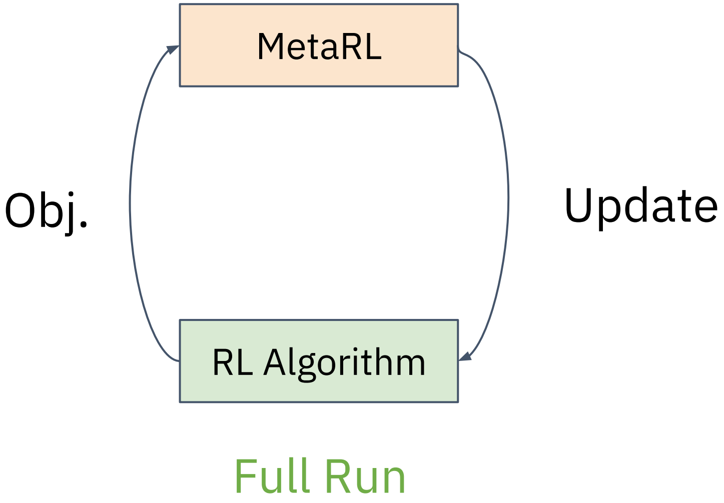
Advanced Topics in Deep Reinforcement Learning

MetaRL 1: Outer-Loop Learning

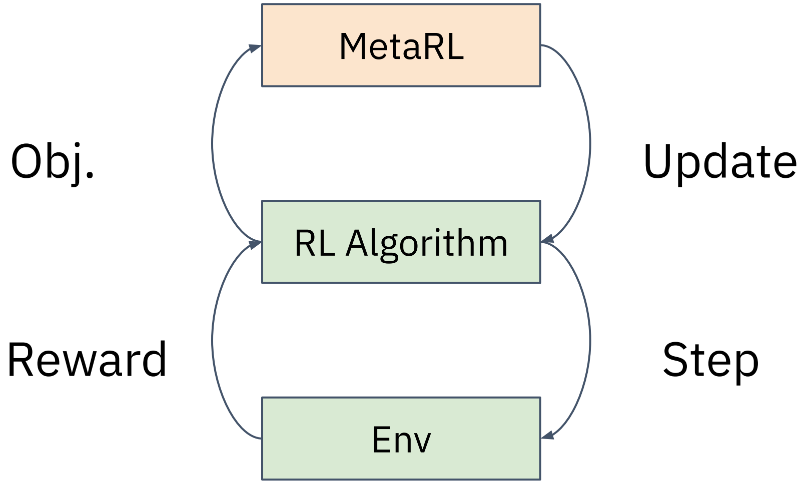
Theresa Eimer

Reminder: Outer-Loop vs In-the-Loop

There are two MetaRL paradigms:



Outer-loop MetaRL



In-the-loop MetaRL

Characteristics of Outer-Loop MetaRL

- Resets agents between evaluations

Characteristics of Outer-Loop MetaRL

- Resets agents between evaluations
- Usually full RL runs of a single algorithm

Characteristics of Outer-Loop MetaRL

- Resets agents between evaluations
- Usually full RL runs of a single algorithm
- Most commonly meta-gradients or black-box setting

Characteristics of Outer-Loop MetaRL

- Resets agents between evaluations
- Usually full RL runs of a single algorithm
- Most commonly meta-gradients or black-box setting
- Black-box objective usually evaluation reward

Characteristics of Outer-Loop MetaRL

- Resets agents between evaluations
- Usually full RL runs of a single algorithm
- Most commonly meta-gradients or black-box setting
- Black-box objective usually evaluation reward
- Has been successful in learning most components of the RL loop

The Template: MAML [Finn et al. 2017]

- Very successful metaRL with a lot of follow-up work

The Template: MAML [Finn et al. 2017]

