Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables

International Conference on Machine Learning (ICML), 2019

Kate Rakelly*, Aurick Zhou*, Deirdre Quillen
Chelsea Finn, Sergey Levine

Dongmin Lee

Biointelligence Laboratory
Seoul National University

March, 2020

Outline

- Abstract
- Introduction
- Probabilistic Latent Context
- Off-Policy Meta-Reinforcement Learning
- Experiments

Abstract

Challenges of meta-reinforcement learning (meta-RL)

Abstract

Challenges of meta-reinforcement learning (meta-RL)

- Current methods rely heavily on on-policy experience, limiting their sample efficiency
- There is also lack of mechanisms to reason about task uncertainty when adapting to new tasks, limiting their effectiveness in sparse reward problems

Abstract

Challenges of meta-reinforcement learning (meta-RL)

- Current methods rely heavily on on-policy experience, limiting their sample efficiency
- There is also lack of mechanisms to reason about task uncertainty when adapting to new tasks, limiting their effectiveness in sparse reward problems

PEARL (Probabilistic Embeddings for Actor-critic RL)

- An off-policy meta-RL algorithm to achieve **both meta-training and adaptation efficiency**
- Perform probabilistic encoder filtering of latent task variables to enables posterior sampling for structured and efficient exploration

Abstract

Challenges of meta-reinforcement learning (meta-RL)

- Current methods rely heavily on on-policy experience, limiting their sample efficiency
- There is also lack of mechanisms to reason about task uncertainty when adapting to new tasks, limiting their effectiveness in sparse reward problems

PEARL (Probabilistic Embeddings for Actor-critic RL)

- An off-policy meta-RL algorithm to achieve **both meta-training and adaptation efficiency**
- Perform probabilistic encoder filtering of latent task variables to enables posterior sampling for structured and efficient exploration

Experiments

- Six continuous control meta-learning environments (MuJoCo simulator)
- 2-D navigation environment with sparse rewards

Introduction

Meta-learning problem statement

- The goal of meta-learning is to learn to adapt quickly to new task given a small amount of experience from previous tasks

Introduction

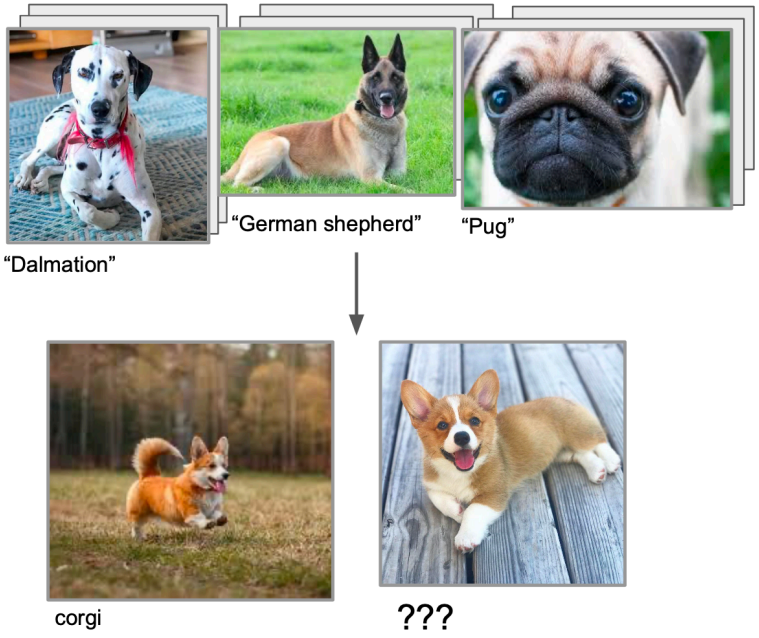
Meta-learning problem statement

adaptation efficiency

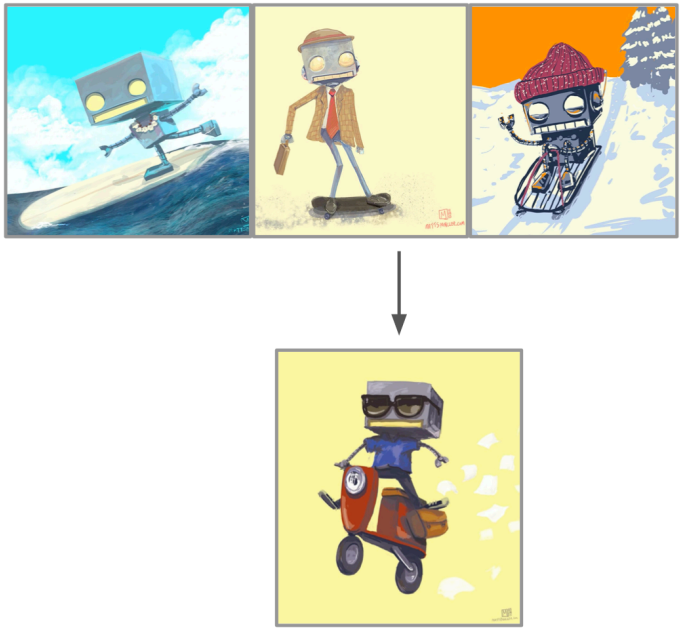
- The goal of meta-learning is to learn to adapt quickly to new task given a small amount of experience from previous tasks

meta-training efficiency

Supervised learning



Reinforcement learning



Introduction

Meta-RL problem statement

- The goal of meta-RL is to learn to adapt quickly to new task given a small amount of experience from previous tasks

Regular RL: learn policy for single task

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

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

MDP

Meta-RL: learn adaptation rule

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

meta-training / outer loop

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

adaptation / inner loop

MDP for task i

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$

$\mathcal{M}_{test}$

Introduction

General meta-RL algorithm outline

- While training:
  1. Sample task i, collect data $\mathcal{D}_i$;
  2. **Adapt** policy by computing $\phi_i = f(\theta, \mathcal{D}_i)$;
  3. Collect data $\mathcal{D}_i'$ with adapted policy π_{ϕ_i};
  4. **Update** θ according to $\mathcal{L}(\mathcal{D}_i', \phi_i)$;

Introduction

General meta-RL algorithm outline

- While training:
  1. Sample task i, collect data $\mathcal{D}_i$;
  2. **Adapt** policy by computing $\phi_i = f(\theta, \mathcal{D}_i)$;
  3. Collect data $\mathcal{D}_i'$ with adapted policy π_{ϕ_i};
  4. **Update** θ according to $\mathcal{L}(\mathcal{D}_i', \phi_i)$;

→ Different algorithms:

- Choice of function f
- Choice of loss function $\mathcal{L}$

Introduction

Solution I: recurrent policies

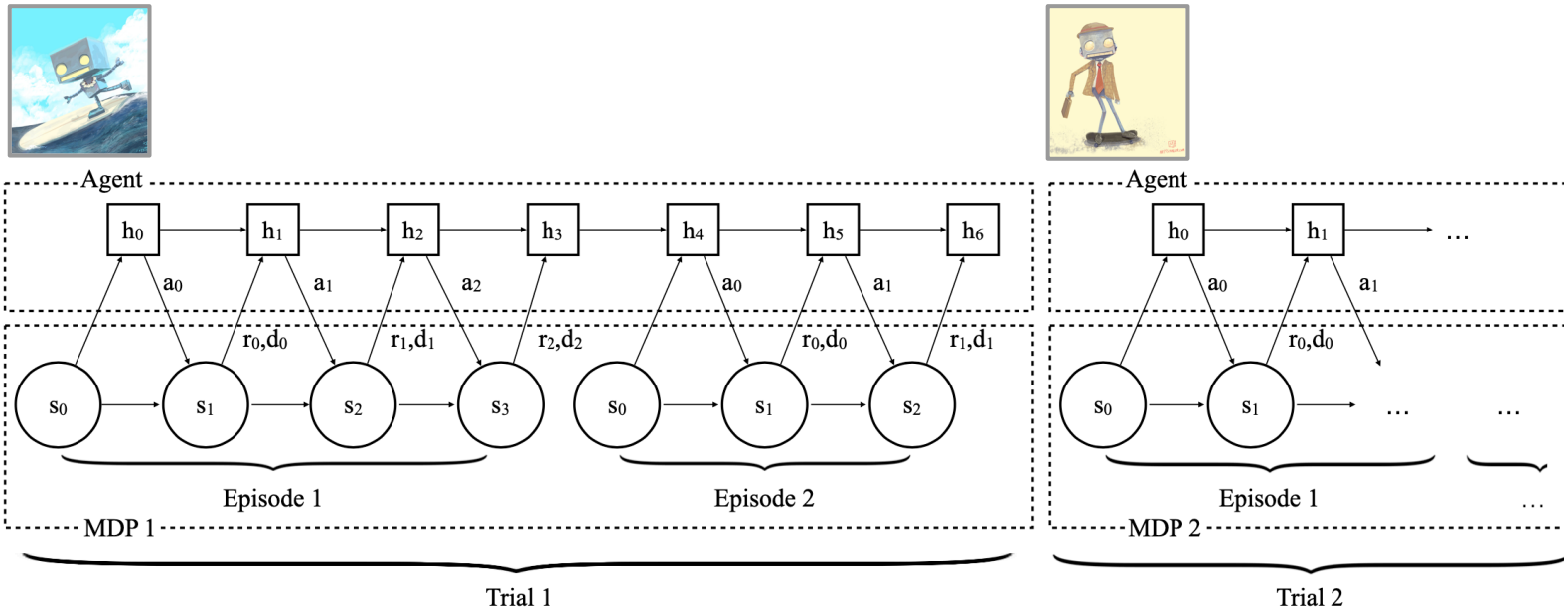
- Implement the policy as a recurrent network, train across a set of tasks

