Advanced Topics in Deep Reinforcement Learning

MetaRL 1: Meta-Gradients

Theresa Eimer

What Is Meta-Learning?

Nobody knows

What Is Meta-Learning?

The community does not fully agree

What Is Meta-Learning?

The community does not fully agree

Characteristics:

- Learning about algorithms or algorithm components
- Enhancing algorithm capabilities (e.g. speed, generalization, etc.)
- Combining optimization paradigms

What Is MetaRL?

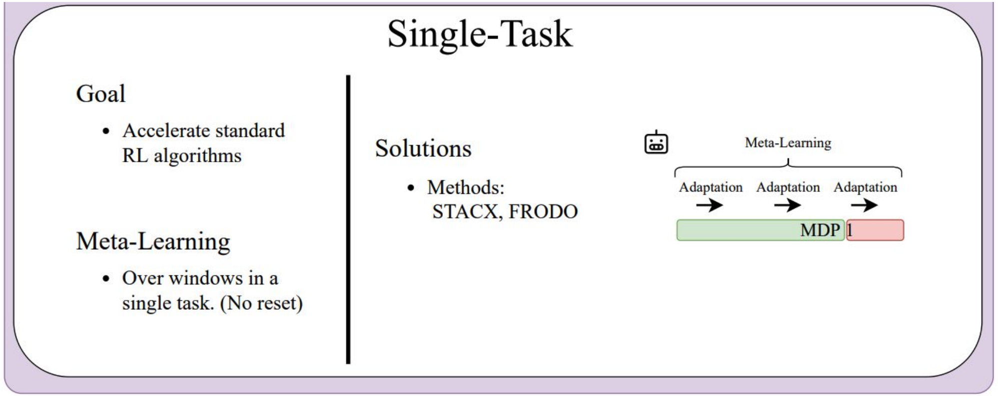
Definition by Beck et al. 2023:

“Where RL learns a policy, meta-RL learns the RL algorithm f that outputs the policy.”

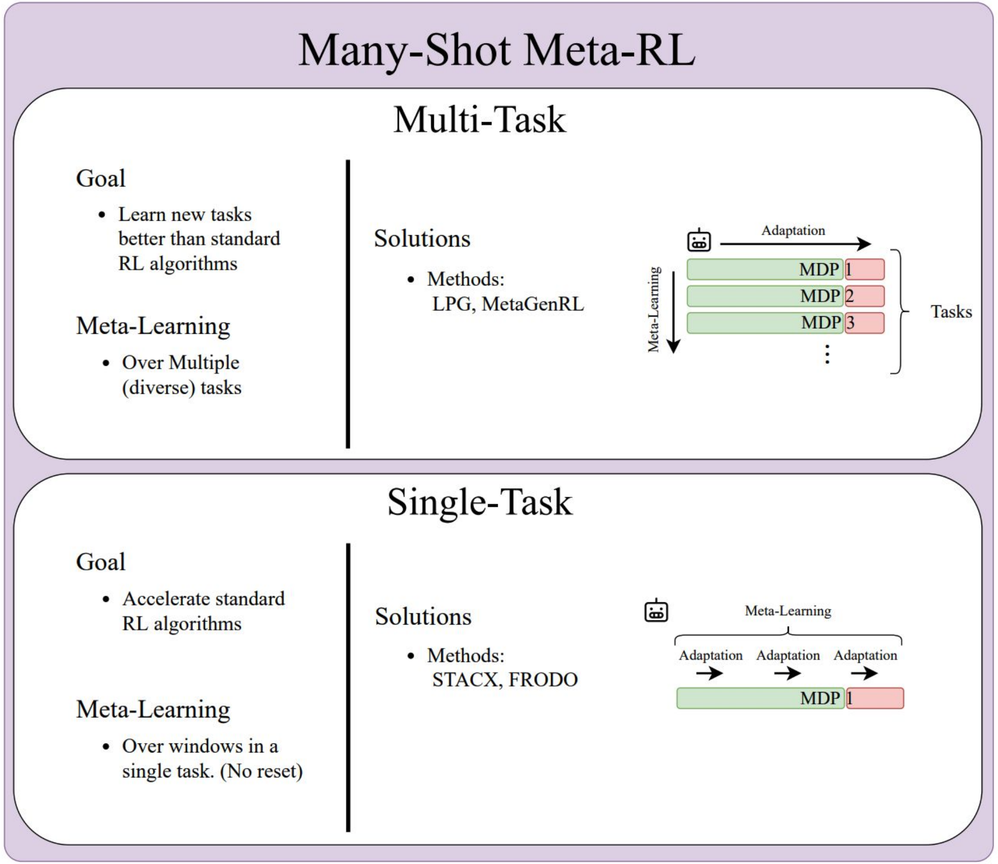
Performance is measured across a set of target tasks instead of a single one:

