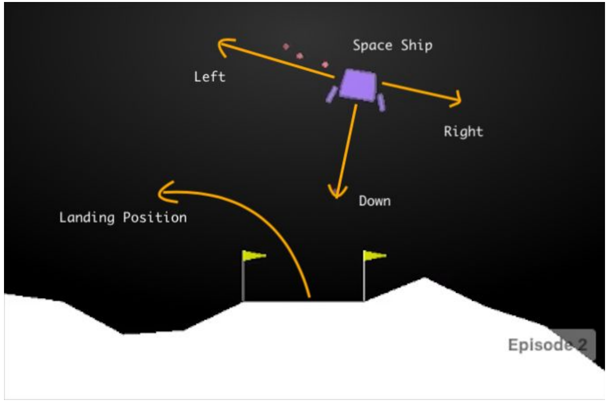
The reinforcement learning problem

Markov Decision Process : $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}$

$\mathcal{S}$: *state space*
$\mathcal{A}$: *action space*
$\mathcal{P}$: *transition probability*, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
r : *reward function*, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

$\pi(a|s)$: *the policy*, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ *or* $\pi : \mathcal{S} \to \mathcal{A}$

Transitions : $\{s_t, a_t, r_t, s_{t+1}\}_i$
Trajectory : $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T\}$



The reinforcement learning problem

Goal :
learn a policy that **maximizes the expected (discounted) sum of rewards**

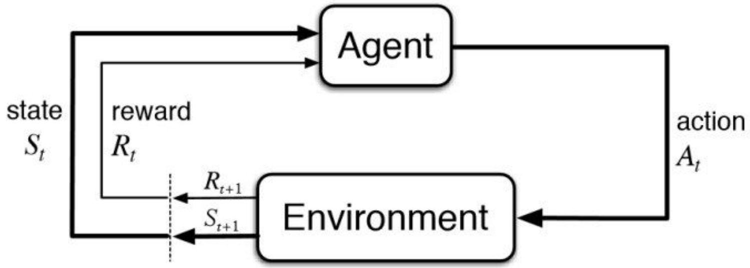
Parameterized policy (*infinite horizon*) :

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^{\mathrm{t}} r(s_t, a_t)\right]$$

Expectation over (*discounted*) *state visitation distribution*

$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta(\mathbf{s},\mathbf{a})}[r(s, a)]$

General procedure



How do we optimize our policy?

1. **policy gradient**
2. *value function or Q function estimation*
3. *model learning + MPC*

1. *get initial state* $s_0 \sim p(s)$
2. *choose an action from policy* $a_t \sim \pi_\theta(\cdot|s_t)$
3. *observe reward* $r_t = r(s_t, a_t)$ *and new state* $s_{t+1} \sim p(\cdot|s_t, a_t)$
4. *optimize* $\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}[R(\tau)]$
5. *store experiences* (s_t, a_t, s_{t+1}, r_t) *in replay buffer*
6. *repeat until convergence*

The meta learning problem

θ : *meta parameter*
ϕ_i : *task specific adaptation parameter*
$p(\mathfrak{D})$: *a distribution over meta training dataset (or tasks)*

$$\textit{learn } \theta \textit{ such that } \phi_i = f_\theta\left(\mathfrak{D}_i^{tr}\right) \textit{ fits } \mathfrak{D}_i^{ts} \textit{ well}$$

Probabilistic view :

$$\theta^* = \arg\max_\theta \sum logp\left(\phi_i | \mathfrak{D}_i^{ts}\right)$$

Deterministic view :

$$\theta^* = \arg\min_\theta \sum \mathcal{L}_i\left(\phi_i, \mathfrak{D}_i^{ts}\right)$$

Meta learning + RL

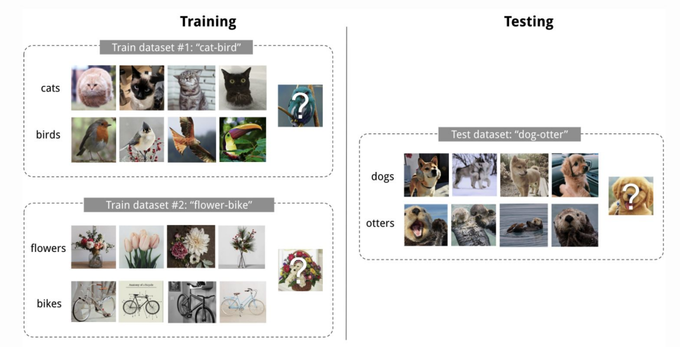
Traditional (supervised) learning :

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

Traditional (supervised) meta learning :

$$\theta^* = \arg\min_\theta \sum \mathcal{L}_i\left(\phi_i, \mathcal{D}_i^{ts}\right) \textit{ where } \phi_i = f_\theta\left(\mathcal{D}_i^{tr}\right)$$

$$\mathcal{D}_{meta-train} = \left\{\left(D_0^{tr}, D_0^{ts}\right), \left(D_1^{tr}, D_1^{ts}\right), \ldots\right\}$$