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

PG

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

RNN



Y. Duan et al., "RL^2 : Fast Reinforcement Learning via Slow Reinforcement Learning", ICLR 2017

Introduction

Solution II: optimization problem

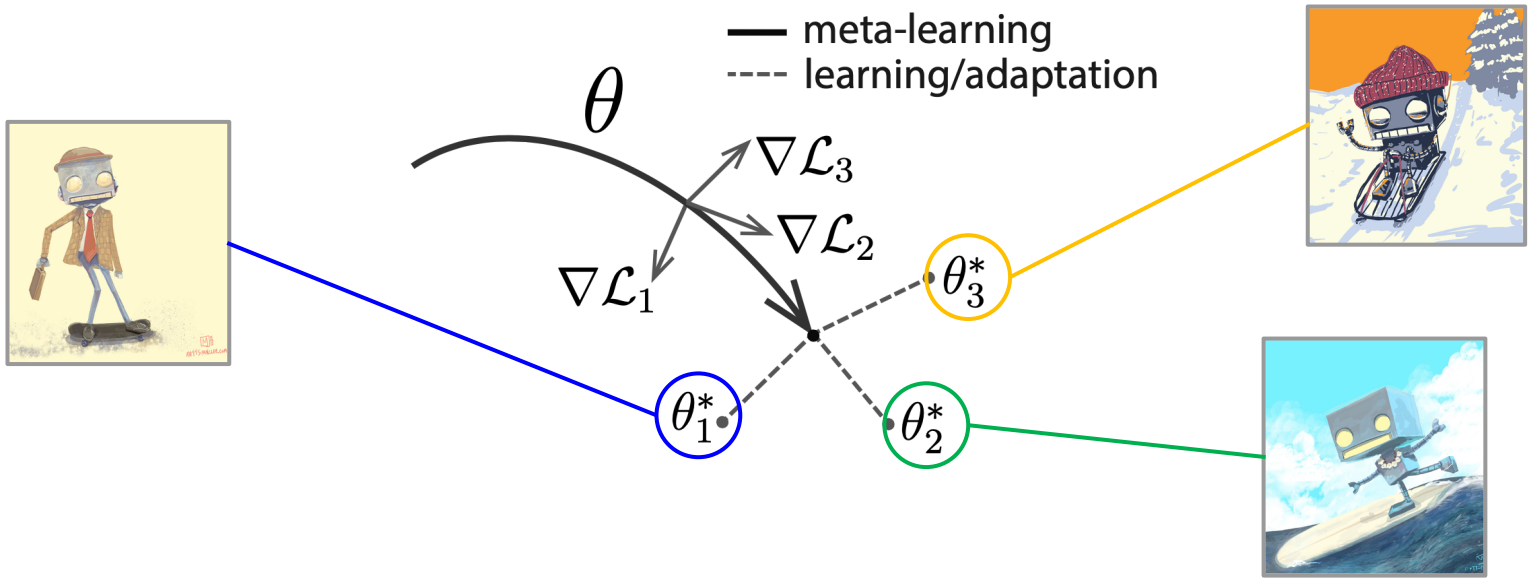
- Learn a parameter initialization from which fine-tuning for a new task works

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

PG

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

PG



C. Finn et al., "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", ICML 2017

Introduction

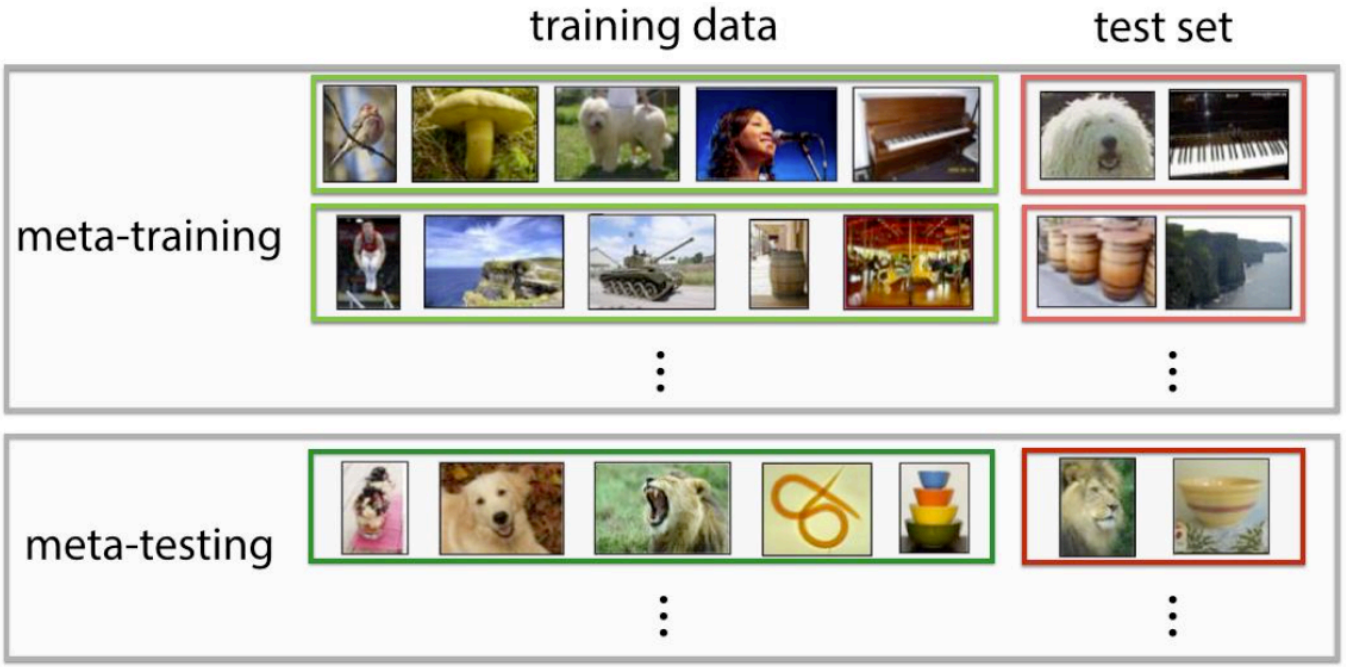
Challenge of meta-RL: sample efficiency

- Most current meta-RL methods require on-policy data during **both meta-training and adaptation**, which makes them exceedingly inefficient during meta-training

Introduction

Challenge of meta-RL: sample efficiency

- Most current meta-RL methods require on-policy data during **both meta-training and adaptation**, which makes them exceedingly inefficient during meta-training
- In image classification case, meta-learning typically operates on the principle that meta-training time should match meta-test time



Introduction

Challenge of meta-RL: sample efficiency

- Most current meta-RL methods require on-policy data during **both meta-training and adaptation**, which makes them exceedingly inefficient during meta-training
- In image classification case, meta-learning typically operates on the principle that meta-training time should match meta-test time
- In meta-RL, this makes it inherently difficult to meta-train a policy to adapt using off-policy data, which is systematically different from the data that the policy would see when it explores on-policy in a new task at meta-test time.

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

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

Train with off-policy data,
but then f_{θ^*} is on-policy...

Introduction

Challenge of meta-RL: sample efficiency

- Most current meta-RL methods require on-policy data during **both meta-training and adaptation**, which makes them exceedingly inefficient during meta-training
- In image classification case, meta-learning typically operates on the principle that meta-training time should match meta-test time
- In meta-RL, this makes it inherently difficult to meta-train a policy to adapt using off-policy data, which is systematically different from the data that the policy would see when it explores on-policy in a new task at meta-test time.

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

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

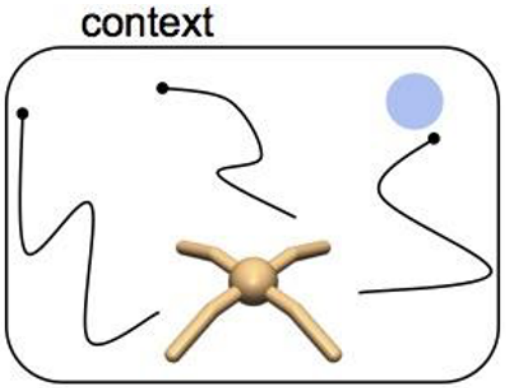
Train with off-policy data,
but then f_{θ^*} is on-policy...

→ **Can we meta-train as off-policy data and meta-test as on-policy data?**

Introduction

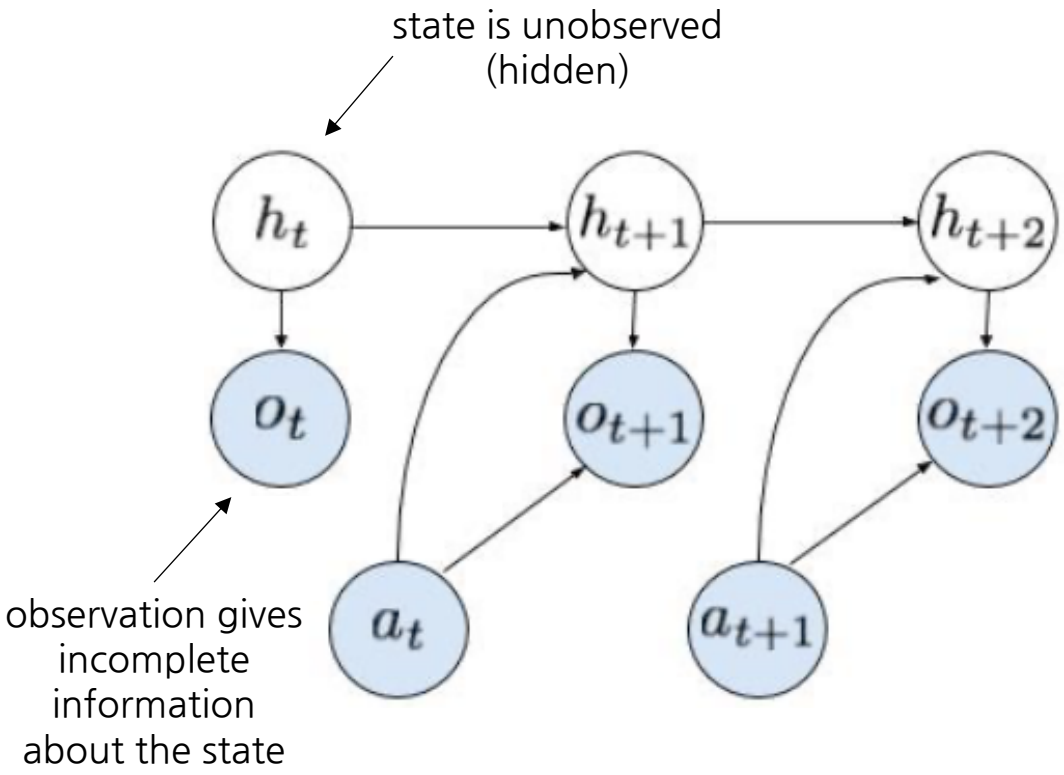
Definition

- $p(\mathcal{T})$: distribution of tasks
- S: set of states
- A: set of actions
- $p(s_0)$: initial state distribution
- $p(s_{t+1}|s_t, a_t)$: transition distribution
- $r(s_t, a_t)$: reward function
- $\mathcal{T} = \{p(s_0), p(s_{t+1}|s_t, a_t), r(s_t, a_t)\}$: a task
- $context\ c$: history of past transitions
- $c_n^{\mathcal{T}} = (s_n, a_n, r_n, s'_n)$: one transition in task $\mathcal{T}$