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{M}^i \sim p(\mathcal{M})}\left[\mathbb{E}_{\mathcal{D}}\left[\sum_{\tau \in \mathcal{D}_{K:H}} G(\tau) \middle| f_\theta, \mathcal{M}^i\right]\right]$$

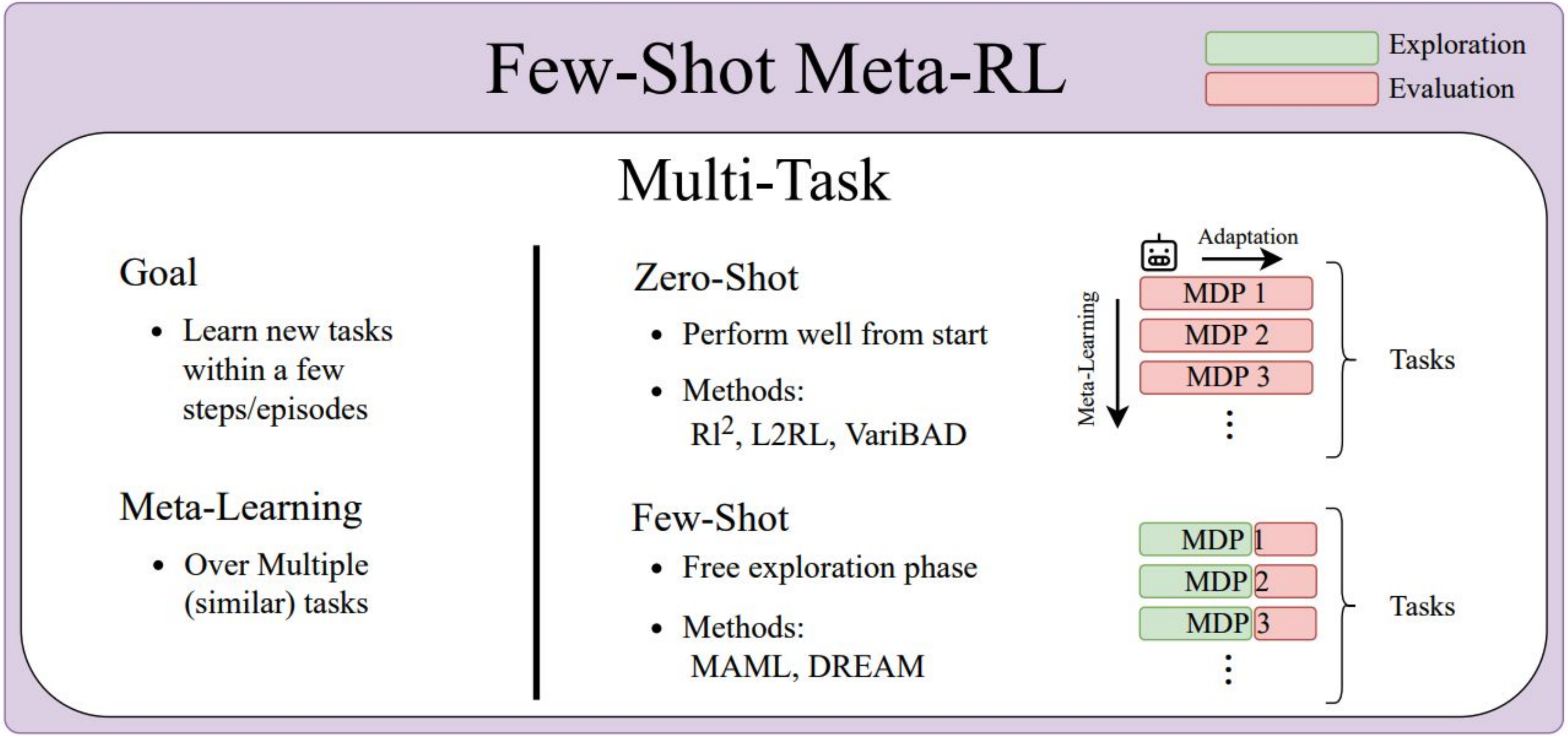
MetaRL Settings [Beck et al. 2023]



