

Meta Reinforcement Learning

Learning to Explore

CS 330

Reminders

Homework 3, due **tonight**.

Project milestone due **Monday 11/2**.

Logistics

Optional Homework 4, out **tonight**.

Based on your feedback: Homework 4 will be in PyTorch
PyTorch tutorial session on Tuesday 6 pm PT.

Plan for Today

Recap: Meta-RL Basics

Learning to Explore

End-to-End Optimization of Exploration Strategies

Alternative Exploration Strategies

Decoupled Exploration & Exploitation

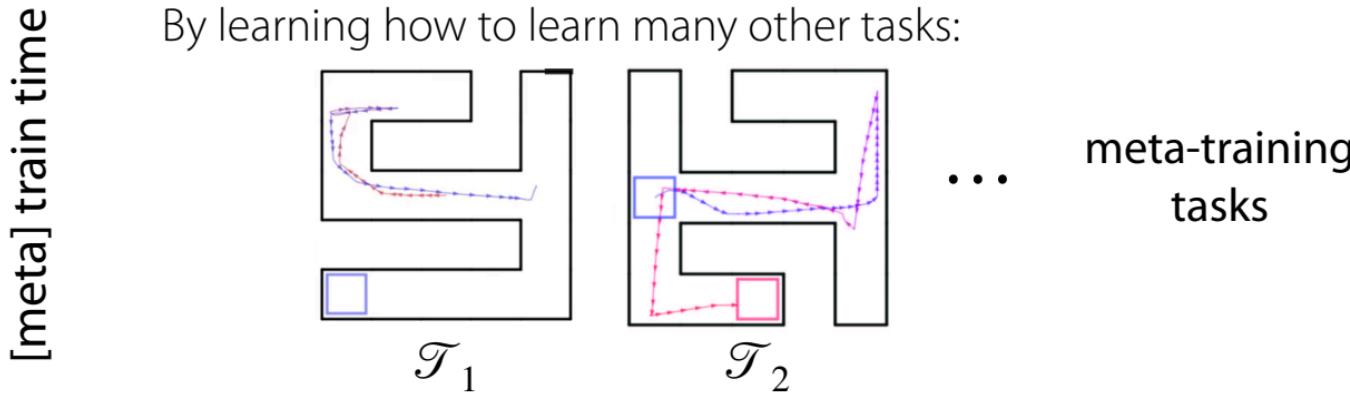
<- focus of HW4

Lecture goals:

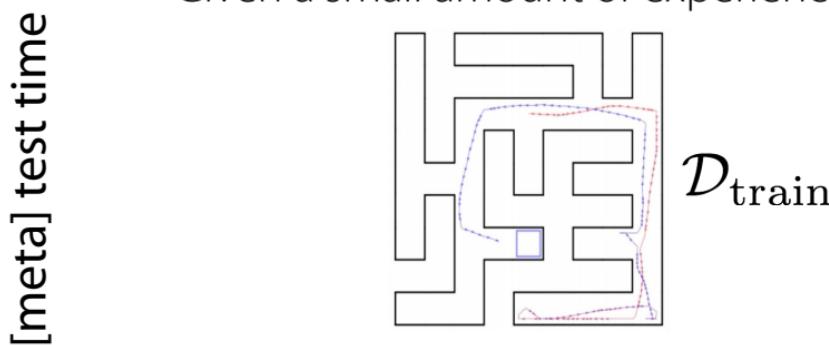
- Understand the challenge of end-to-end optimization of exploration
- Understand the basics of using alternative exploration strategies in meta-RL
- Understand & be able to implement decoupled exploration & exploitation

Recall: Meta-RL Maze Navigation Example

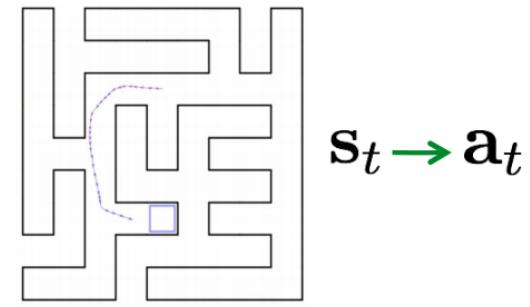
By learning how to learn many other tasks:



Given a small amount of experience

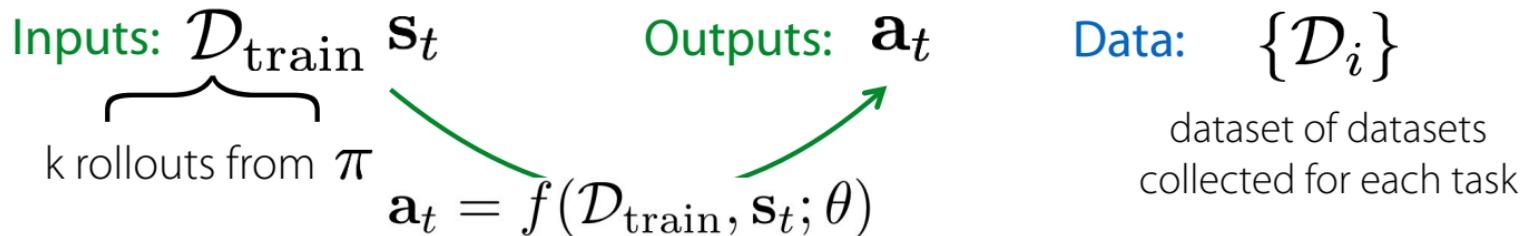


Learn to solve the task



Recall: The Meta Reinforcement Learning Problem

Meta Reinforcement Learning:



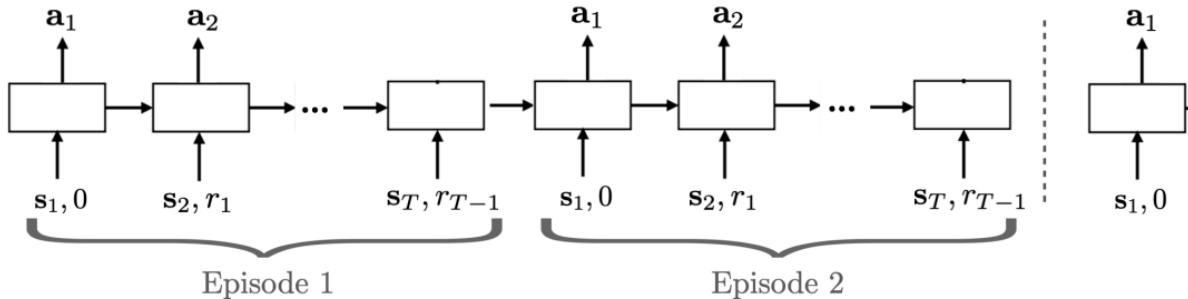
Design & optimization of f *and* collecting appropriate data
(learning to explore)

This lecture: How to learn to collect $\mathcal{D}_{\text{train}}$

Recall: Black-Box Meta-RL

Black-box network
(LSTM, NTM, Conv, ...)

$$\mathbf{a}_t = f(\mathcal{D}_{\text{train}}, \mathbf{s}_t; \theta)$$



- + general & expressive
- + a variety of design choices in architecture
- hard to optimize
- ~ inherits sample efficiency from outer RL optimizer

Plan for Today

Recap: Meta-RL Basics

Learning to Explore

End-to-End Optimization of Exploration Strategies

Alternative Exploration Strategies

Decoupled Exploration & Exploitation

<- focus of HW4

Taking a step back...

Should we be using the same exploration algorithm for:

- Learning to navigate an environment
- Learning to make recommendations to users
- Learning a policy for computer system caching
- Learning to physically operate a new tool or machine

This is how we currently approach exploration.

Can we *learn exploration strategies* based on experience from other tasks in that domain?

How Do We Learn to Explore?

Solution #1: Optimize for Exploration & Exploitation *End-to-End* w.r.t. Task Reward

(Duan et al., 2016, Wang et al., 2016, Mishra et al., 2017, Stadie et al., 2018, Zintgraf et al., 2019, Kamienny et al., 2020)

- + simple
- + leads to optimal strategy
in principle
- challenging optimization
when exploration is hard

Example of a Hard Exploration Meta-RL Problem

Learned cooking tasks in previous kitchens



Goal: Quickly learn tasks in a new kitchen.



meta-training

meta-testing

Why is End-to-End Training Hard?

