Meta Reinforcement Learning

Kate Rakelly
11/13/19
Questions we seek to answer

**Motivation**: What problem is meta-RL trying to solve?

**Context**: What is the connection to other problems in RL?

**Solutions**: What are solution methods for meta-RL and their limitations?

**Open Problems**: What are the open problems in meta-RL?
Meta-learning problem statement

**supervised learning**

- "Dalmation"
- "German shepherd"
- "Pug"
- corgi

**reinforcement learning**

Robot art by Matt Spangler, mattspangler.com
Meta-RL problem statement

**Regular RL**: learn policy for single task

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

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

**Meta-RL**: learn adaptation rule

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

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

**Meta-training / Outer loop**

**Adaptation / Inner loop**

MDP

$\mathcal{M}_1$  $\mathcal{M}_2$  $\mathcal{M}_3$  $\mathcal{M}_{test}$
Meta-RL can be viewed as a goal-conditioned policy where the task information is inferred from *experience*.

Task information could be about the dynamics or reward functions.

Rewards are a strict generalization of goals.
Q: What is an example of a reward function that can’t be expressed as a goal state?

A: E.g., seek while avoiding, action penalties
Adaptation

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

where \( \phi_i = f_{\theta}(M_i) \)

What should the adaptation procedure do?

- **Explore**: Collect the most informative data

- **Adapt**: Use that data to obtain the optimal policy
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 $\pi_{\phi_i}$
4. update $\theta$ according to $\mathcal{L}(\mathcal{D}'_i, \phi_i)$

Can do more than one round of adaptation

In practice, compute update across a batch of tasks

Different algorithms:
- Choice of function $f$
- Choice of loss function $\mathcal{L}$
Solution Methods
Solution #1: recurrence

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

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

where \( \phi_i = f_{\theta}(M_i) \)

Persist the hidden state across episode boundaries for continued adaptation!

Solution #1: recurrence

while training:
  for i in tasks:
    initialize hidden state $h_0 = 0$
  for $t$ in timesteps:
    1. sample 1 transition $D_i = D_i \cup \{(s_t, a_t, s_{t+1}, r_t)\}$ from $\pi_{h_t}$
    2. update policy hidden state $h_{t+1} = f_\theta(h_t, s_t, a_t, s_{t+1}, r_t)$

update policy parameters $\theta \leftarrow \theta - \nabla_\theta \sum_i L_i(D_i, \pi_h)$
Solution #1: recurrence

**Pro: general, expressive**

There exists an RNN that can compute any function

**Con: not consistent**

What does it mean for adaptation to be “consistent”?

*Will converge to the optimal policy given enough data*
Solution #1: recurrence

is pretraining a type of meta-learning?
better features = faster learning of new task!

Sample inefficient, prone to overfitting, and is particularly difficult in RL
Solution #2: optimization

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

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

where \( \phi_i = f_{\theta}(M_i) \)

Finn et al. 2017. Fig adapted from Finn et al. 2017
Solution #2: optimization

while training:
  for $i$ in tasks:
    1. sample $k$ episodes $\mathcal{D}_i = \{(s, a, s', r)\}_{1:k}$ from $\pi_\theta$
    2. compute adapted parameters $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_i(\pi_\theta, \mathcal{D}_i)$
    3. sample $k$ episodes $\mathcal{D}'_i = \{(s, a, s', r)\}_{1:k}$ from $\pi_{\phi_i}$

update policy parameters $\theta \leftarrow \theta - \nabla_\theta \sum_i \mathcal{L}_i(\mathcal{D}'_i, \pi_{\phi_i})$

Requires second order derivatives!

Finn et al. 2017. Fig adapted from Finn et al. 2017
Solution #2: optimization

How exploration is learned automatically

Pre-update parameters receive credit for producing good exploration trajectories

Causal relationship between pre and post-update trajectories is taken into account

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

where $$\phi_i = f_\theta(M_i)$$

Fig adapted from Rothfuss et al. 2018
Solution #2: optimization

\[ \nabla_\theta J(\theta) = \mathbb{E}_{T \sim \rho(T)} \left[ \mathbb{E}_{\tau \sim P_T(\tau | \theta)} \mathbb{E}_{\tau' \sim P_T(\tau' | \theta')} \left[ \nabla_\theta J_{\text{post}}(\tau, \tau') + \nabla_\theta J_{\text{pre}}(\tau, \tau') \right] \right] \]

\[ \nabla_\theta J_{\text{post}}(\tau, \tau') = \frac{\nabla_{\theta'} \log \pi_\theta(\tau') R(\tau')}{\nabla_{\theta'} J_{\text{outer}}} \left( I + \alpha R(\tau) \nabla_\theta^2 \log \pi_{\theta'}(\tau) \right) \]

\[ \nabla_\theta J_{\text{pre}}(\tau, \tau') = \alpha \nabla_\theta \log \pi_\theta(\tau) \left( \begin{pmatrix} \nabla_\theta \log \pi_\theta(\tau) R(\tau) \end{pmatrix}^T \begin{pmatrix} \nabla_{\theta'} \log \pi_{\theta'}(\tau') R(\tau') \end{pmatrix} \right) \]

View this as a “return” that encourages gradient alignment

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

where \( \phi_i = f_\theta(M_i) \)

Fig adapted from Rothfuss et al. 2018
Solution #2: optimization

**Pro:** consistent!

**Con:** not as expressive

**Q:** When could the optimization strategy be less expressive than the recurrent strategy?

Example: when no rewards are collected, adaptation will not change the policy, even though this data gives information about which states to avoid.