MetaRL Settings [Beck et al. 2023]



MetaRL Settings [Beck et al. 2023]



MetaRL vs AutoRL

Meta RL is similar to AutoRL and there is overlap:

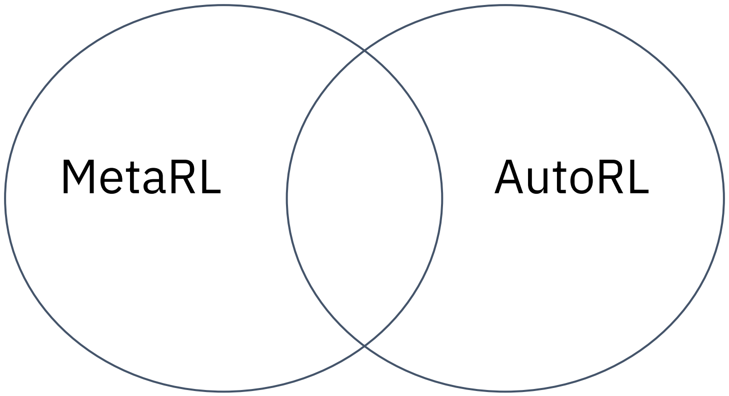
- Both are concerned with improving the function that produces the policy
- The MetaRL settings apply to AutoRL as well
- A lot of MetaRL is AutoRL and vice versa

MetaRL vs AutoRL

Meta RL is similar to AutoRL and there is overlap:

- Both are concerned with improving the function that produces the policy
- The MetaRL settings apply to AutoRL as well
- A lot of MetaRL is AutoRL and vice versa

Generalization focus



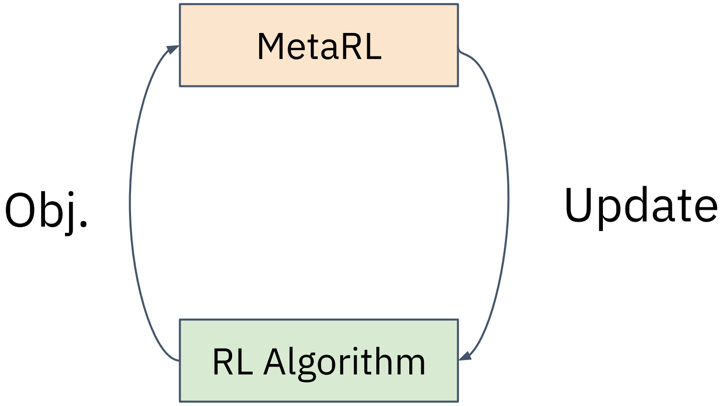
Tuning focus

This Week: Gradient Approaches

There are two MetaRL paradigms

This Week: Gradient Approaches

There are two MetaRL paradigms:

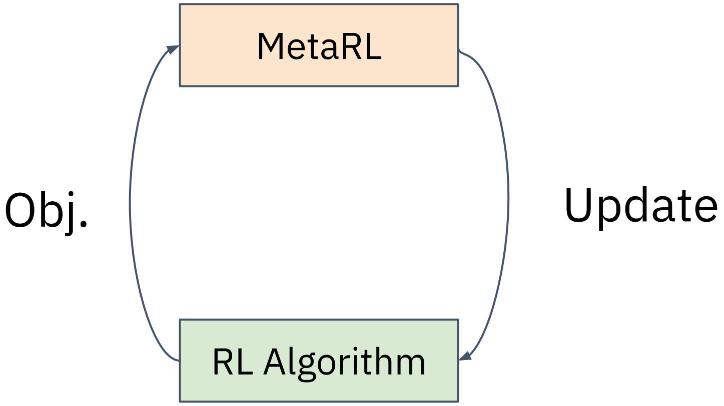


Full Run

Outer loop MetaRL

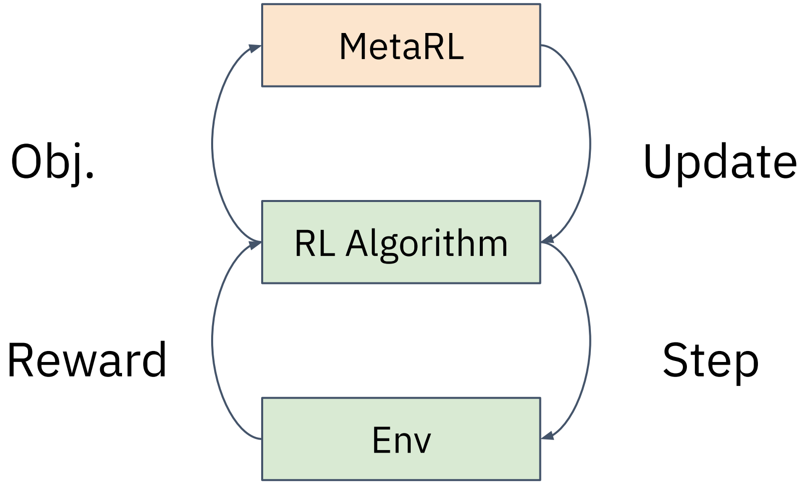
This Week: Gradient Approaches

There are two MetaRL paradigms:



Full Run

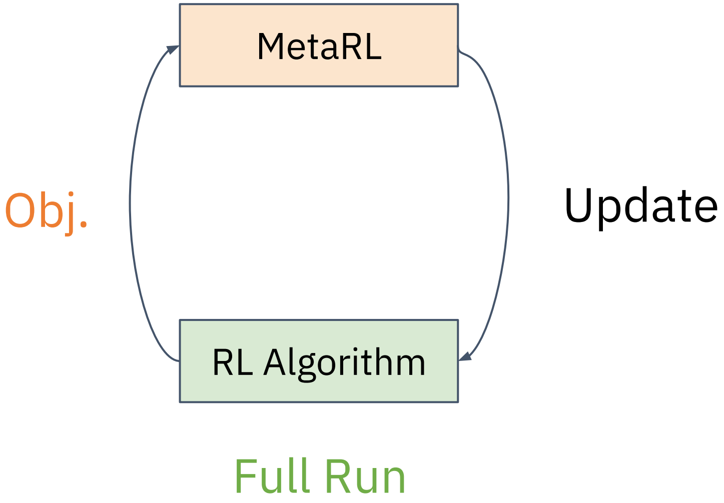
Outer loop MetaRL



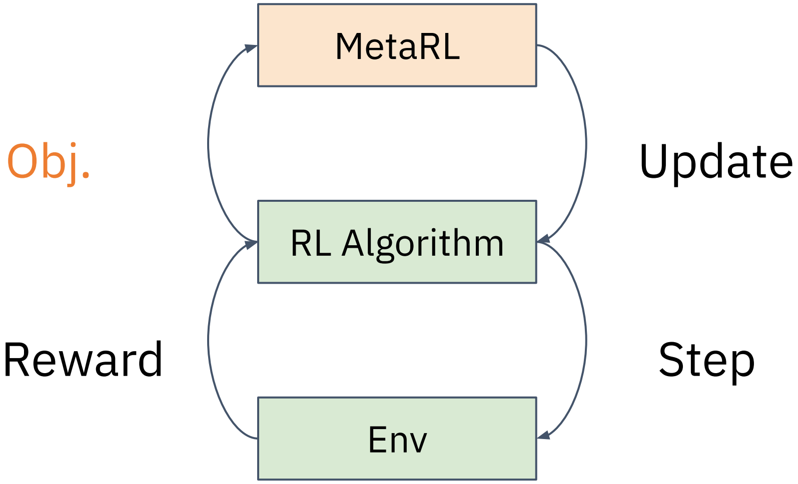
In-the-loop MetaRL

This Week: Gradient Approaches

There are two MetaRL paradigms:



Outer loop MetaRL



In-the-loop MetaRL

This Week: Gradient Approaches

There are two MetaRL paradigms:

Outer loop MetaRL

- Acts on full algorithm runs
- Performance estimate via return across tasks
- Can be relatively algorithm agnostic

In-the-loop MetaRL

- Acts during algorithm runs (though can be few shot)
- Can use additional features for performance estimation
- Thus harder to apply across different algorithms

Meta-Gradients Revisited

Alternate term for meta-gradients: second order optimization

Prerequisite: differentiable meta-objective function $J'(\tau', \theta', \eta')$

Meta-Gradients Revisited

Alternate term for meta-gradients: second order optimization

Prerequisite: differentiable meta-objective function $J'(\tau', \theta', \eta')$

Sample Policy Meta-decision (e.g. HP)

Meta-Gradients Revisited [Xu et al. 2018]

Alternate term for meta-gradients: second order optimization

Prerequisite: differentiable meta-objective function $J'(\tau', \theta', \eta')$

We update the meta-parameters s.t. we predict η for a better J'

$$\frac{\partial J'(\tau', \theta', \eta')}{\partial \eta} = \frac{\partial J'(\tau', \theta', \eta')}{\partial \theta'} \frac{\mathrm{d}\theta'}{\mathrm{d}\eta}$$

Meta-Gradients Revisited [Xu et al. 2018]

$$\frac{\partial J'(\tau', \theta', \eta')}{\partial \eta} = \frac{\partial J'(\tau', \theta', \eta')}{\partial \theta'} \frac{\mathrm{d}\theta'}{\mathrm{d}\eta}$$

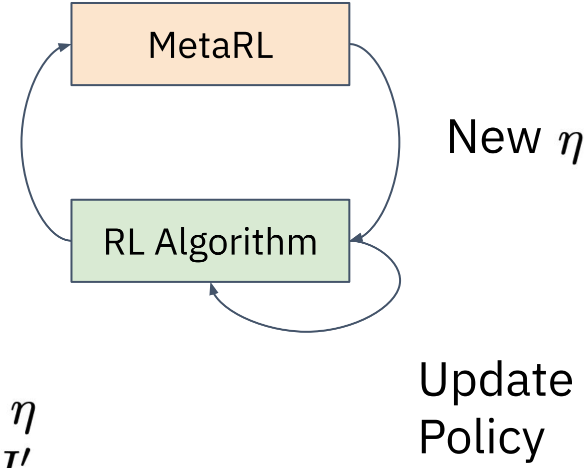
In practice, we use $z \approx \mathrm{d}\theta / \mathrm{d}\eta$ with:

$$z' = \underbrace{\mu}_{\text{Decay Factor}} z + \underbrace{\frac{\partial f(\tau, \theta, \eta)}{\partial \eta}}_{\text{Single Step}}$$