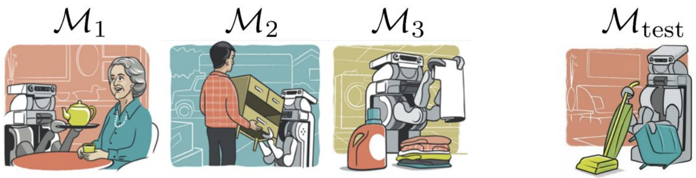
Reinforcement learning :

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = f_{RL}(\mathcal{M})$$

Meta reinforcement learning :

$$\theta^* = \arg\max_\theta \sum \mathbb{E}_{\tau\sim\pi_{\phi_i}}[R(\tau)] \textit{ where } \phi_i = f_\theta(\mathcal{M}_i)$$

$$\mathcal{D}_{meta-train} = \{\mathcal{M}_0, \mathcal{M}_1, \ldots\}$$



Graph: **ICML 2019 tutorial on meta learning**

Meta learning RL procedure

1. [*initialization*] *given a distribution over MDPs* $p(\mathcal{M})$, *draw* $\mathcal{M}_i \sim p(\mathcal{M})$
2. [*task adaptation*] *get our policy* π_{ϕ_i} *by the meta learner* $f_\theta(\mathcal{M}_i)$
3. [*data collection*] *explore or exploit* $\mathcal{M}_i$ *with* π_{ϕ_i} *and collect experiences*
4. [*meta learning*] *maximize the meta parameter* θ *with* collected data
5. *repeat*

Core problem

$$\theta^* = \arg\max_\theta \sum \mathbb{E}_{\tau\sim\pi_{\phi_i}}[R(\tau)] \text{ where } \phi_i = f_\theta(\mathcal{M}_i)$$

How do we design $f_\theta(\mathcal{M}_i)$? What does $f_\theta(\mathcal{M}_i)$ do?

1. f_θ *improves the policy with experiences from* $\mathcal{M}_i$
2. f_θ *can also choose how to interact with* $\mathcal{M}_i$ *(exploration vs exploitation)*

Popular approaches to meta-rl

- Memory-based approach (black-box adaptation)
  - Recurrent policy (RNN, LSTM)
  - Attention + temporal convolution
  - Mean field assumption

- Optimization-based approach
  - MAML and its variants

- POMDP perspective
  - Task inferences and embedding

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

Memory-based approach (black-box adaptation)

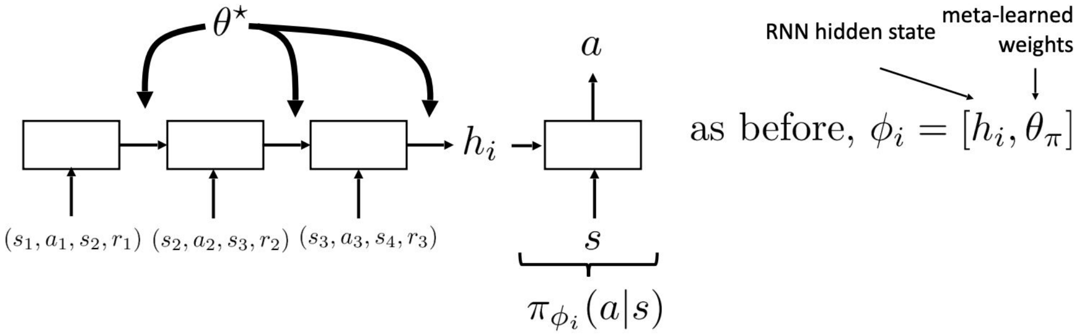
- Key idea: in order to learn a **"good"** prior, we need to somehow 1) **"memorize"** experiences we've seen so far, 2) and to “**adapt”** quickly to new tasks with our memory.

- “Good” prior:
  - Internalize the dynamics about the MDP; interactions with previous tasks help future tasks
- “Memorization”:
  - Recurrent networks, temporal convolutions + attentions
- “Adapt”:
  - Few shot experiences from the test MDP lead to a decent policy

Memory-based approach (black-box adaptation)

- Key idea: in order to learn a **"good"** prior, we need to somehow 1) **"memorize"** experiences we've seen so far, 2) and to "**adapt"** quickly to new tasks with our memory.

- Recipe:
  - Augmented "observation space": include **past experience (states, actions, rewards)**
  - A policy that takes into account all its past trajectory in a MDP by using this augmented observation (RNN policy for example)