Introduction

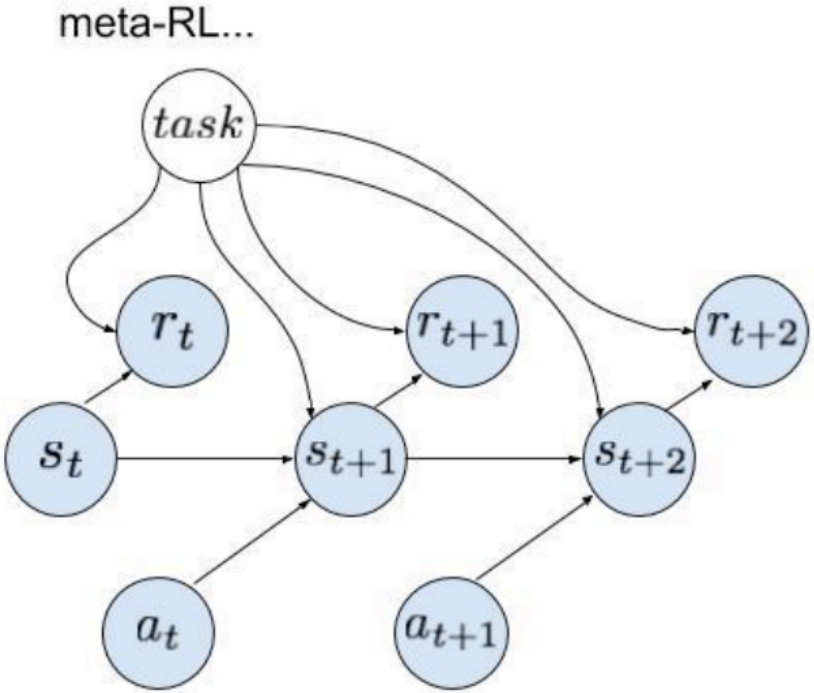
Aside: POMDPs (Partially-Observable Markov Decision Processes)

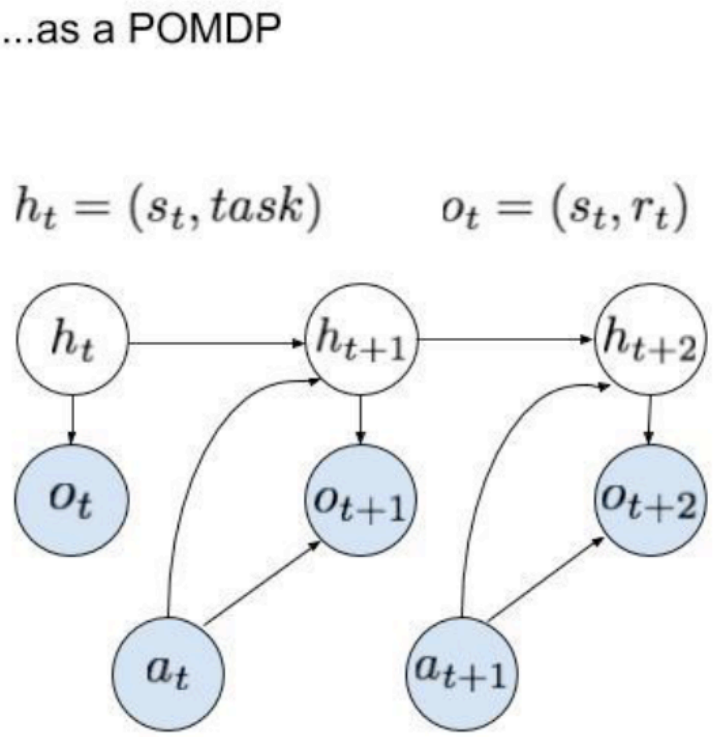


"That Way We Go" by Matt Spangler

Introduction

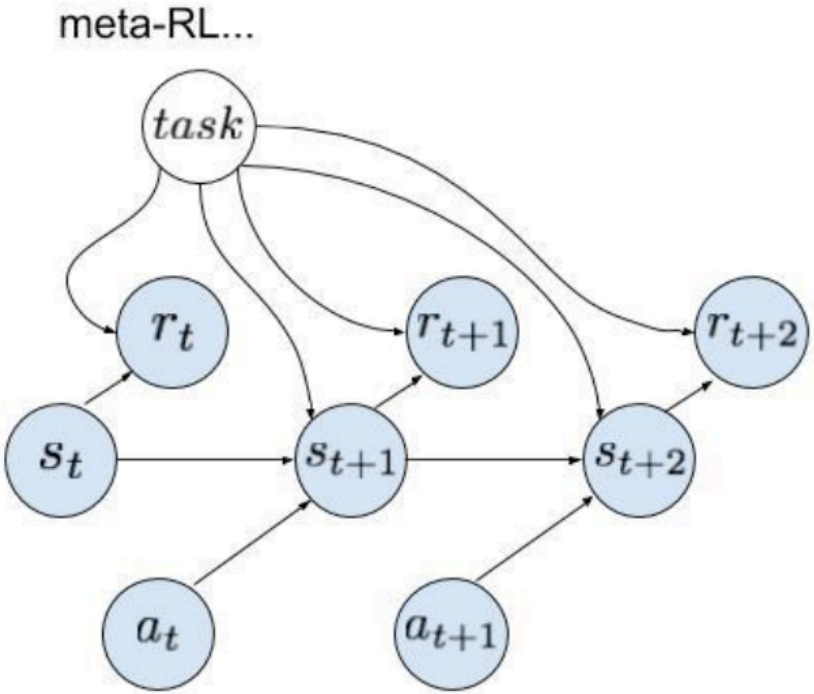
The POMDP view of meta-RL

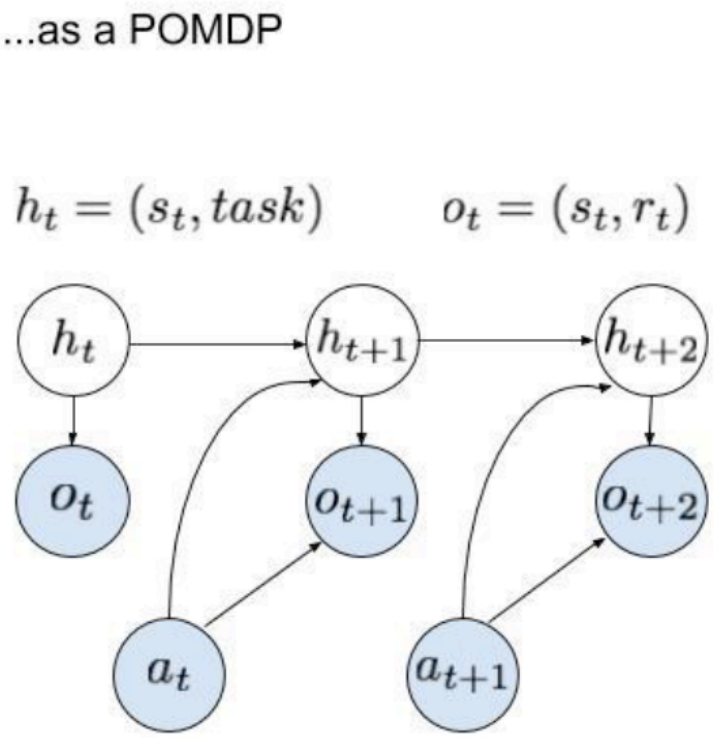




Introduction

The POMDP view of meta-RL

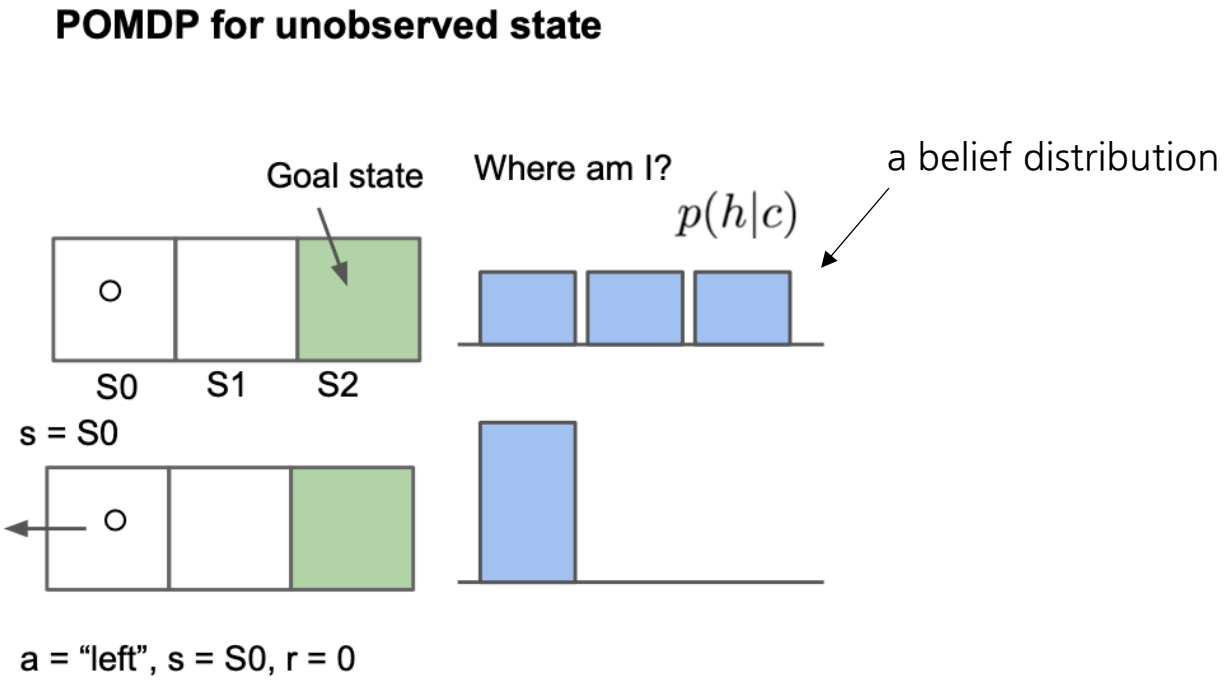




→ Can we leverage this connection to design a new meta-RL algorithm?