End-to-end approach: optimize exploration and execution episode behaviors end-to-end to maximize reward of execution



Coupling problem: learning exploration and execution depend on each other

—> can lead to poor local optima, poor sample efficiency

Plan for Today

Recap: Meta-RL Basics

Learning to Explore

End-to-End Optimization of Exploration Strategies

Alternative Exploration Strategies

Decoupled Exploration & Exploitation

<- focus of HW4

Solution #2: Leverage Alternative Exploration Strategies

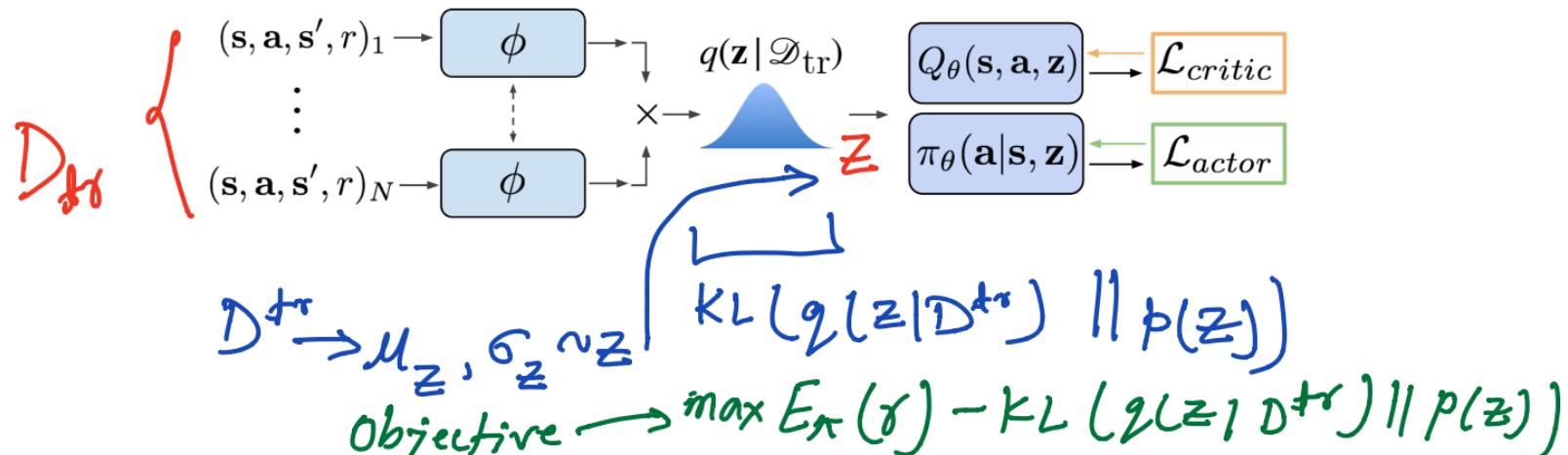
2a. Use posterior sampling
(also called Thompson sampling)

PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

i. Learn distribution over latent task variable \mathbf{z} and task-conditioned policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

Prior $p(\mathbf{z})$ - latent representation of the task distribution

Posterior $q(\mathbf{z} | \mathcal{D}_{\text{tr}})$ - updated belief about the task based on experience so far.