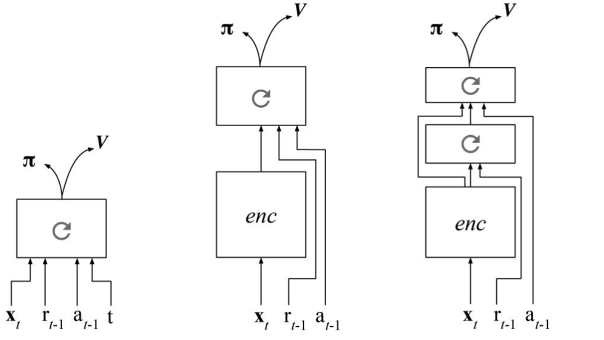


Graph: **ICML 2019 tutorial on meta learning**

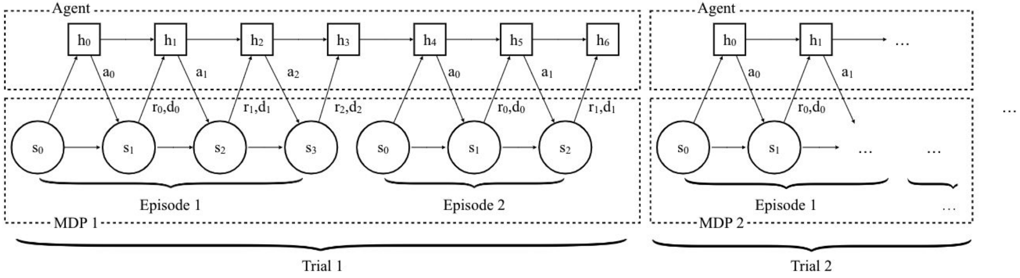
Memory-based approach (black-box adaptation)

- Key idea: in order to learn a **"good"** prior, we need to somehow 1) **"memorize"** experiences we've seen so far, 2) and to “**adapt”** quickly to new tasks with our memory.

- Procedures:
  - Sample a new MDP
  - Reset the hidden state
  - Collect trajectories and update the model by **maximizing total return** (using RL algorithms)



(a) LSTM A2C (b) LSTM A3C (c) Stacked-LSTM A3C



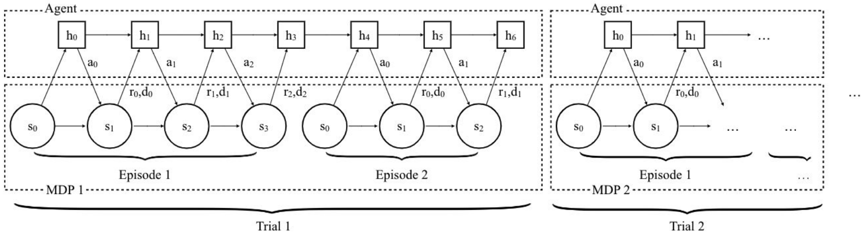
Wang et al. **Learning to Reinforcement Learn** [2016]

Duan et al. **RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning** [2016]

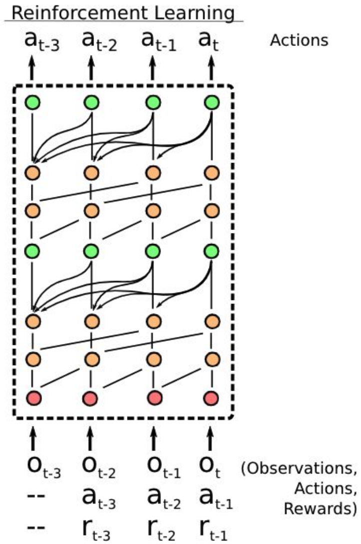
Memory-based approach (black-box adaptation)

- Key idea: in order to learn a **"good"** prior, we need to somehow 1) **"memorize"** experiences we've seen so far, 2) and to “**adapt”** quickly to new tasks with our memory.

- How to design architectures for the memory?
    - RNN, LSTM, GRU
    - Attention + temporal convolution



Duan et al. **RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning** [2016]



Mishra, Rohaninejad et al. **A Simple Neural Attentive Meta-Learner** [2018]

Memory-based approach (black-box adaptation)

- Problems?
  - **[Learnability]** Memory (gradient vanishing/explosion during BPTT, etc)

  - **[Data efficiency]** Works mostly in conjunction with on-policy RL algorithms

  - **[Optimality]** Trade-offs between exploration and exploitation

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

Optimization-based approach

- Most of the works in this category is based on ideas from MAML.
- Learn a proper **initialization** of the parameters so that after **few-shot experiences** from the new MDP, the policy nicely **adapts** to the new task.
- The learned meta parameter lies in the parameter space where it's close to the optimal task specific parameters on average.
- The meta parameters and the task-specific parameters coincide.

Optimization-based approach

- A quick recap of MAML (meta-rl as an optimization problem)

Meta reinforcement learning goal :

$$\theta^* = \arg\max_\theta \sum \mathbb{E}_{\tau\sim\pi_{\phi_i}}[R(\tau)] \text{ where } \phi_i = f_\theta^{generic}(\mathcal{M}_i)$$

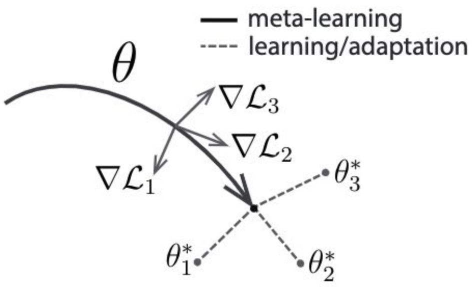
In MAML we have $\theta \equiv \phi$, *so the goal of MAML RL* :

$$\theta^* = \arg\max_\theta \sum \mathbb{E}_{\tau\sim\pi_{\theta_i}}[R(\tau)] \text{ where } \theta_i = f_\theta^{MAML}(\mathcal{M}_i)$$