Introduction

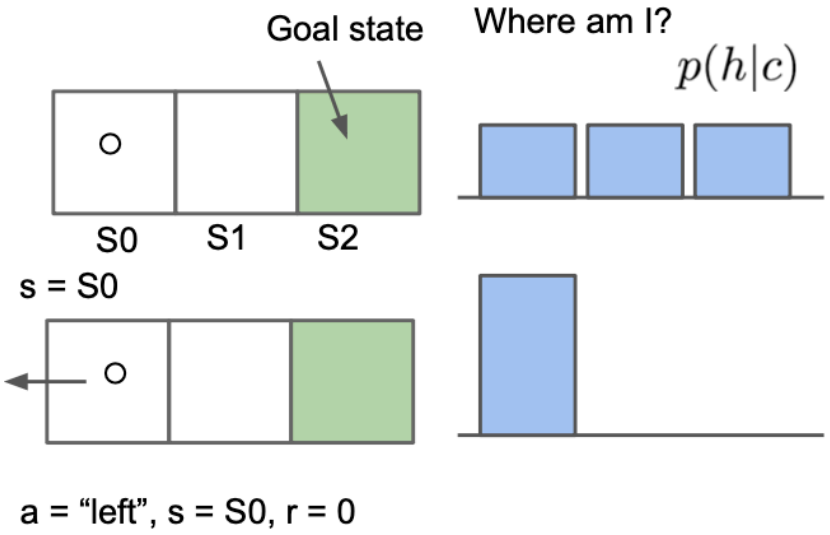
Model belief over latent task variables



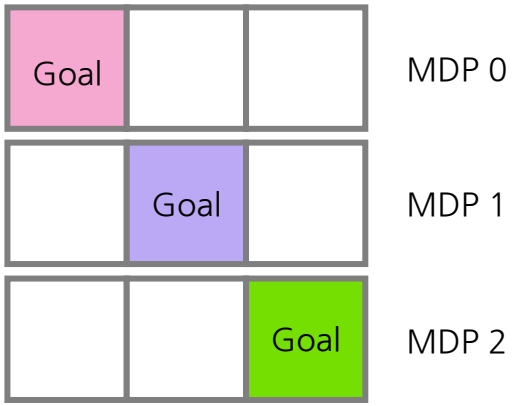
Introduction

Model belief over latent task variables

POMDP for unobserved state

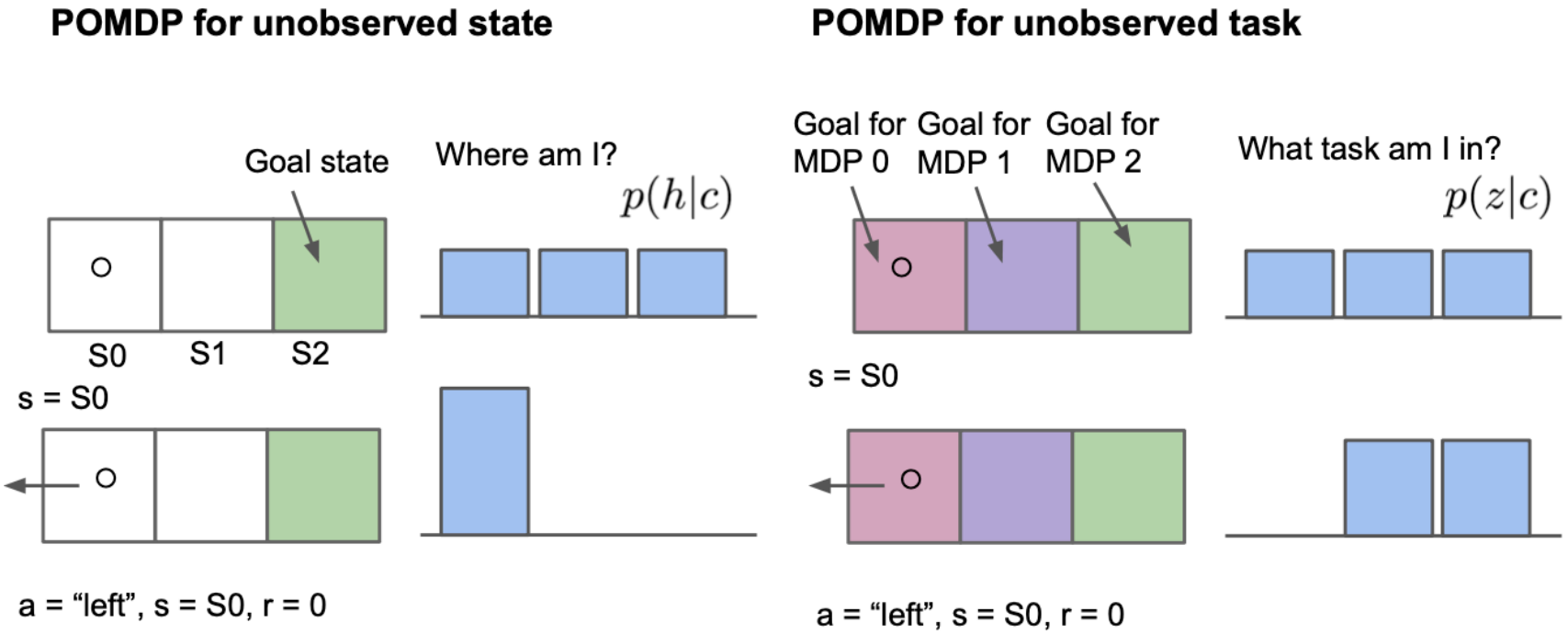


POMDP for unobserved task



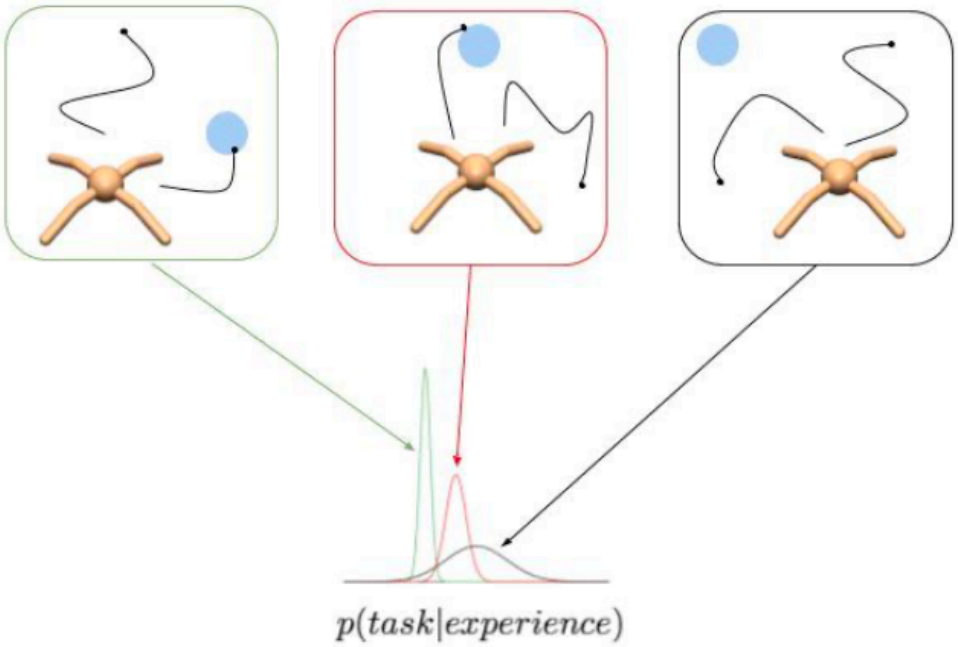
Introduction

Model belief over latent task variables



Introduction

RL with task-belief states



- "Task" can be supervised by **reconstructing the MDP**

Introduction

Can we achieve **both meta-training and adaptation efficiency**?

Introduction

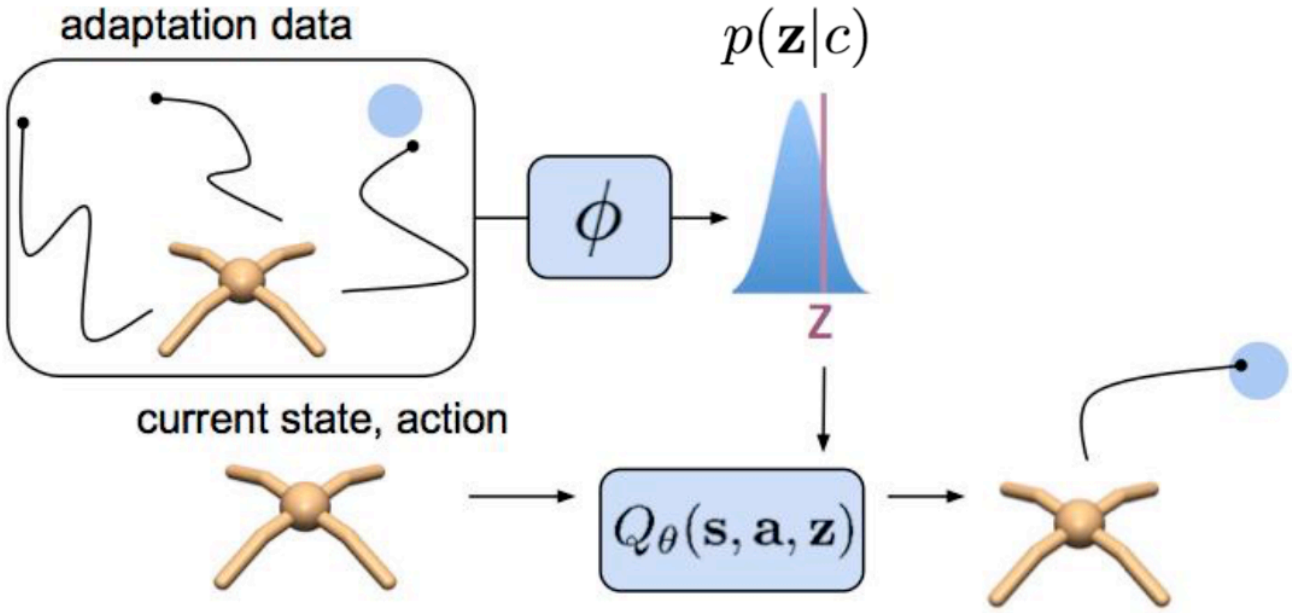
Can we achieve **both meta-training and adaptation efficiency**?