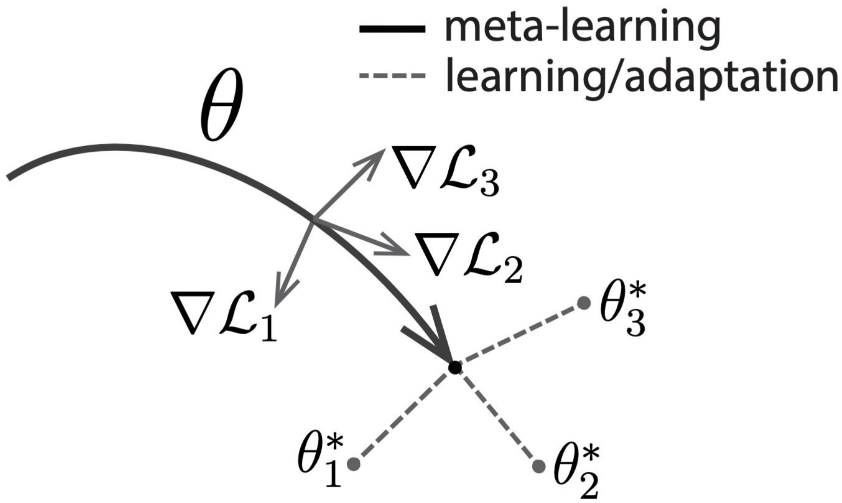
- Very successful metaRL with a lot of follow-up work
- Goal: meta-learn a good set of starting parameters to later finetune

The Template: MAML [Finn et al. 2017]

- Very successful metaRL with a lot of follow-up work
- Goal: meta-learn a good set of starting parameters to later finetune
- Method: Train local policy copies on a set of task and then update the meta-parameters using their performance

The Template: MAML [Finn et al. 2017]

- Very successful metaRL with a lot of follow-up work
- Goal: meta-learn a good set of starting parameters to later finetune
- Method: Train local policy copies on a set of task and then update the meta-parameters using their performance



The Template: MAML [Finn et al. 2017]

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: **end while**

Evaluate meta-parameters

The Template: MAML [Finn et al. 2017]

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ ⇦ Do local update
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: **end while**

The Template: MAML [Finn et al. 2017]

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: **end while**

⇦ Evaluate local update

The Template: MAML [Finn et al. 2017]