Where the meta learner takes a specific form :
$f_\theta^{MAML}(\mathcal{M}_i) = \theta + \alpha\nabla_\theta J_i(\theta)$

When in the context of reinforcement learning :
$J_i(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$, *the expected sum of rewards in* $\mathcal{M}_i$

Which can be estimated by interacting with $\mathcal{M}_i$



Finn et al. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks** [2017]

Optimization-based approach

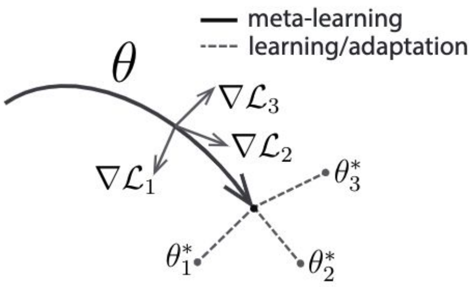
- Recipe:

In traditional RL, we optimize our parameter via :

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

In MAML, we optimize our parameter via :

$$\theta \leftarrow \theta + \beta \underbrace{\sum_i}_{\textit{over all tasks}} \nabla_\theta J_i\Big(\underbrace{\theta + \alpha \nabla_\theta J_i(\theta)}_{\textit{per task adaptation}}\Big)$$



Finn et al. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks** [2017]

- Interpretations:
  - Run one iteration of ascent and update our parameter based on how much such one step optimization can help with the task.
  - We want to optimize the parameter so that when we later do one step gradient ascent (task adaptation) on the test task, the objective is maximized in expectation (over the task distribution)

Optimization-based approach

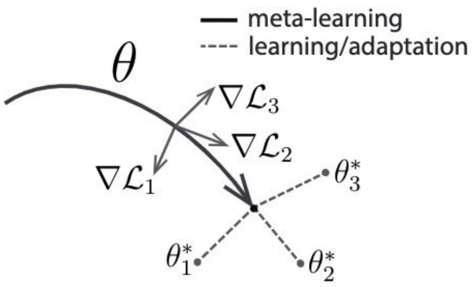
- Recipe:

In traditional RL, we optimize our parameter via :

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

In MAML, we optimize our parameter via :

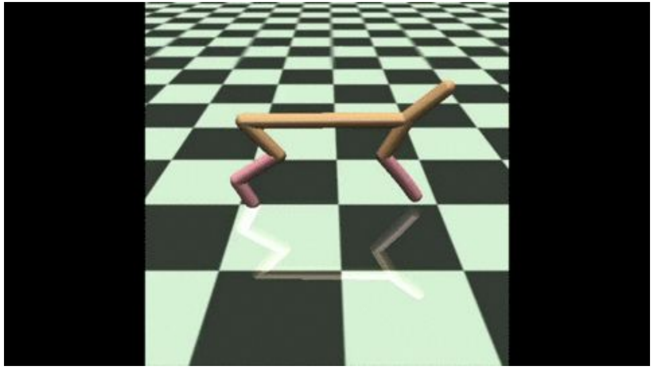
$$\theta \leftarrow \theta + \beta \underbrace{\sum_i}_{\textit{over all tasks}} \nabla_\theta J_i\Big(\underbrace{\theta + \alpha \nabla_\theta J_i(\theta)}_{\textit{per task adaptation}}\Big)$$

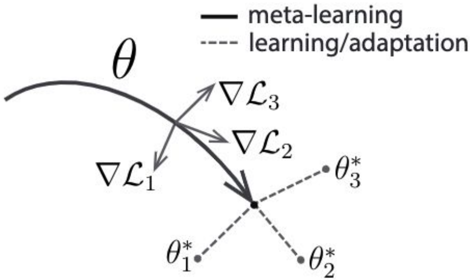


Finn et al. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks** [2017]

- Procedure:
  - Pick a random task i
  - Make one (or more) gradient step(s) to find its adapted parameter $\theta + \alpha \nabla_\theta J_i(\theta)$
  - Optimize the objective based on how good this adapted parameter performs $\nabla_\theta J_i(\theta + \alpha \nabla_\theta J_i(\theta))$
  - So the final parameter results in policy that performs well on average

Optimization-based approach



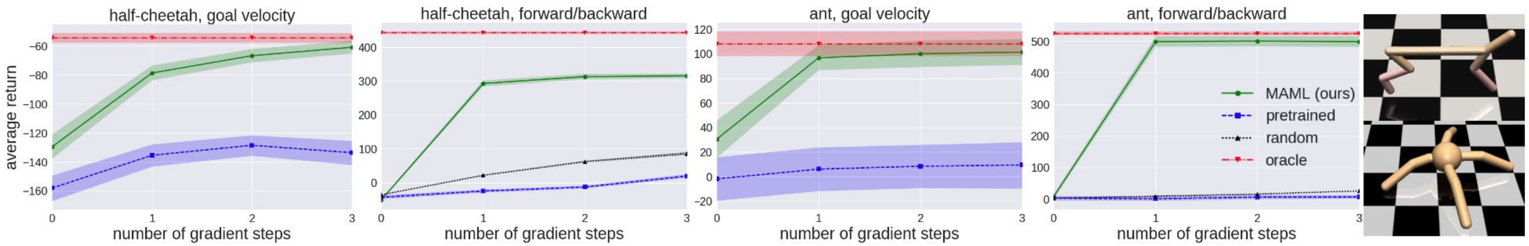


- Procedure:
  - Pick a random task i
  - Make one (or more) gradient step(s) to find its adapted parameter $\theta + \alpha\nabla_\theta J_i(\theta)$
  - Optimize the objective based on how good this adapted parameter performs $\nabla_\theta J_i(\theta + \alpha\nabla_\theta J_i(\theta))$
  - So the final parameter results in policy that performs well on average