Solution #2: Leverage Alternative Exploration Strategies

2a. Use posterior sampling

(also called Thompson sampling)

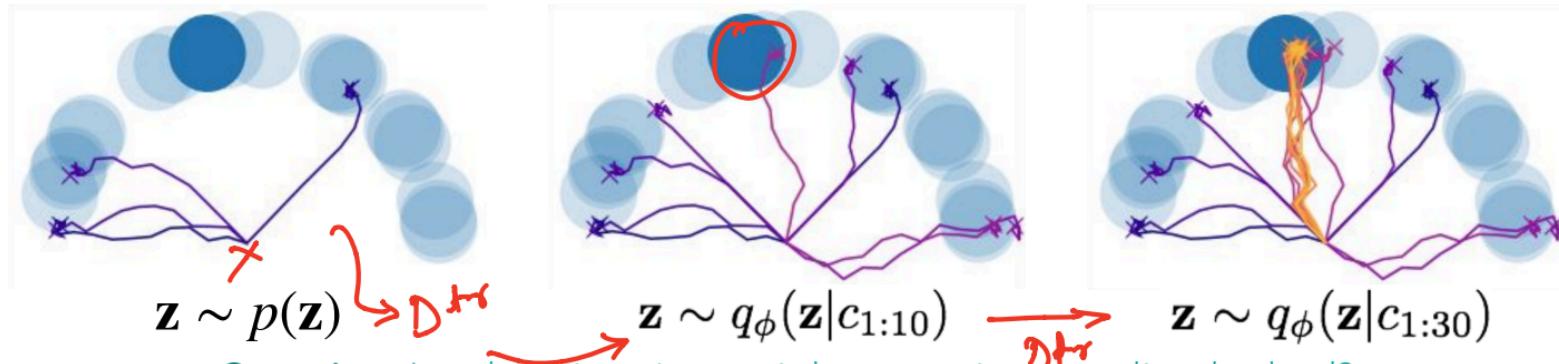
PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

i. Learn distribution over latent task variable \mathbf{z} and task-conditioned policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

Prior $p(\mathbf{z})$ - latent representation of the task distribution

Posterior $q(\mathbf{z} | \mathcal{D}_{\text{tr}})$ - updated belief about the task based on experience so far.

ii. Sample \mathbf{z} from current posterior $q(\mathbf{z} | \mathcal{D}_{\text{tr}})$, sample from policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$, add experience to \mathcal{D}_{tr}



Question: In what situations might posterior sampling be bad?

eg. Goals far away & sign on wall that tells you the correct goal.

Solution #2: Leverage Alternative Exploration Strategies

2a. Use posterior sampling

(also called Thompson sampling)

PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

i. Learn distribution over latent task variable \mathbf{z} and task-conditioned policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

ii. Sample \mathbf{z} from current posterior $q(\mathbf{z} | \mathcal{D}_{\text{tr}})$, sample from policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$, add experience to \mathcal{D}_{tr}

2b. Task dynamics & reward prediction

MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

Key idea: Train model $f(\mathbf{s}', r | \mathbf{s}, \mathbf{a}, \mathcal{D}_{\text{tr}})$ & collect \mathcal{D}_{tr} so that model is accurate.

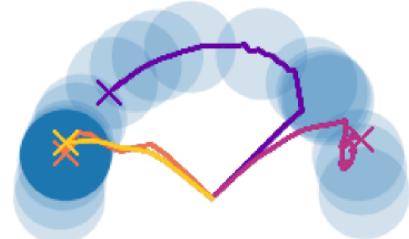
Iteratively, for each task:

1. Collect data \mathcal{D}_{tr} with exploration policy π_{exp}

2. Collect data \mathcal{D}_{ts} with policy $\pi_{\text{exe}}(\mathbf{a} | \mathbf{s}, \mathcal{D}_{\text{tr}})$

3. Train policy π_{exp} w.r.t. reward of $r_{\text{exp}} = -\|(\hat{\mathbf{s}}', \hat{r}) - (\mathbf{s}', r)\|^2$
where $\hat{\mathbf{s}}', \hat{r} \sim f(\cdot, \cdot | \mathbf{s}, \mathbf{a}, \mathcal{D}_{\text{tr}})$

4. Update policy π_{exe} w.r.t. task reward



Solution #2: Leverage Alternative Exploration Strategies

2a. Use posterior sampling

(also called Thompson sampling)

PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

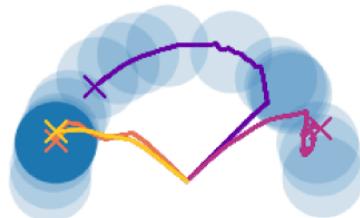
i. Learn distribution over latent task variable \mathbf{z} and task-conditioned policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

ii. Sample \mathbf{z} from current posterior $q(\mathbf{z} | \mathcal{D}_{\text{tr}})$, sample from policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$, add experience to \mathcal{D}_{tr}

2b. Task dynamics & reward prediction

MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

Key idea: Train model $f(\mathbf{s}', r | \mathbf{s}, \mathbf{a}, \mathcal{D}_{\text{tr}})$ & collect \mathcal{D}_{tr} so that model is accurate.