Meta-Gradients Revisited [Xu et al. 2018]

Update with
Meta-Objective:

- Evaluate policy
- Update trace with gradient of policy wrt. η
- Compute gradient of J' wrt. η



Extensions of Meta-Gradients

Bootstrapped Meta-Gradients [Flennerhag et al. 2022]

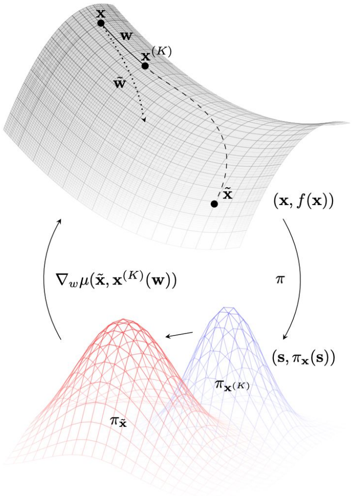
- **Goal:** make meta-objective more independent of the learner's objective and optimization less myopic
- Instead of focusing on the current rollout, bootstrapped targets extend the meta-optimization horizon (similar to TD Learning)

$$\tilde{\mathbf{x}} = \mathbf{x}^{(K+L-1)} - \alpha \nabla f(\mathbf{x}^{(K+L-1)})$$

Extensions of Meta-Gradients

Bootstrapped Meta-Gradients [Flennerhag et al. 2022]

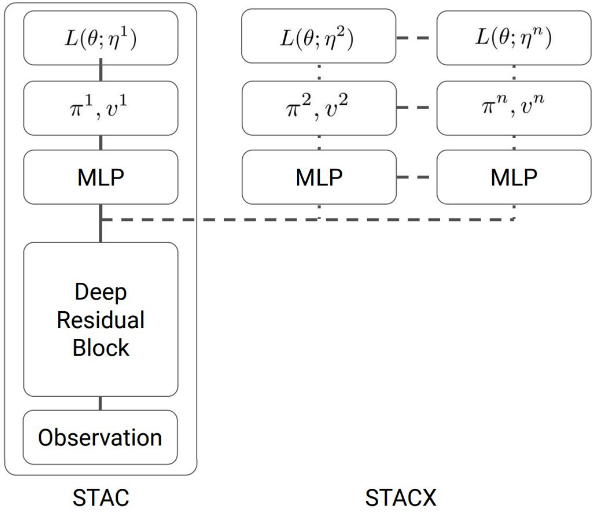
- **Goal:** make meta-objective more independent of the learner's objective and optimization less myopic
- Instead of focusing on the current rollout, bootstrapped targets extend the meta-optimization horizon (similar to TD Learning)
- Targets are matched to the learner's landscape s.t. gradients don't look too dissimilar and are easier to learn



Extensions of Meta-Gradients

Self-Tuning Actor Critic (STAC) [Zahavy et al. 2021]

- **Goal:** Make an AC algorithm that needs no tuning and generalizes well
- Tunes discount factor, GAE lambda and loss weights of IMPALA
- V-trace needs to be adapted to be differentiable
- Optional add-on: meta-learned auxiliary tasks for improved representations (STACX)



Meta-Gradients for Objectives

MetaRL is very flexible, we can learn anything we can parameterize:

- rewards
- environment variations
- hyperparameters
- algorithms
- algorithm components

Meta-Gradients for Objectives

MetaRL is very flexible, we can learn anything we can parameterize:

- rewards
- environment variations
- hyperparameters
- algorithms
- **algorithm components**

Meta-Gradients for Objectives

Why learn an objective function?

- Shapes the algorithm behavior
- Can encode concepts like optimism
- Is used for updates, so implicitly encodes several hyperparameters

Is this different from learning an algorithm?