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Sample K trajectories D = {(x_1, a_1, ...x_H)} using f_θ in T_i
        Evaluate ∇_θ L_{T_i}(f_θ) using D and L_{T_i} in Equation 4
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
        Sample trajectories D'_i = {(x_1, a_1, ...x_H)} using f_{θ'_i} in T_i
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i}) using each D'_i and L_{T_i} in Equation 4
end while
```

Propagate local updates

The Template: MAML [Finn et al. 2017]

- Simple method without much overhead
- No meta-model to keep track of
- No additional hyperparameters
- Still performs fairly well as a baseline

The Template: MAML [Finn et al. 2017]

- Simple method without much overhead
- No meta-model to keep track of
- No additional hyperparameters
- Still performs fairly well as a baseline
- But: we need meta-gradients for the meta-parameter update which is expensive and difficult to compute

Alternative: Reptile [Nichol et al. 2018]

What if we just ignore the meta-gradients?

Alternative: Reptile [Nichol et al. 2018]

What if we just ignore the meta-gradients?

Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

Alternative: Reptile [Nichol et al. 2018]

What if we just ignore the meta-gradients?

- Directly interpolating meta-parameters using the local parameters
- Loss of some gradient information
- Could cause instability depending on task diversity

Alternative: Reptile [Nichol et al. 2018]

What if we just ignore the meta-gradients?

- Directly interpolating meta-parameters using the local parameters
- Loss of some gradient information
- Could cause instability depending on task diversity
- But: performs quite well in evaluation

ES-MAML [Song et al. 2020]

Can we avoid meta-gradients in MAML? - Version 2

ES-MAML [Song et al. 2020]

Can we avoid meta-gradients in MAML? - Version 2

- Since we mainly care about performance on the task distribution, they are not necessary in the way they are in single-shot in-the-loop methods

ES-MAML [Song et al. 2020]

Can we avoid meta-gradients in MAML? - Version 2

- Since we mainly care about performance on the task distribution, they are not necessary in the way they are in single-shot in-the-loop methods
- Final evaluation performance can take the place of the loss

ES-MAML [Song et al. 2020]

Can we avoid meta-gradients in MAML? - Version 2

- Since we mainly care about performance on the task distribution, they are not necessary in the way they are in single-shot in-the-loop methods
- Final evaluation performance can take the place of the loss
- Now we can use black-box methods as meta-optimizers

ES-MAML [Song et al. 2020]

Data: initial policy θ_0, meta step size β

for $t = 0, 1, \ldots$ **do**

  Sample n tasks $T_1, \ldots, T_n$ and iid vectors $\mathbf{g}_1, \ldots, \mathbf{g}_n \sim \mathcal{N}(0, \mathbf{I})$;

  foreach $(T_i, \mathbf{g}_i)$ **do**

    $v_i \leftarrow f^{T_i}(U(\theta_t + \sigma \mathbf{g}_i, T_i))$

  end

  $\theta_{t+1} \leftarrow \theta_t + \frac{\beta}{\sigma n} \sum_{i=1}^{n} v_i \mathbf{g}_i$

end

What Is “ES”?

- ES is short for “Evolutionary Strategies”

What Is “ES”?

- ES is short for “Evolutionary Strategies”
- Most popular approach for outer-loop metaRL

What Is “ES”?

- ES is short for “Evolutionary Strategies”
- Most popular approach for outer-loop metaRL
- They have even be used as RL algorithms [Salimans et al. 2017]

What Is “ES”?

- ES is short for “Evolutionary Strategies”
- Most popular approach for outer-loop metaRL
- They have even be used as RL algorithms [Salimans et al. 2017]
- Interesting for metaRL since they are highly parallelizable

What Is “ES”?

- ES is short for “Evolutionary Strategies”
- Most popular approach for outer-loop metaRL
- They have even be used as RL algorithms [Salimans et al. 2017]
- Interesting for metaRL since they are highly parallelizable
- ES does not usually refer to a single algorithm, most common are:

What Is “ES”?

- ES is short for “Evolutionary Strategies”
- Most popular approach for outer-loop metaRL
- They have even be used as RL algorithms [Salimans et al. 2017]
- Interesting for metaRL since they are highly parallelizable
- ES does not usually refer to a single algorithm, most common are:
  - CMA-ES [Hansen and Ostermeier, 2001]: population is a multivariate Gaussian

What Is “ES”?

- ES is short for “Evolutionary Strategies”
- Most popular approach for outer-loop metaRL
- They have even be used as RL algorithms [Salimans et al. 2017]
- Interesting for metaRL since they are highly parallelizable
- ES does not usually refer to a single algorithm, most common are:
  - CMA-ES [Hansen and Ostermeier, 2001]: population is a multivariate Gaussian
  - NES [Wierstra et al., 2008]: population is a distribution over parameters

What Is “ES”?

Basic structure of an ES algorithm:

1. Generate a population of samples to evaluate