Question: In what situations might this be bad?

Lots of task-irrelevant distractors,
or complex, high-dim state dynamics

Solution #2: Leverage Alternative Exploration Strategies

2a. Use posterior sampling

(also called Thompson sampling)

PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

i. Learn distribution over latent task variable \mathbf{z} and task-conditioned policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

ii. Sample \mathbf{z} from current posterior $q(\mathbf{z} | \mathcal{D}_{\text{tr}})$, sample from policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$, add experience to \mathcal{D}_{tr}

2b. Task dynamics & reward prediction

MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

Key idea: Train model $f(\mathbf{s}', r | \mathbf{s}, \mathbf{a}, \mathcal{D}_{\text{tr}})$ & collect \mathcal{D}_{tr} so that model is accurate.

Overall

- + **easy** to optimize
- + many based on
principled strategies

- **suboptimal** by arbitrarily large amount in some environments.

Can we avoid the chicken-and-egg problem without sacrificing optimality?

Plan for Today

Recap: Meta-RL Basics

Learning to Explore

End-to-End Optimization of Exploration Strategies

Alternative Exploration Strategies

Decoupled Exploration & Exploitation

<- focus of HW4

Solution #3

Idea from solution #2b: Train model $f(\mathbf{s}', r | \mathbf{s}, \mathbf{a}, \mathcal{D}_{\text{tr}})$ & collect \mathcal{D}_{tr} so that model is accurate.

How might we explore **only** information **relevant for solving the task**?

Idea 3.0: Label each task/MDP with a unique ID $\mu \rightarrow \text{One-hot encoding}$
Train model $f(\mu | \mathbf{s}, \mathbf{a}, \mathcal{D}_{\text{tr}}) \quad \& \text{ collect } \mathcal{D}_{\text{tr}}$ so that model is accurate.

Question: What if two tasks are functionally identical but differ in task-irrelevant ways?

(e.g. identical kitchen layouts,
but different brand of garbage bags)

+ no longer need to model dynamics, rewards
— may still capture task-irrelevant details

→ Need a task representation that only distinguishes features needed for solving each task.

Solution #3: Decouple by acquiring representation of task relevant information

Meta-training

1) Learn execution & identify key information



MDP identifier μ
wall color [2]
ingredients [0]
decorations [1]

*Could be a
One-hot vector*

Bottlenecked representation



[0]

Execution policy

π^{exec}

2) Learn to explore by recovering that information



Exploration policy

π^{exp}

Exploration episode τ

Information recovery reward
 $MI(z; \tau)$

z

(z) Only keeps relevant representation

Meta-testing

π^{exp}

Exploration episode τ

π^{exec}

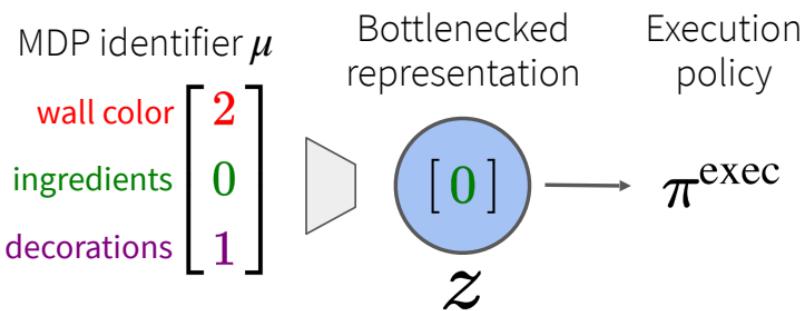
Decoupled Reward-free ExplorAtion and Execution in Meta-Reinforcement Learning (DREAM)