- Yes, let's make an off-policy meta-RL algorithm that disentangles task inference and control

Introduction

Can we achieve **both meta-training and adaptation efficiency**?

- Yes, let's make an off-policy meta-RL algorithm that disentangles task inference and control



Introduction

Can we achieve **both meta-training and adaptation efficiency**?

- Yes, let's make an off-policy meta-RL algorithm that disentangles task inference and control

Solution III: partially observed RL

- Meta-training efficiency: off-policy RL algorithm
- Adaptation efficiency: probabilistic encoder to estimate task-belief

Introduction

Can we achieve **both meta-training and adaptation efficiency**?

- Yes, let's make an off-policy meta-RL algorithm that disentangles task inference and control

Solution III: partially observed RL

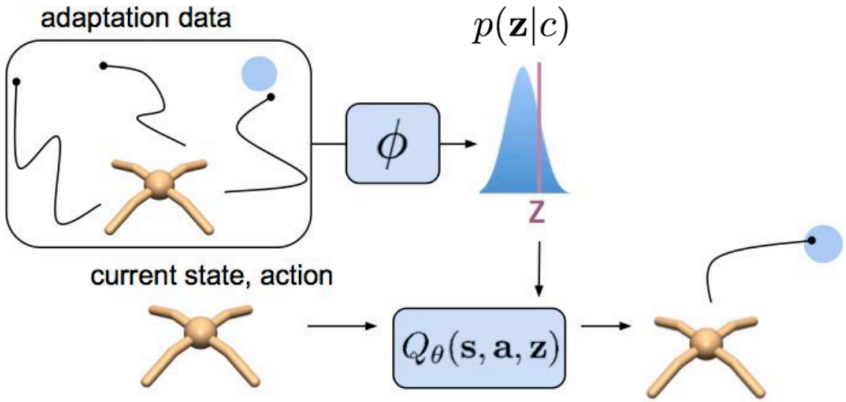
- Meta-training efficiency: off-policy RL algorithm
- Adaptation efficiency: probabilistic encoder to estimate task-belief

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

SAC

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

Probabilistic encoder



Probabilistic Latent Context

Modeling and learning latent contexts

- Adopt an amortized variational inference (VAE) approach to learn to infer a latent probabilistic context variable Z

  → **Inference network** $q_\phi(z|c)$ that estimates the true posterior $p(z|c)$

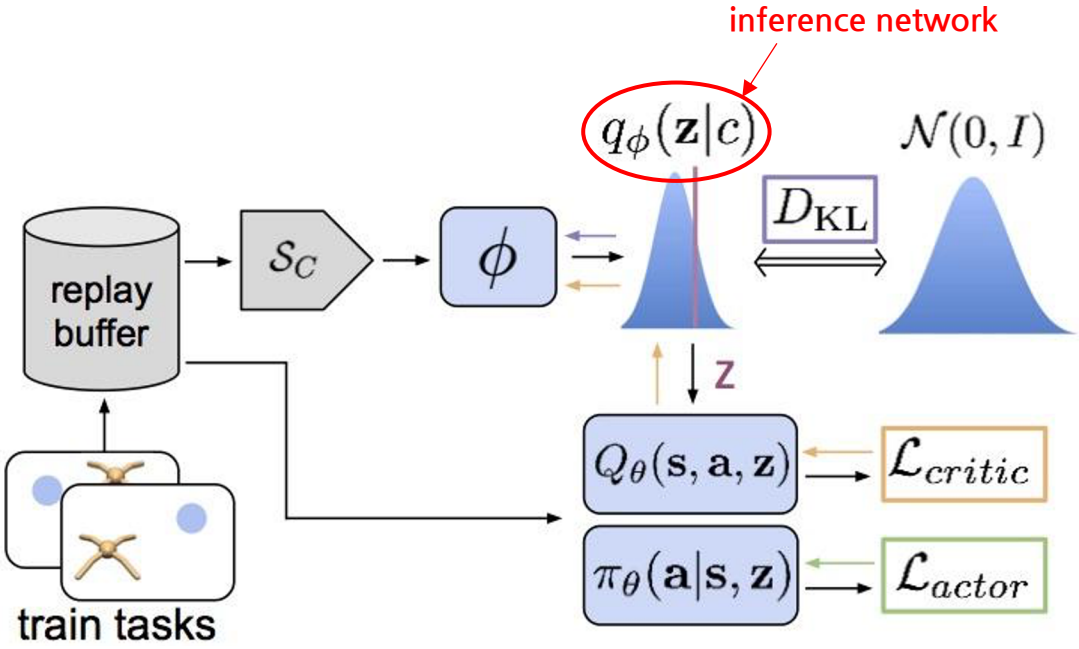
Probabilistic Latent Context

Modeling and learning latent contexts

- Adopt an amortized variational inference (VAE) approach to learn to infer a latent probabilistic context variable Z

  → **Inference network** $q_\phi(z|c)$ that estimates the true posterior $p(z|c)$

- Learn a predictive model of reward and dynamics by optimizing $q_\phi(z|c)$, and reconstruct the MDPs



Probabilistic Latent Context

Modeling and learning latent contexts

- Adopt an amortized variational inference (VAE) approach to learn to infer a latent probabilistic context variable Z

  → **Inference network** $q_\phi(z|c)$ that estimates the true posterior $p(z|c)$

- Learn a predictive model of reward and dynamics by optimizing $q_\phi(z|c)$, and reconstruct the MDP
- Assuming an objective to be a log-likelihood, the resulting variational lower bound is:

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{z\sim q_\phi(z|c^{\mathcal{T}})}\left[R(\mathcal{T},z)+\beta D_{\mathrm{KL}}\left(q_\phi(z|c^{\mathcal{T}})\,\|\,p(z)\right)\right]\right]$$

- $p(z)$ is a unit Gaussian prior over Z
- $\beta = 0.1$ is weight on KL divergence term

Probabilistic Latent Context

Modeling and learning latent contexts

- Adopt an amortized variational inference (VAE) approach to learn to infer a latent probabilistic context variable Z

  → **Inference network** $q_\phi(z|c)$ that estimates the true posterior $p(z|c)$

- Learn a predictive model of reward and dynamics by optimizing $q_\phi(z|c)$, and reconstruct the MDP
- Assuming an objective to be a log-likelihood, the resulting variational lower bound is:

"likelihood" term
(Bellman error)

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{z\sim q_\phi(z|c^{\mathcal{T}})}\left[R(\mathcal{T},z)+\beta D_{\mathrm{KL}}\left(q_\phi(z|c^{\mathcal{T}})\,\|\,p(z)\right)\right]\right]$$

- $p(z)$ is a unit Gaussian prior over Z
- $\beta = 0.1$ is weight on KL divergence term

Probabilistic Latent Context

Modeling and learning latent contexts

- Adopt an amortized variational inference (VAE) approach to learn to infer a latent probabilistic context variable Z

  → **Inference network** $q_\phi(z|c)$ that estimates the true posterior $p(z|c)$

- Learn a predictive model of reward and dynamics by optimizing $q_\phi(z|c)$, and reconstruct the MDP
- Assuming an objective to be a log-likelihood, the resulting variational lower bound is:

"likelihood" term (Bellman error)

variational approximation to posterior and prior

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{z\sim q_\phi(z|c^{\mathcal{T}})}\left[R(\mathcal{T},z)+\beta D_{\mathrm{KL}}\left(q_\phi(z|c^{\mathcal{T}})\,\|\,p(z)\right)\right]\right]$$

- $p(z)$ is a unit Gaussian prior over Z
- $\beta = 0.1$ is weight on KL divergence term

Probabilistic Latent Context

Modeling and learning latent contexts

- Adopt an amortized variational inference (VAE) approach to learn to infer a latent probabilistic context variable Z

  → **Inference network** $q_\phi(z|c)$ that estimates the true posterior $p(z|c)$

- Learn a predictive model of reward and dynamics by optimizing $q_\phi(z|c)$, and reconstruct the MDP
- Assuming an objective to be a log-likelihood, the resulting variational lower bound is:

"likelihood" term (Bellman error)

variational approximation to posterior and prior

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{z\sim q_\phi(z|c^{\mathcal{T}})}\left[R(\mathcal{T},z)+\beta D_{\mathrm{KL}}\left(q_\phi(z|c^{\mathcal{T}})\,\|\,p(z)\right)\right]\right]$$