Finn et al. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks** [2017]
gif. **tristandeleu/pytorch-maml-rl**

Optimization-based approach

- One (or few) shot learning with new sampled goals in robotic controls
  - Major drawbacks
    - Requires Hessian calculation. Tricks for approximation or acceleration?
    - What if the optimal parameters are not in the vicinity of each other in the parameter space? Do we have guarantees on the generalization and adaptation power?



Finn et al. **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks** [2017]

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

POMDP perspective

- How is meta-rl fundamentally different from generic reinforcement learning?
- Or is it different?
- In fact, meta-rl can be seen as a regular reinforcement learning **except that the state has to be partially observable.**

POMDP perspective

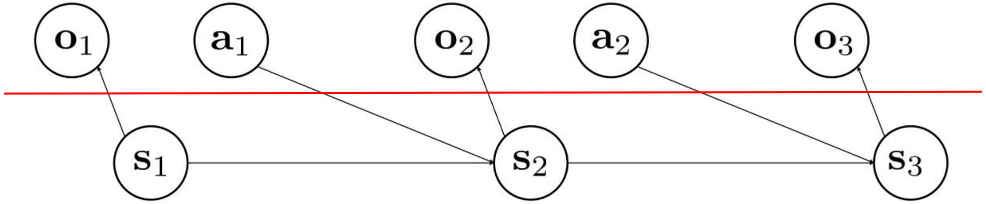
- A quick recap of partially observed Markov decision process

Augment a regular MDP with **observation space** *and* **emission probability**

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

$\mathcal{O}$: *observation space*

$\mathcal{E}$: *emission probability, i.e.* $p(o_t|s_t)$



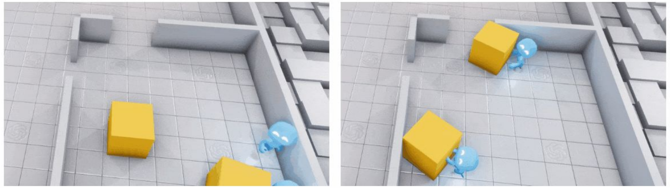
Graph: **ICML 2019 tutorial on meta learning**

POMDP perspective

- Under POMDPs, policy can only **act on observations** instead of the underlying states.
- POMDPs are known to be **extremely difficult to solve** as it requires reasoning about true states.
- Typically, to solve POMDPs:
  - **State estimations**: model the distribution of states given the observations (history), and apply usual RL procedures to find the optimal policy.
  - **Use policies with memory**: implicitly infer the internal dynamics of the MDP based on previous experiences.

POMDP perspective

- Meta-rl in the lens of regular rl in POMDPs:
- Key idea:
  - 1. encapsulate **task-specific information with a latent variable** on which the policy depends
  - 2. Learning involves **inferring the task context variable** and **optimizing the policy**



The agents can **grab and move** objects in front of them.

The agents can **lock** objects in place. Only the team that locked an object can unlock it.

Regular RL:

1. $MDP : \mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\}$

2. $policy : \pi_\theta(a|s)$

Meta RL in the lens of POMDPs:

1. $POMDP : \widetilde{\mathcal{M}} = \left\{\widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{\mathcal{P}}, r, \mathcal{E}, \mathcal{O}\right\}$ *where*

$\widetilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}$, *the concatenation of the state space and the task context*

$\widetilde{\mathcal{P}} = p\left(\tilde{s}_{t+1}|\tilde{s}_t, a_t\right)$, *the new transition function on the new state space*

$\mathcal{O} = \mathcal{S}$

$\mathcal{E} = p\left(o_t|\tilde{s}_t\right)$

2. $policy : \pi_\theta\left(a|s, \underbrace{z}_{task\ context}\right)$

gif: DeepMind. **Emergent Tool Use from Multi-agent Auto Curricula** [2020]

POMDP perspective

Meta RL in the lens of POMDPs :

1\. $POMDP : \widetilde{\mathcal{M}} = \left\{ \widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{\mathcal{P}}, r, \mathcal{E}, \mathcal{O} \right\}$ *where*

$\widetilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z},$ *the concatenation of the state space and the task context*

$\widetilde{\mathcal{P}} = p\left(\tilde{s}_{t+1} | \tilde{s}_t, a_t\right),$ *the new transition function on the new state space*

$\mathcal{O} = \mathcal{S}$

$\mathcal{E} = p\left(o_t | \tilde{s}_t\right)$

2\. $policy : \pi_\theta\left(a | s, \underbrace{z}_{\text{task context}}\right)$