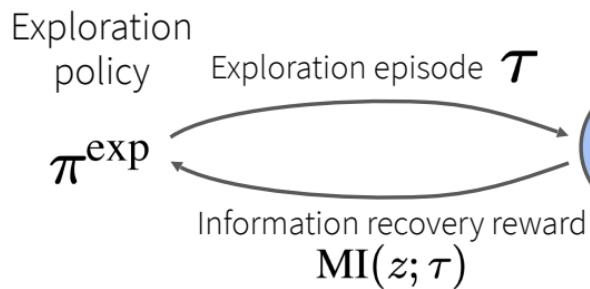
Solution #3: Decouple by acquiring representation of task relevant information

Meta-training

1) Learn execution & identify key information



2) Learn to explore by recovering that information



Train $\pi^{\text{exec}}(\mathbf{a} | \mathbf{s}, z_i)$ and encoder $F(z_i | \mu_i)$ to:

$$\max \sum_i \mathbb{E}_{\pi^{\text{exec}}} [r_i] - D_{\text{KL}} (F(z_i | \mu_i) \| \mathcal{N}(0, 1))$$

Train π^{exp} such that collected \mathcal{D}_{tr} is predictive of z_i .

In practice: (1) and (2) can be trained simultaneously.

Solution #3: Decouple by acquiring representation of task relevant information

Meta-training

1) Learn execution & identify key information



2) Learn to explore by recovering that information



Train $\pi^{\text{exec}}(\mathbf{a} | \mathbf{s}, z_i)$ and encoder $F(z_i | \mu_i)$ to:

$$\max_i \mathbb{E}_{\pi^{\text{exec}}} [r_i] - D_{\text{KL}}(F(z_i | \mu_i) \| \mathcal{N}(0, 1))$$

Train π^{exp} such that collected \mathcal{D}_{tr} is predictive of z_i .

How to formulate the *reward function* for π^{exp} ?

(a) Train model $q(z_i | \mathcal{D}_{\text{tr}})$

Prediction error at end of exploration episode?

--> too sparse 😞

Prediction error so far?

--> double counting 😞

(b) r_t = per-step information gain

Prediction error from $\tau_{1:t-1}$
– prediction error at $\tau_{1:t}$

$$\gamma_t = -\|q(z_i | \mathcal{I}_{1:t-1}) - z_i\|^2$$

$$\begin{aligned}\gamma_t &= \|q(z_i | \mathcal{I}_{1:t-1}) - z_i\|^2 \\ &\quad - \|q(z_i | \mathcal{I}_{1:t}) - z_i\|^2\end{aligned}$$

Solution #3: Decouple by acquiring representation of task relevant information

(Informal) Theoretical Analysis

(1) DREAM objective is **consistent** with end-to-end optimization.

[under mild assumptions]

-> can in principle recover the optimal exploration strategy

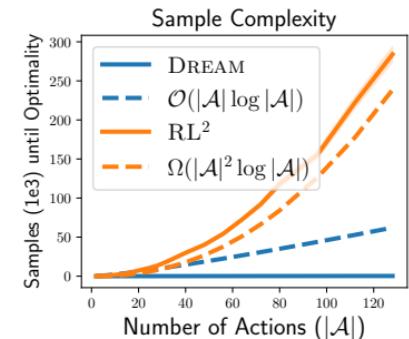
(2) Consider a bandit-like setting with $|\mathcal{A}|$ arms.

In MDP i , arm i yields reward. In all MDPs, arm 0 reveals the rewarding arm.

RL² requires $\Omega(|\mathcal{A}|^2 \log |\mathcal{A}|)$ samples for meta-optimization.

DREAM requires $\mathcal{O}(|\mathcal{A}| \log |\mathcal{A}|)$ samples for meta-optimization.

[assuming Q-learning with uniform outer-loop exploration]



How Do We Learn to Explore?