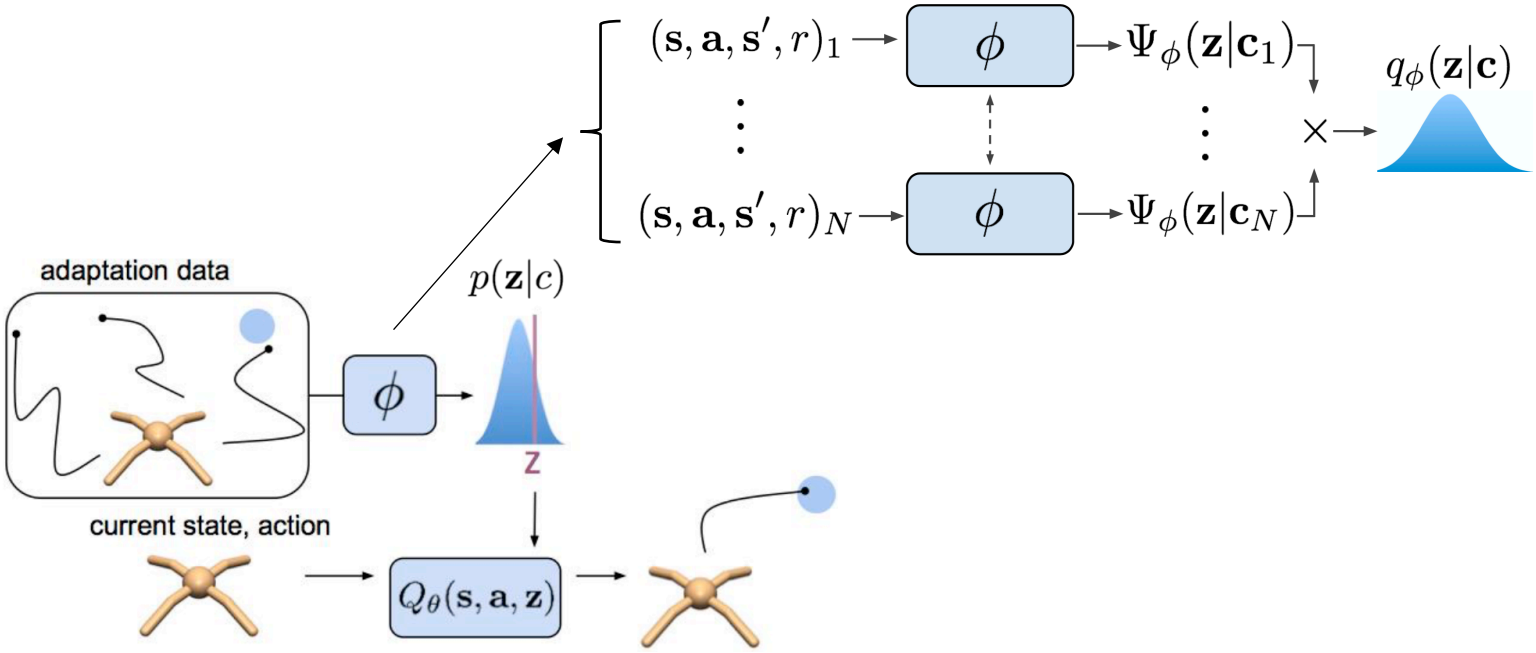
"regularization" term / information bottleneck

constrain the mutual information between Z and c

- $p(z)$ is a unit Gaussian prior over Z
- $\beta = 0.1$ is weight on KL divergence term

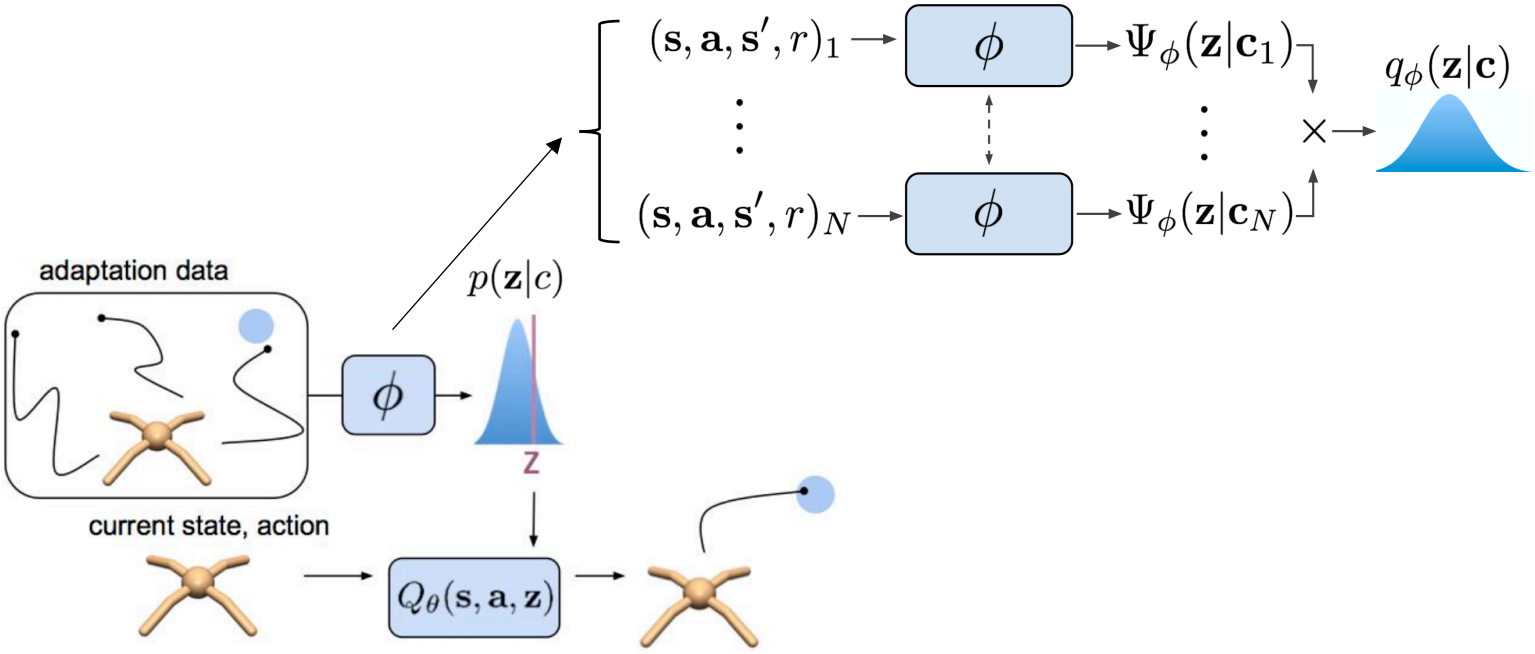
Probabilistic Latent Context

Designing architecture of inference network



Probabilistic Latent Context

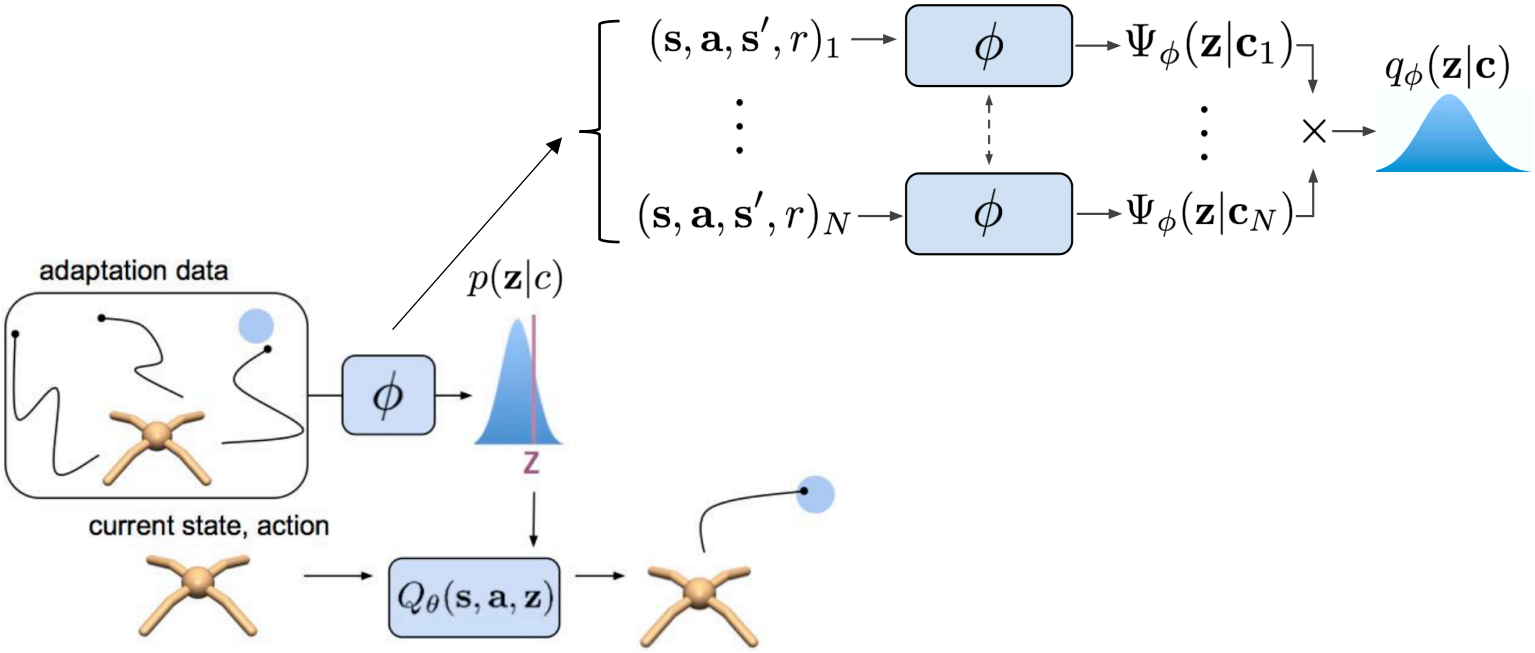
Designing architecture of inference network



- If we would like to infer what the task is or identify the MDP, it is enough to have access to a collection of transitions, without regard for the order of transitions