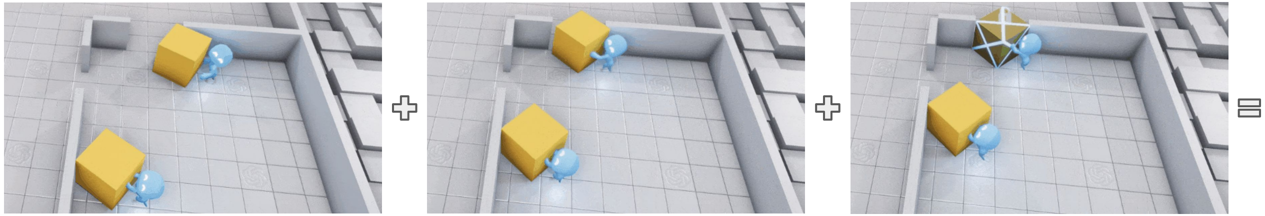
- Remember, typically, to solve POMDPs:
  - **State estimations**: model the distribution of states given the observations (history), and apply usual RL procedures to find the optimal policy
  - **Use policies with memory**: implicitly infer the internal dynamics of the MDP based on previous experiences.

POMDP perspective

$$POMDP : \widetilde{\mathcal{M}} = \left\{\widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{\mathcal{P}}, r, \mathcal{E}, \mathcal{O}\right\}, policy : \pi_\theta\left(a|s, \underbrace{z}_{task\ context}\right)$$

The goal is to estimate the posterior probability of the task context variable given experiences

$$p\left(\underbrace{z_t}_{task\ context} \mid \underbrace{s_{1:t}, a_{1:t}, r_{1:t}}_{experiences\ from\ tha\ task}\right)$$



Use locked objects to block the room

POMDP perspective

$$POMDP : \widetilde{\mathcal{M}} = \left\{ \widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{\mathcal{P}}, r, \mathcal{E}, \mathcal{O} \right\}, policy : \pi_\theta\Big(a|s, \underbrace{z}_{task\ context}\Big)$$

The goal is to estimate the posterior probability of the task context variable given experiences

$$p\Big(\underbrace{z_t}_{task\ context} \mid \underbrace{s_{1:t}, a_{1:t}, r_{1:t}}_{experiences\ from\ tha\ task} \Big)$$

posterior sampling with latent context :

1. *sample the latent variable with our current model*
$z \sim \tilde{p}(z_t | s_{1:t}, a_{1:t}, r_{1:t})$

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

- Comments on posterior sampling:
  - Often uses **variational inference** to approximate the posterior
  - Enables **exploration**
  - **Not optimal**
  - **Works well in practice**

POMDP perspective

Goal : optimize both the policy $\pi_\theta(a_t|s_t, z_t)$ *and the posterior context variable* $q_\lambda(z_t|\tau_{1:t})$
Optimization :

$$(\theta, \lambda) = \arg\max_{\theta,\lambda} \underbrace{\frac{1}{N}\Sigma_{i=1}^{n}}_{\text{average over tasks}} \mathbb{E}_{q_\lambda,\pi_\theta}\Bigg[\overbrace{\underbrace{R_i(\tau)}_{\substack{\text{trajectory return}\\ \text{under the}\\ \text{context dependent policy}}} - \underbrace{D_{KL}(q_\lambda(z)||p(z))}_{\substack{\text{regularizing the}\\ \text{variational}\\ \text{distribution}}}}^{ELBO}\Bigg]$$

- Comments:
    - We can think of the return as the likelihood as in VI, which means we want to find the task context variable that makes high trajectory rewards more likely.

- This is actually an important design choice.

Rakelly, Zhou et al. **Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables** [2019]

POMDP perspective

- How do we optimize the policy?
- How do we parameterize the variational family?
- Can we choose other “likelihood” function?

Rakelly, Zhou et al. **Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables** [2019]

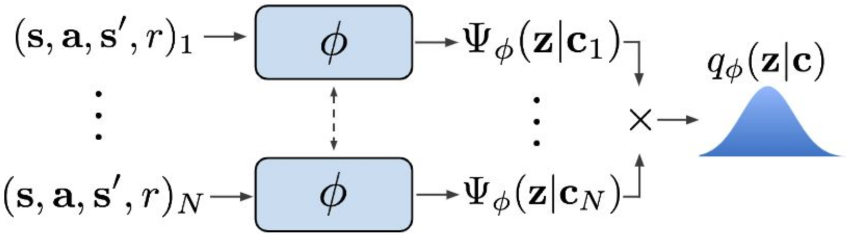
POMDP perspective

- How do we optimize the policy?
  - Using soft actor critic (**SAC**)

- How do we parameterize the variational family?
  - Mean-field assumption (**permutation invariance** of MDP encoding)
  - Accept **variable** length of history

- Can we choose other "likelihood" function?
  - Maximize the return (as mentioned before)
  - **Reconstruction of the MDP** (reward and dynamics modeling)
  - **Model state, or state-action value functions**

- PEARL = all above



Rakelly, Zhou et al. **Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables** [2019]

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

Model-free meta-rl perspectives summary

Recipes for three model free perspectives :

1. memory based :

$$f_\theta(\mathcal{M}_i) := RNN(\tau_{1:t})$$
$$:= TemporalConvAttentive(\tau_{1:t})$$

2. bi level optimization :

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

3. POMDP and posterior inference :

task context aware policy $\pi_\theta(a|s,z)$

posterior $p(z_t|\tau_{1:t})$

- Relationships:
    - 3 is the **stochastic** version of 1 where the task context variable z is the adaptation parameter.
    - 2 is the same as 1 and 3 conceptually except that it chooses **a specific form** of the meta learner than a black-box function approximator.