End-to-End

- + leads to optimal strategy in principle
- challenging optimization when exploration is hard

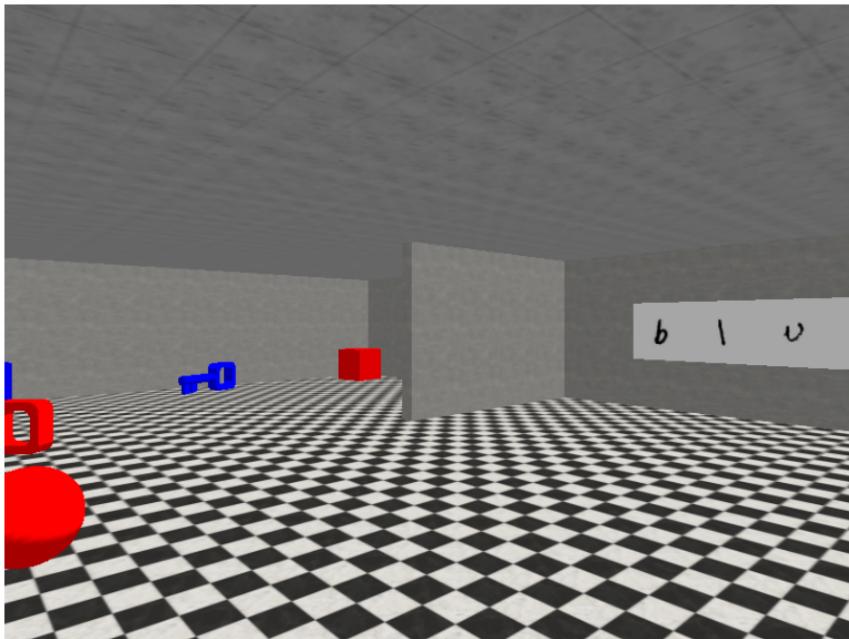
Alternative Strategies

- + easy to optimize
- + many based on principled strategies
- suboptimal by arbitrarily large amount in some environments.

Decoupled Exploration & Execution

- + leads to optimal strategy in principle
- + easy to optimize in practice
- requires task identifier

Empirical Comparison: Sparse Reward 3D Visual Navigation Problem

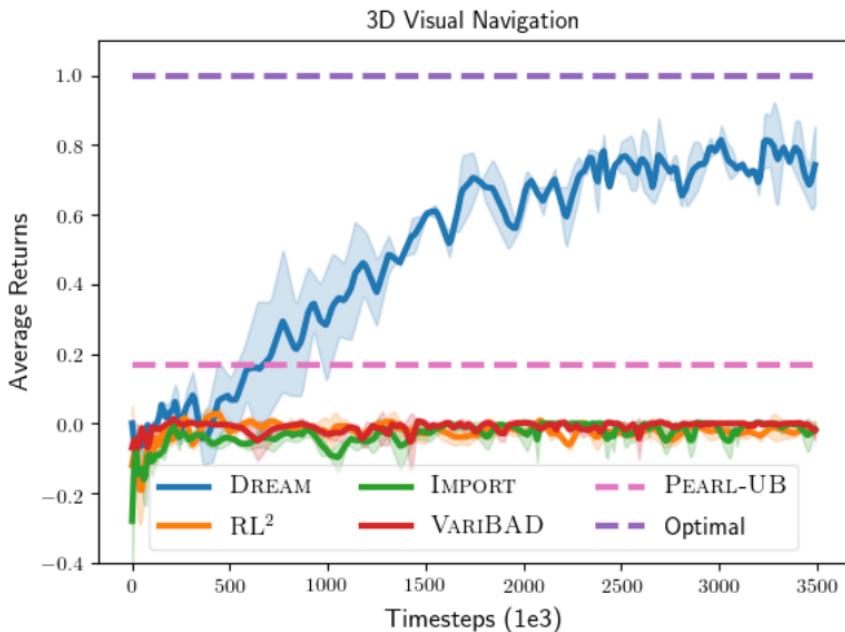


More challenging variant of task from Kamienny et al., 2020

- Task: go to the (key / block / ball), color specified by the sign
- Agent starts on other side of barrier, must walk around to read the sign
- Pixels observations (80 x 60 RGB)
- Sparse binary reward

$$\pi^{\text{exec}}(a|s, z_i) \quad q(z_i | D^{\text{tar}})$$
$$F(z_i | \mu_i) \quad \pi^{\text{exp}}(a|s) \rightarrow D^{\text{tar}}$$