2. Evaluate to get fitness of each population member
3. Use fitness to select or mutate best population members

What Is “ES”?

Basic structure of an ES algorithm:

1. Generate a population of samples to evaluate
2. Evaluate to get fitness of each population member
3. Use fitness to select or mutate best population members

There are many ways to implement the sample procedure and research on improved ES algorithms is a research field in its own right.

Meta-Optimizers

- Meta-Gradients or second-order gradients
  - SGD update using the gradient of a gradient
  - Detailed information
  - Potentially noisy
  - Expensive to compute

Meta-Optimizers

- Meta-Gradients or second-order gradients
  - SGD update using the gradient of a gradient
  - Detailed information
  - Potentially noisy
  - Expensive to compute
- Evolutionary strategies
  - Evolving population of samples
  - Black-box, relies on reward
  - Potentially needs more function evaluations than meta-gradients
  - Easy to parallelize

MetaRL Settings

- Most commonly: train and test model-free online policies
- In most cases training tasks should not be tested on!
- Often still very similar training and test environments in the literature
- However: MetaRL also exists for other RL paradigms

Unsupervised MetaRL

- Different combinations of reward/reward-free meta-training and evaluation are possible

Unsupervised MetaRL

- Different combinations of reward/reward-free meta-training and evaluation are possible
- Most important one: unsupervised meta-learning for online rewards [Gupta et al. 2020]

Unsupervised MetaRL

- Different combinations of reward/reward-free meta-training and evaluation are possible
- Most important one: unsupervised meta-learning for online rewards [Gupta et al. 2020]
- Challenge: find meaningful parts of the state space to explore

Unsupervised MetaRL

- Different combinations of reward/reward-free meta-training and evaluation are possible
- Most important one: unsupervised meta-learning for online rewards [Gupta et al. 2020]
- Challenge: find meaningful parts of the state space to explore
- Can be used to find policies, but also to meta-learn e.g. curricula via mutual information or similar metrics [Jabri et al. 2019]

Model-based MetaRL

- Since we already learn a model, learning a model with meta-capabilities makes intuitive sense

Model-based MetaRL

- Since we already learn a model, learning a model with meta-capabilities makes intuitive sense
- Example: have the model infer the task specification [Belkhale et al. 2021]

Model-based MetaRL

- Since we already learn a model, learning a model with meta-capabilities makes intuitive sense
- Example: have the model infer the task specification [Belkhale et al. 2021]
- We can even use this to learn to switch between policies for different tasks, e.g. different robot models [Anne et al. 2021]

Model-based MetaRL

- Since we already learn a model, learning a model with meta-capabilities makes intuitive sense
- Example: have the model infer the task specification [Belkhale et al. 2021]
- We can even use this to learn to switch between policies for different tasks, e.g. different robot models [Anne et al. 2021]
- We can also condition the model itself on a known task specification to make it more useful [Prasanna et al. 2024]

Offline MetaRL

- Very little work
- Likely cause: offline RL itself is hard not well solved
- Dataset collection becomes even harder [Dorfman et al. 2021]

Offline MetaRL

- Very little work
- Likely cause: offline RL itself is hard not well solved
- Dataset collection becomes even harder [Dorfman et al. 2021]
- Potential fixes:

Offline MetaRL

- Very little work
- Likely cause: offline RL itself is hard not well solved
- Dataset collection becomes even harder [Dorfman et al. 2021]
- Potential fixes:
  - synthetic meta-exploration data [Rafailov et al. 2021]

Offline MetaRL

- Very little work
- Likely cause: offline RL itself is hard not well solved
- Dataset collection becomes even harder [Dorfman et al. 2021]
- Potential fixes:
  - synthetic meta-exploration data [Rafailov et al. 2021]
  - In-between stage of unsupervised online data collection [Pong et al. 2022]

Offline MetaRL

- Very little work
- Likely cause: offline RL itself is hard not well solved
- Dataset collection becomes even harder [Dorfman et al. 2021]
- Potential fixes:
  - synthetic meta-exploration data [Rafailov et al. 2021]
  - In-between stage of unsupervised online data collection [Pong et al. 2022]
  - Use simple offline RL algorithms [Mitchell et al. 2021]

My Understanding of MetaRL (pt. 2)

- ❑ I can describe the difference between outer-loop and in-the-loop methods
- ❑ I can name one meta-optimization method

- ❑ I can explain the intuition of MAML
- ❑ I understand the difference between meta-gradients and gradients on the meta-level

- ❑ I know an improvement on MAML
- ❑ I can describe two meta-optimizers