Model-free meta-rl perspectives summary

Recipes for three model free perspectives :

1. memory based :

$$f_\theta(\mathcal{M}_i) := RNN(\tau_{1:t})$$
$$:= TemporalConvAttentive(\tau_{1:t})$$

2. bi level optimization :

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

3. POMDP and posterior inference :

task context aware policy $\pi_\theta(a|s, z)$

posterior $p(z_t|\tau_{1:t})$

1. memory based :

simple to understand and implement

vulnerable to meta overfitting

optimization challenges

2. bi level optimization :

consistency

poor sample efficiency

3. POMDP and posterior inference :

effective exploration

special perspective

same problems as memory based approaches

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions

How to design the meta training tasks?

- All the methods we talked about so far **take as granted a distribution of tasks (MDPs)**.

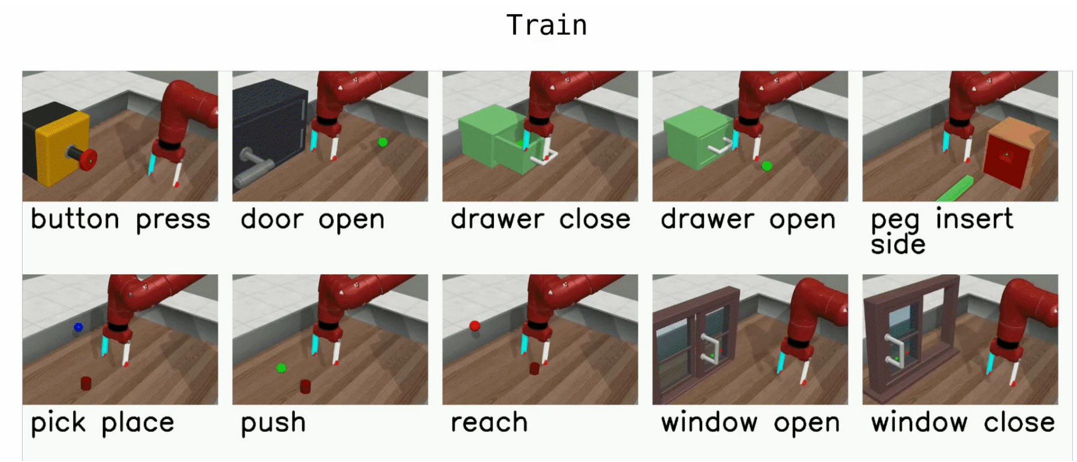
- In lots of scenarios, **the performance of the meta testing heavily depends on this distribution**:
    - Are the tasks structurally related?
    - Is the testing task in-distribution or similar to tasks from the meta training distribution?
    - Are the tasks rich enough to provide powerful prior?
    - How to systematically design such tasks for different problem?
    - …

- Successful applications of these methods often are coupled with **hand-crafted tasks**.

- Can we **automate** task designing while maintaining the power of meta RL?

Unsupervised Meta Reinforcement Learning

- Designing general task proposal algorithm can be infeasible.
- We restrict our attention to the setting where all tasks only differ in the reward function.
  - **In this case, the dynamics of the environment serves as the supervision for our task proposal algorithm**



Yu, Quillen, He, Julian, Narayan et al. **Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning** [2019]

Unsupervised Meta Reinforcement Learning

- In essence, the **optimal unsupervised meta RL learner** for a **Controlled Markov process** (MDP without reward functions) is the procedure producing the **policy which achieves the minimal worst case regret.** (Appendix for rigorous definition)
    - Worst case over all possible reward distribution (task distribution).
    - Minimal regret on expectation over the worst-case reward function distribution.
- Use **a latent variable** to control the reward function.
- Therefore, the most important design decision is the mapping from the latent variable to the reward function.