- Algorithms also contain elements like exploration, loss and return computation, etc.
- Learning an algorithm is more complex than learning an objective

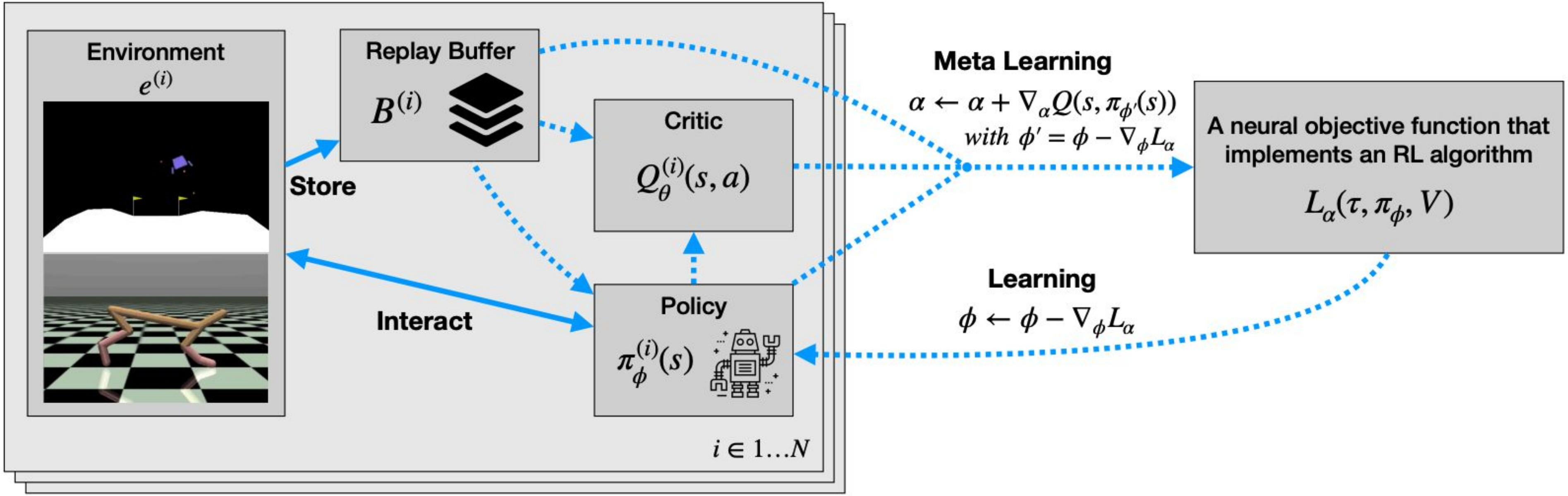
Meta-Gradients for Objectives

Important note: objective function can usually not be represented by a single number!

- Common approach: use a NN to learn objective functions
- Interesting future work: symbolic learning
- Alternative: learn a NN, evaluate often and fit a simple function on the result - limited complexity, but faster deployment [Lu et al. 2022]

Meta-Gradients for Objectives

MetaGenRL [Kirsch et al. 2020]:



Meta-Gradients for Objectives

Algorithm 1 MetaGenRL: Meta-Training

Require: $p(e)$ a distribution of environments

$P \Leftarrow \{(e_1 \sim p(e), \phi_1, \theta_1, B_1 \leftarrow \varnothing), \ldots\}$ ▷ Randomly initialize population of agents