Probabilistic Latent Context

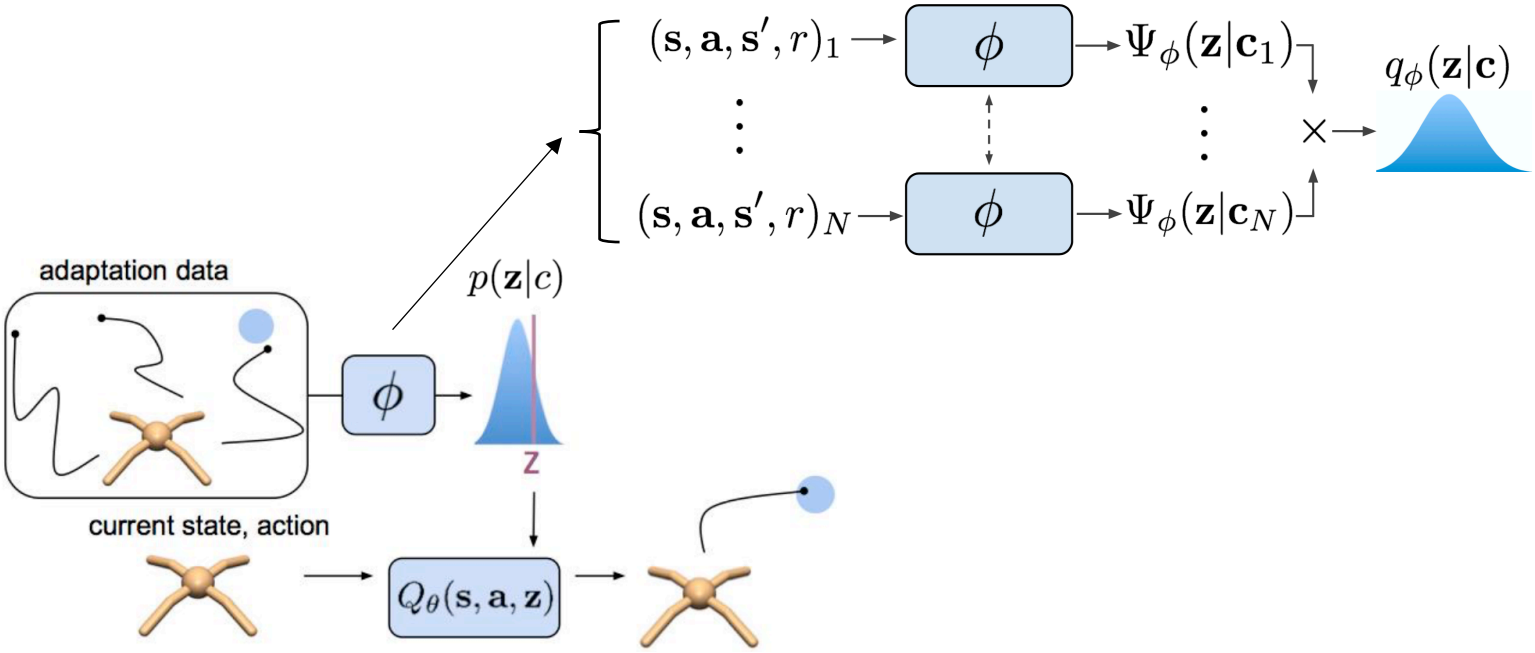
Designing architecture of inference network



- If we would like to infer what the task is or identify the MDP, it is enough to have access to a collection of transitions, without regard for the order of transitions
- So, use a **permutation-invariant encoder** for simplicity and speed

Probabilistic Latent Context

Designing architecture of inference network



- If we would like to infer what the task is or identify the MDP, it is enough to have access to a collection of transitions, without regard for the order of transitions
- So, use a **permutation-invariant encoder** for simplicity and speed
- To keep the method tractable, we use product of Gaussians, which result in Gaussian posterior

Off-Policy Meta-RL

Aside: Soft Actor-Critic (SAC)

- "Soft": maximize rewards and entropy of the policy (higher entropy policies explore better)

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t)}\left[r(s_t, a_t) + \alpha H\left(\pi(\cdot\,|s_t)\right)\right]$$

- "Actor-Critic": model both the actor (aka the policy) and the critic (aka the Q-function)

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t)}\left[\frac{1}{2}\left(Q_\theta(s_t, a_t) - \left[r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_{\bar{\theta}}(s_{t+1})]\right]\right)^2\right]$$

$$J_\pi(\phi) = \mathbb{E}_{(s_t, a_t)}\left[Q_\theta(s_t, a_t) + \alpha H(\pi_\phi(\cdot\,|s_t))\right]$$

- Much more **sample efficient** than on-policy algorithms!

Off-Policy Meta-RL

Algorithm 1 PEARL Meta-training

Require: Batch of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$, learning rates $\alpha_1, \alpha_2, \alpha_3$

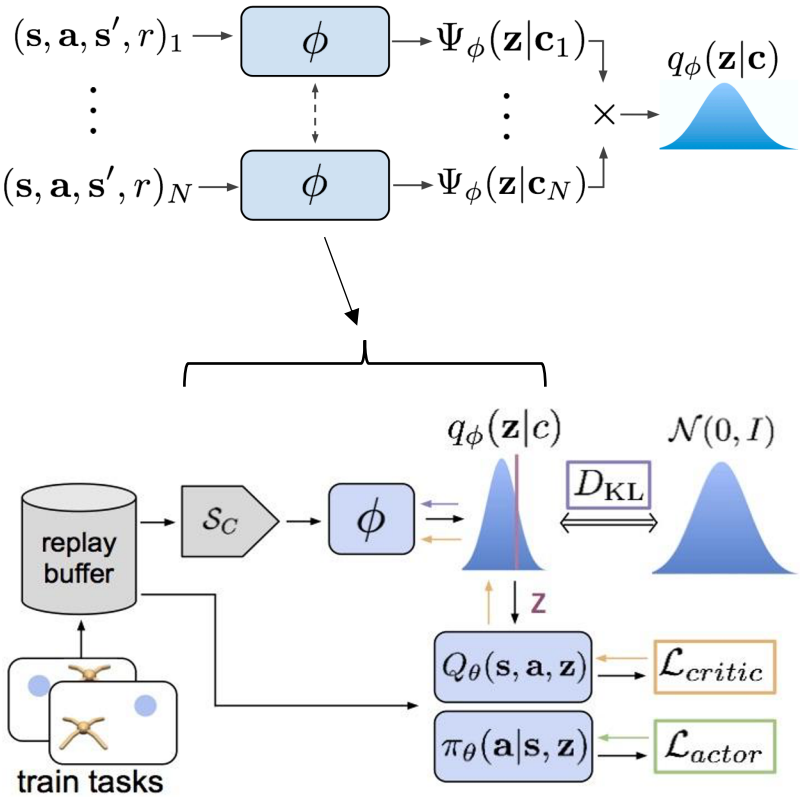
Initialize replay buffers $\mathcal{B}^i$ for each training task
while not done **do**
  for each $\mathcal{T}_i$ **do**
    Initialize context $\mathbf{c}^i = \{\}$
    for $k = 1, \ldots, K$ **do**
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      Gather data from $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ and add to $\mathcal{B}^i$
      Update $\mathbf{c}^i = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N} \sim \mathcal{B}^i$
    end for
  end for
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context $\mathbf{c}^i \sim \mathcal{S}_c(\mathcal{B}^i)$ and RL batch $b^i \sim \mathcal{B}^i$
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      $\mathcal{L}^i_{actor} = \mathcal{L}_{actor}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{critic} = \mathcal{L}_{critic}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{KL} = \beta D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{c}^i)||r(\mathbf{z}))$
    end for
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i (\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL})$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{critic}$
  end for
end while

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

SAC

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

Probabilistic encoder



Off-Policy Meta-RL

Algorithm 1 PEARL Meta-training

Require: Batch of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$, learning rates $\alpha_1, \alpha_2, \alpha_3$

Initialize replay buffers $\mathcal{B}^i$ for each training task
while not done **do**
  for each $\mathcal{T}_i$ **do**
    Initialize context $\mathbf{c}^i = \{\}$
    for $k = 1, \ldots, K$ **do**
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      Gather data from $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ and add to $\mathcal{B}^i$
      Update $\mathbf{c}^i = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N} \sim \mathcal{B}^i$
    end for
  end for
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context $\mathbf{c}^i \sim \mathcal{S}_c(\mathcal{B}^i)$ and RL batch $b^i \sim \mathcal{B}^i$
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      $\mathcal{L}^i_{actor} = \mathcal{L}_{actor}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{critic} = \mathcal{L}_{critic}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{KL} = \beta D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{c}^i)||r(\mathbf{z}))$
    end for
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i \left(\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL}\right)$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{critic}$
  end for
end while

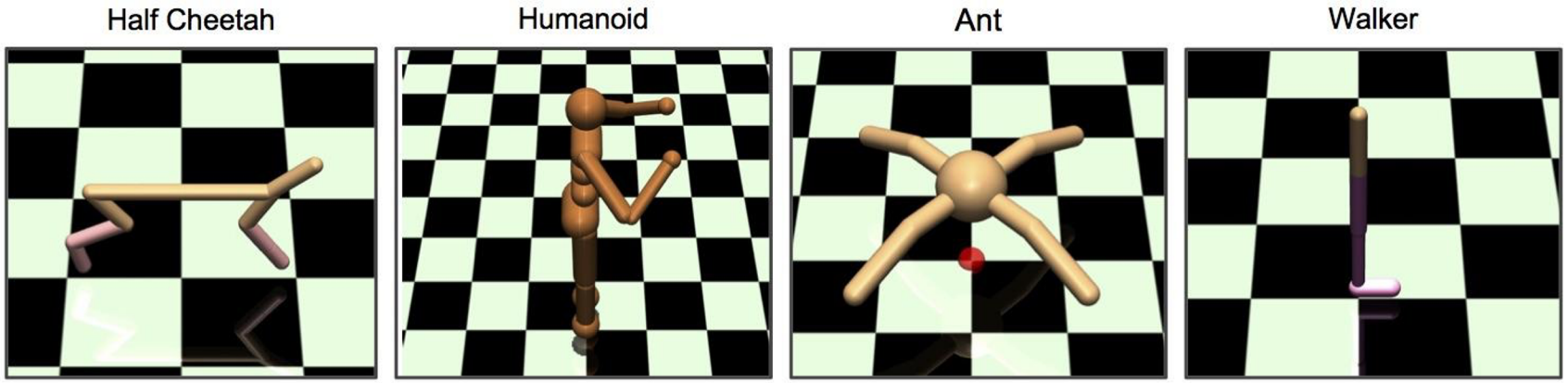
Algorithm 2 PEARL Meta-testing

Require: test task $\mathcal{T} \sim p(\mathcal{T})$

Initialize context $\mathbf{c}^\mathcal{T} = \{\}$
for $k = 1, \ldots, K$ **do**
  Sample $z \sim q_\phi(\mathbf{z}|c^\mathcal{T})$
  Roll out policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ to collect data $D^\mathcal{T}_k = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N}$
  Accumulate context $\mathbf{c}^\mathcal{T} = \mathbf{c}^\mathcal{T} \cup D^\mathcal{T}_k$
end for