Gupta, Eysenbach et al. **Unsupervised Meta-Learning for Reinforcement Learning** [2019]

Unsupervised Meta Reinforcement Learning

- A practical implementation of the unsupervised reinforcement learning algorithm

Given a CMP

1. *obtain the reward proposal procedure*

2. *sample latent task variable* $z \sim p(z)$

3. *define task reward* r_z *using the reward proposal procedure and* z

4. *use standard meta learning algorithm with* r_z

- Reward proposal procedure can be defined in many ways
  - Randomly initialized
  - Or **optimized with some objective**
- Latent task variable can be simple distribution
- The proposal procedure takes in the value of the latent variable and produces a family of reward functions

Gupta, Eysenbach et al. **Unsupervised Meta-Learning for Reinforcement Learning** [2019]

Unsupervised Meta Reinforcement Learning

$DIAYN: \text{ a method for learning useful skills without a reward function}$

$DIAYN$ *does not depend on the rewards* and *only uses dyanmics as supervision.*

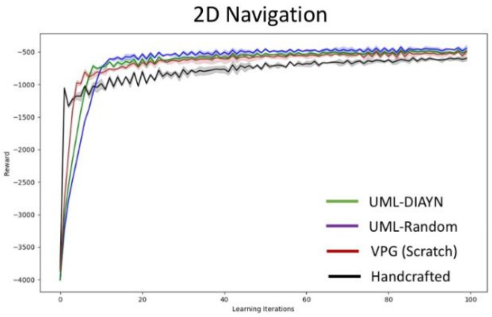
$Given\ D_\theta(z|s),\ define\ reward\ function\ r_z(s,a) := log(D_\theta(z|s))$

Use DIAYN to optimize the mutual information by training a discriminator $D_\theta(z|s)$ which predicts which latent variable was used to generate the rollout according to the policy $\pi(a|s,z)$.

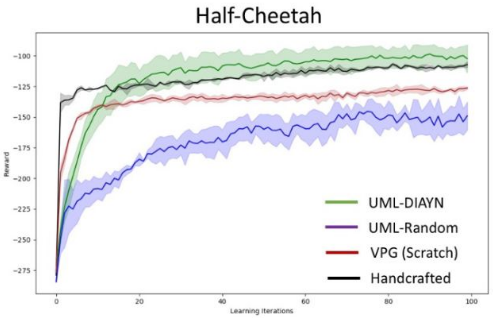
Gupta, Eysenbach et al. **Unsupervised Meta-Learning for Reinforcement Learning.** [2019]

Eysenbach et al. **Diversity is All You Need: Learning Skills without a Reward Function** [2018]

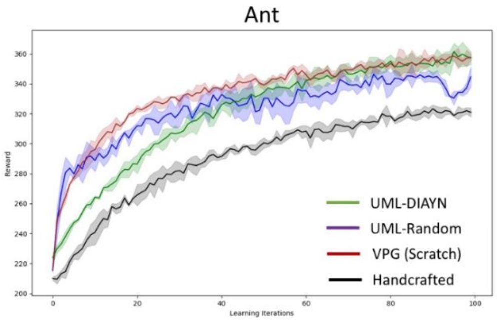
Unsupervised Meta Reinforcement Learning



2D Navigation



Half-Cheetah



Ant Navigation

Gupta, Eysenbach et al. **Unsupervised Meta-Learning for Reinforcement Learning** [2019]

Summary

- What we've covered:
  - Model free meta reinforcement learning
    - Black-box adaptation
    - Optimization based methods
    - Inference on POMDP
  - Unsupervised Task designs (kinda of)

- What we haven't covered:
  - Model based meta reinforcement learning
  - Hybrid methods
  - Enhanced exploration
  - Optimization beyond gradient descent (evolution strategies)
  - Heterogeneous architectures to handle different state and action spaces
  - …

References

- General
  - Finn, Sergey. **ICML 2019 tutorial on meta learning**

- Black-box adaptation
  - Wang et al. (2016) **Learning to Reinforcement Learn**
  - Duan et al. (2016) **RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning**
  - Mishra, Rohaninejad et al. (2018) **A Simple Neural Attentive Meta-Learner**

- Optimization-based methods
  - Finn et al. (2017) **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**

- POMDP perspective
  - Rakelly, Zhou et al. (2019) **Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables**

- Unsupervised meta learning
  - Gupta, Eysenbach et al. (2019) **Unsupervised Meta-Learning for Reinforcement Learning.**

Thank you!

- Questions are welcome

Appendix: Unsupervised Meta Reinforcement Learning

Controlled Markov process (CMP)
$C = \{\mathcal{S}, \mathcal{A}, \mathcal{P}\}$

A learning procedure (meta learner)
$f : \mathcal{D}(\mathcal{M}_i) \rightarrow \pi$

Evaluation of the meta learner for a specific reward function

$$R(f, r_z) = \sum_i \mathbb{E}_{\substack{\pi = f(\{\tau_1, \ldots, \tau_{i-1}\}) \\ \tau \sim \pi}} \left[\sum_t r_z(s_t, a_t) \right]$$

Task distribution
$\equiv$ *distribution over latent variable z*
$\equiv$ *distribution over reward functions*

Appendix: Unsupervised Meta Reinforcement Learning

The optimal learning procedure under a specific reward function distribution
$f^* := \arg\max_f \mathbb{E}_{p(r_z)}[R(f, r_z)]$

Regret of a learning procedure under a specific reward function distribution
$REGRET(f, p(r_z)) := \mathbb{E}_{p(r_z)}\left[R\left(f^*, r_z\right) - R(f, r_z)\right]$

Regret of a learning procedure for a CMP
$REGRET_{WC}(f, C) := \max_{p(r_z)} REGRET(f, p(r_z))$

Optimal unsupervised learning procedure
$f_C^* := \arg\min_f REGRET_{WC}(f, C)$