Randomly initialize objective function L_α

while L_α has not converged **do**

  for $e, \phi, \theta, B \in P$ **do** ▷ For each agent i in parallel

    if extend replay buffer B **then**

      Extend B using π_ϕ in e

    Sample trajectories from B

    Update critic Q_θ using TD-error

    Update policy by following $\nabla_\phi L_\alpha$

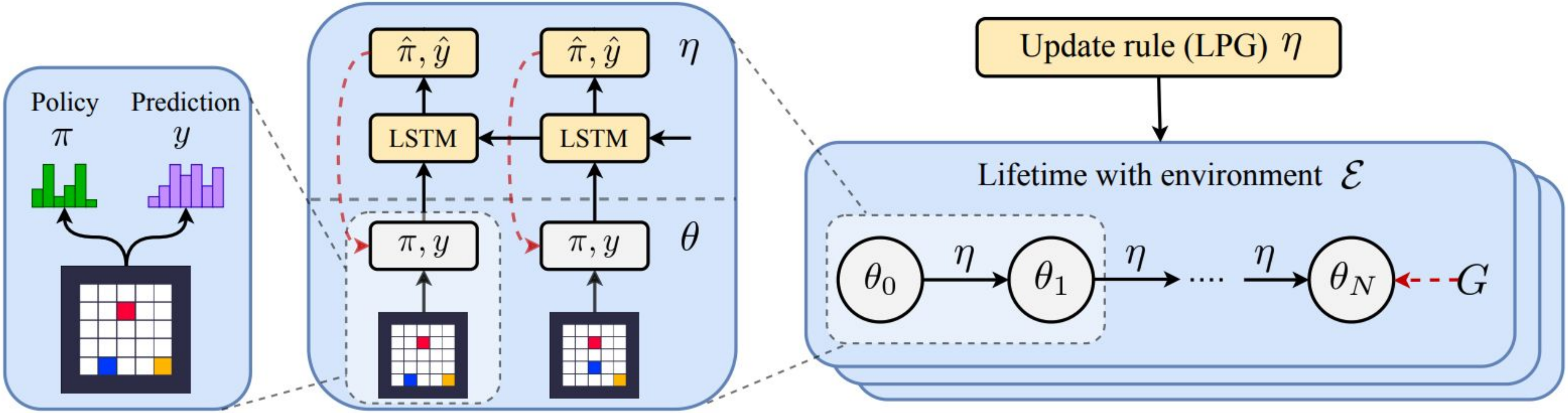
    Compute objective function gradient Δ_i for agent i according to Equation 6

  Sum gradients $\sum_i \Delta_i$ to update L_α

Meta-Gradients for Objectives

Learned Policy Gradient (LPG) [Oh et al. 2021]:

-



Meta-Gradients for Objectives

Algorithm 1 Meta-Training of Learned Policy Gradient

Input: $p(\mathcal{E})$: Environment distribution, $p(\theta_0)$: Initial agent parameter distribution
Initialise meta-parameters η and hyperparameter sampling distribution $p(\alpha|\mathcal{E})$
Sample batch of environment-agent-hyperparameters $\{\mathcal{E} \sim p(\mathcal{E}), \theta \sim p(\theta_0), \alpha \sim p(\alpha|\mathcal{E})\}_i$
repeat
  for all lifetimes $\{\mathcal{E}, \theta, \alpha\}_i$ **do**
    Update parameters θ using η and α for K times using Eq. (2)
    Compute meta-gradient using Eq. (4)
    if lifetime ended **then**
      Update hyperparameter sampling distribution $p(\alpha|\mathcal{E})$
      Reset lifetime $\mathcal{E} \sim p(\mathcal{E}), \theta \sim p(\theta_0), \alpha \sim p(\alpha|\mathcal{E})$
    end if
  end for
  Update meta-parameters η using the meta-gradients averaged over all lifetimes.
until η converges

Meta-Gradients for Objectives

Lessons from MetaGenRL & LPG:

- Learning objectives is likely harder than learning hyperparameters
- Tradeoff between expressiveness and ease of use
- Limited generalization to other tasks
- Computational expense becomes a big factor in this setting

My Understanding of MetaRL

- [ ] I can define MetaRL
- [ ] I understand the different MetaRL settings

- [ ] I know how MetaRL relates to AutoRL
- [ ] I can roughly describe one gradient-based MetaRL approach

- [ ] I know how meta-gradients can be used for MetaRL
- [ ] I could implement one of the algorithms from this lecture