Quantitative Comparison

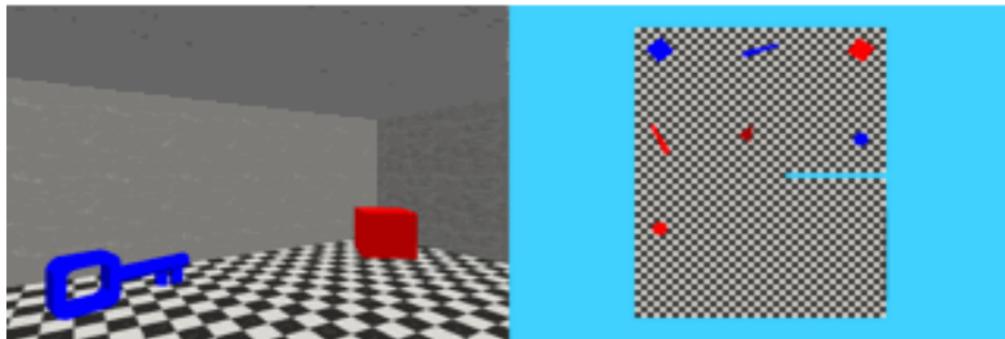


- End-to-end algorithms (RL^2 , IMPORT , VARIBAD) perform poorly due to **coupling**
- Alternate exploration strategies, e.g., posterior/Thompson sampling do not learn the optimal exploration strategy
- **PEARL-UB**: Upper-bound on PEARL reward achieved with optimal policy and Thompson-Sampling exploration
- **DREAM** achieves near-optimal reward

RL^2 (Duan et al., 2016), IMPORT (Kamienny et al., 2020), VARIBAD (Zintgraf et al., 2019), PEARL (Rakelly, et. al., 2019), Thompson, 1933

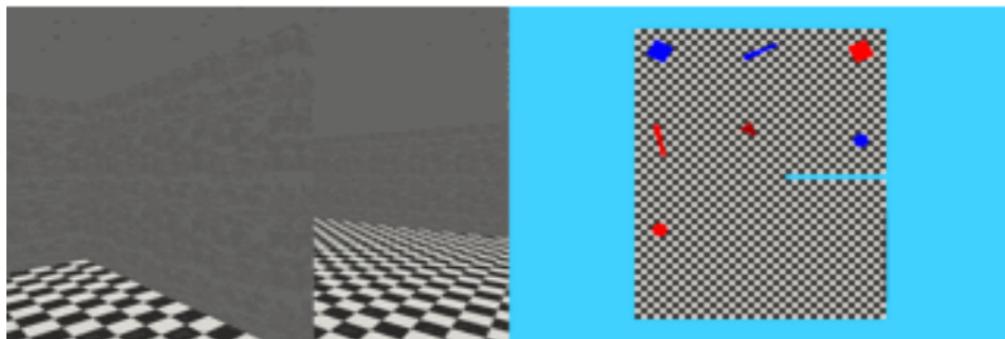
Qualitative Results for DREAM

Exploration episode



Execution episode

Goal: Go to key



Plan for Today

Recap: Meta-RL Basics

Learning to Explore

End-to-End Optimization of Exploration Strategies

Alternative Exploration Strategies

Decoupled Exploration & Exploitation

<- focus of HW4

Lecture goals:

- Understand the challenge of end-to-end optimization of exploration
- Understand the basics of using alternative exploration strategies in meta-RL
- Understand & be able to implement decoupled exploration & exploitation

Next time

Wednesday: Last lecture on meta-RL

Next week: Hierarchical RL & Lifelong Learning

Following week: Guest lecture by Jane Wang
Final frontiers lecture

Reminders

Homework 3, due **tonight**.

Optional Homework 4 out tonight

Project milestone due **Monday 11/2.**