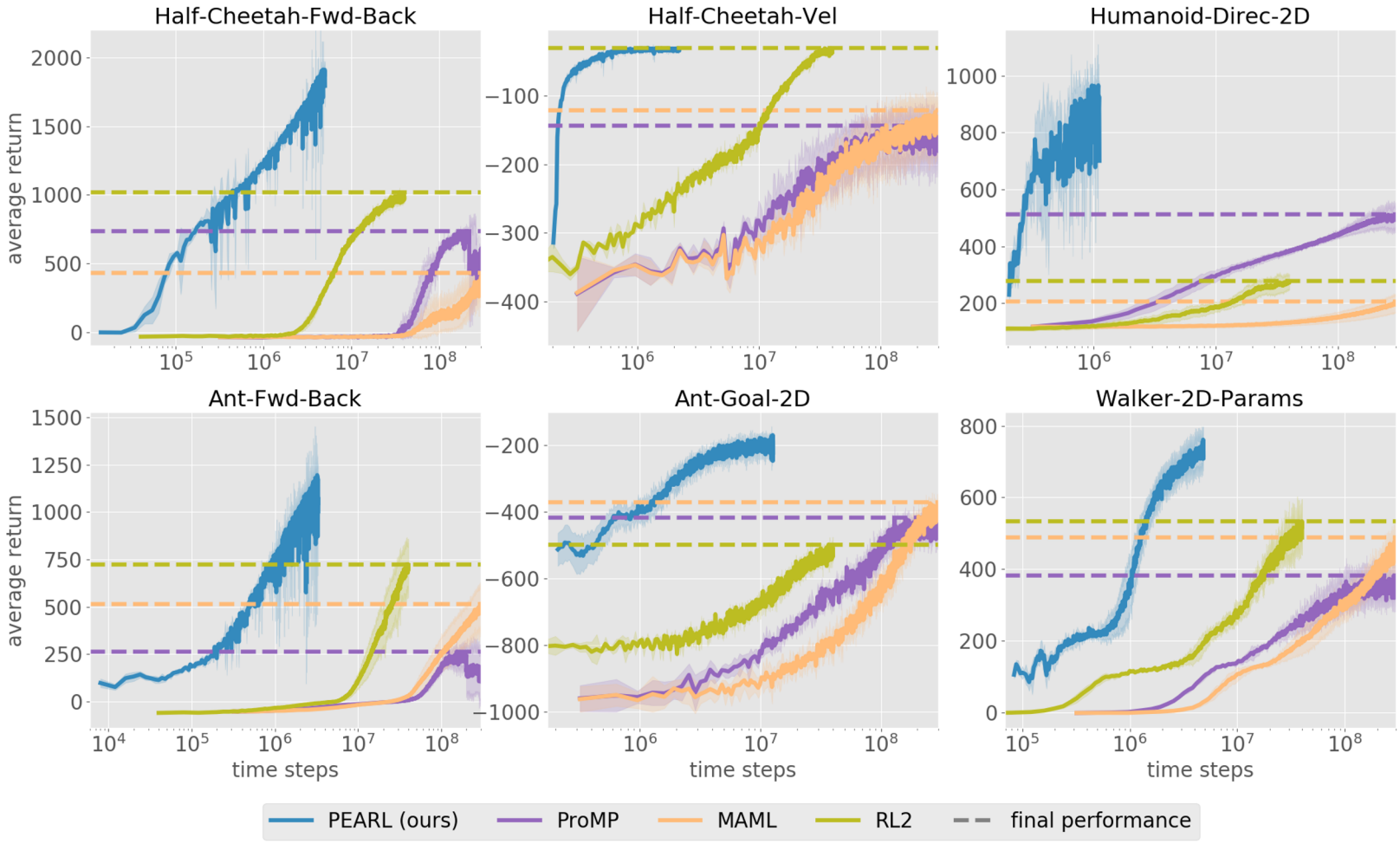
Experiments

Experimental setup



Experiments

Experimental results



Experiments

Ablations I: inference network architecture

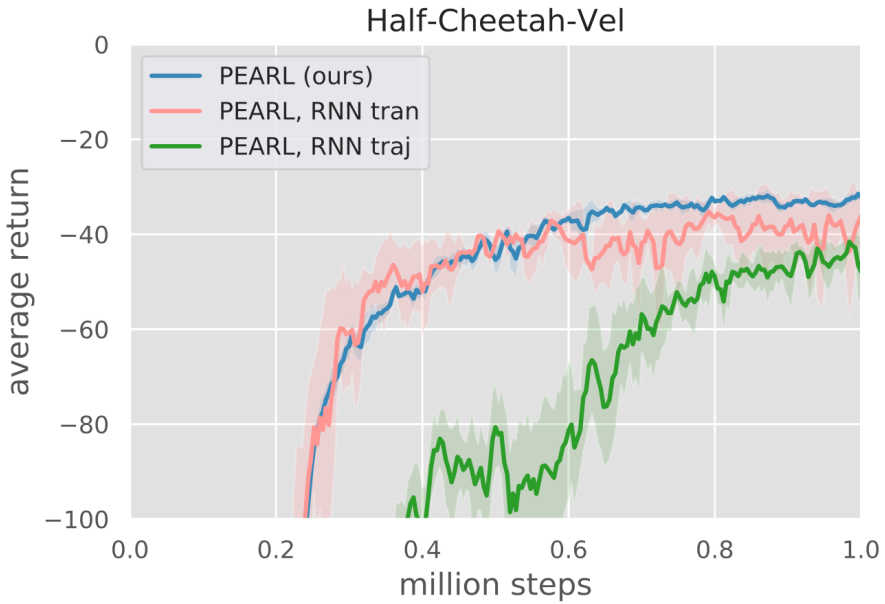


Figure 5. **Recurrent encoder ablation**. We compare our encoder architecture to a recurrent network. We sample context as trajectories rather than unordered transitions. Sampling the RL batch as de-correlated transitions ("RNN tran") fares much better than sampling trajectories ("RNN traj").

- This result demonstrates **the importance of decorrelating the samples** used for the RL objective

Experiments

Ablations II: data sampling strategies

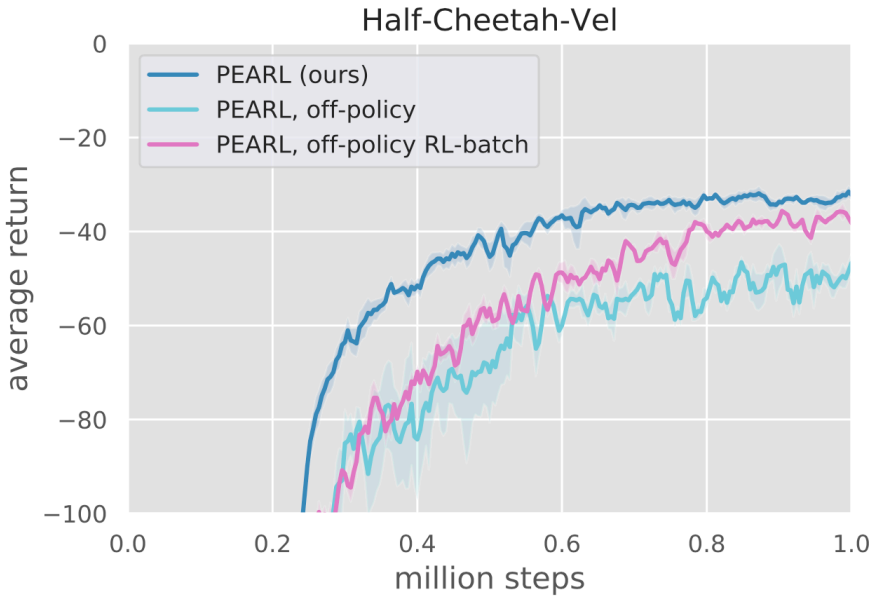


Figure 6. **Context sampling ablation.** PEARL samples context batches of recently collected transitions de-correlated with the batches sampled for RL. We compare to sampling context from the entire history ("off-policy"), as well as using the same sampled batch for the context and the RL batch ("off-policy RL").

- This result demonstrate **the importance of careful data sampling in off-policy** meta-RL

Experiments

Ablations III: deterministic context

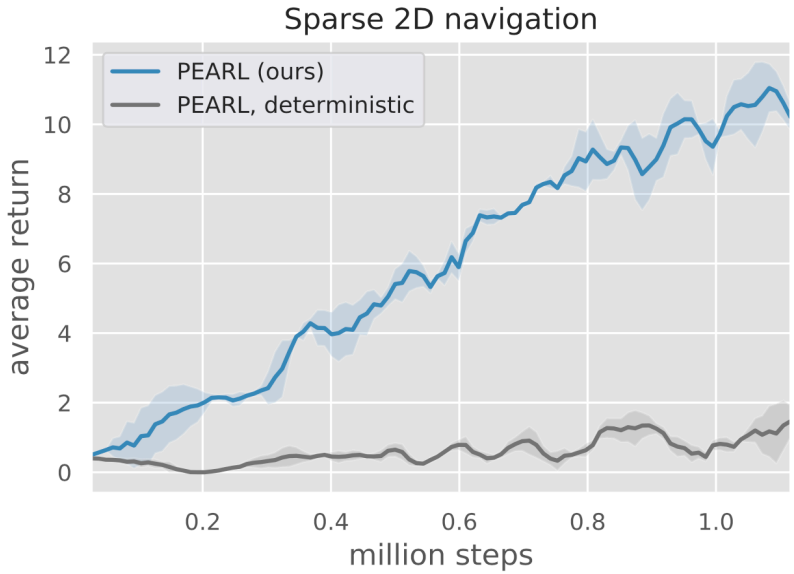


Figure 7. **Deterministic latent context**. We compare PEARL to a variant with deterministic latent context on the sparse reward 2D navigation domain. As expected, without a mechanism for reasoning about uncertainty over tasks, this approach is unable to explore effectively and performs poorly.

- With no stochasticity in the latent context variable, the only stochasticity comes from the policy and is thus **time-invariant, hindering temporally extended exploration**

Thank You!

Any Questions?