Suppose reward is given only in this region.

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

where \( \phi_i = f_{\theta}(M_i) \)

(a) Labryinth I-maze  
(b) Illustrative Episode
Solution #2: optimization

Exploring in a sparse reward setting

Cheetah running forward and back after 1 gradient step

Fig adapted from Rothfuss et al. 2018

Fig adapted from Finn et al. 2017
Meta-RL on robotic systems
Meta-imitation learning

Demonstration

1-shot imitation

Figure adapted from BAIR Blog Post: One-Shot Imitation from Watching Videos
Meta-imitation learning

Test: perform task given single robot demo
Training: run behavior cloning for adaptation

Meta-training
provide demonstration data

teleoperated robot demos
learn how to infer a policy from one demonstration

Test time
provide 1 demo with new object

infer robot policy

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

where \( \phi_i = f_\theta(M_i) \)

PG

Behavior cloning

\[ \phi_i = \theta - \alpha \nabla_{\theta} \sum_{t} \left| \pi_{\theta}(a_t) - a^*_t \right|^2 \]

Yu et al. 2017
Meta-imitation learning from human demos

demonstration

1-shot imitation

Figure adapted from BAIR Blog Post: One-Shot Imitation from Watching Videos
Meta-imitation learning from humans

Test: perform task given single human demo
Training: learn a loss function that adapts policy

Meta-training:
- provide demonstration data
- human demos
- learn how to infer a policy from one human demo

Test time:
- provide one video of human task 1 and task 2
- infer robot policy
- robot demos

\[ \theta^* = \arg \max_\theta \sum_{i=1}^n E_{\pi_{\phi_i}(\tau)}[R(\tau)] \]
where \( \phi_i = f_\theta(M_i) \)

\[ \phi = \theta - \alpha \nabla_\theta \mathcal{L}_\psi(\theta, d^h) \]

Supervised by paired robot-human demos only during meta-training!

Yu et al. 2018
Model-Based meta-RL

1. run base policy $\pi_0(a_t|s_t)$ (e.g., random policy) to collect $D = \{(s, a, s')_i\}$
2. learn dynamics model $f(s, a)$ to minimize $\sum_i \|f(s_i, a_i) - s'_i\|^2$
3. plan through $f(s, a)$ to choose actions

What if the system dynamics change?
- Low battery
- Malfunction
- Different terrain

Re-train model? :(

Figure adapted from Anusha Nagabandi
Model-Based meta-RL

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

where \( \phi_i = f_\theta(M_i) \)

MPC

Supervised model learning

\[ \theta' = h(\theta, \text{recent}) \]

Adapted model \( \theta' \)

Controller

Figure adapted from Anusha Nagabandi
Model-Based meta-RL

Simulation results

Video from Nagabandi et al. 2019
Break
Aside: POMDPs

observation gives incomplete information about the state

Example: incomplete sensor data

"That Way We Go" by Matt Spangler
The POMDP view of meta-RL

Two approaches to solve: 1) policy with memory (RNN) 2) explicit state estimation
Model belief over latent task variables

POMDP for unobserved state

- Goal state
- Where am I?
  - $p(h|c)$
- S0, S1, S2
  - s = S0
  - a = “left”, s = S0, r = 0

POMDP for unobserved task

- Goal for MDP 0
- Goal for MDP 1
- Goal for MDP 2
- What task am I in?
  - $p(z|c)$
- S0
  - s = S0
  - a = “left”, s = S0, r = 0
Model belief over latent task variables

POMDP for unobserved state

Goal state:

- S0
- S1
- S2

Where am I?

- \( p(h|c) \)

s = S0

a = “left”, s = S0, r = 0

POMDP for unobserved task

Goal for MDP 0

Goal for MDP 1

Goal for MDP 2

What task am I in?

- \( p(z|c) \)

s = S0

sample

a = “left”, s = S0, r = 0
Solution #3: task-belief states

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

where \( \phi_i = f_{\theta}(M_i) \)

Stochastic encoder
Solution #3: posterior sampling in action
Solution #3: belief training objective

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

where \( \phi_i = f_{\theta}(M_i) \)

Stochastic encoder

Variational approximations to posterior and prior

“Likelihood” term (Bellman error)

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

“Regularization” term / information bottleneck

See Control as Inference (Levine 2018) for justification of thinking of Q as a pseudo-likelihood
Don’t need to know the order of transitions in order to identify the MDP (Markov property)

Use a permutation-invariant encoder for simplicity and speed
Aside: Soft Actor-Critic (SAC)

“Soft”: Maximize rewards *and* entropy of the policy (higher entropy policies explore better)

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

“Actor-Critic”: Model *both* the actor (aka the policy) and the critic (aka the Q-function)

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[ \frac{1}{2} \left( Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t) \right)^2 \right]$$

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

Much more sample efficient than on-policy algs.
Soft Actor-Critic

- **Train task**
- **Replay buffer**
- **$Q_\theta(s, a)$**
  - $L_{critic}$
- **$\pi_\theta(a|s)$**
  - $L_{actor}$
Solution #3: task-belief + SAC

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

where \( \phi_i = f_\theta(M_i) \)

SAC

Stochastic encoder

\[ q_\phi(z|c) \]

\[ \mathcal{N}(0, I) \]

\[ D_{KL} \]

\[ q_\phi(z|c) \rightarrow \mathcal{N}(0, I) \]

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

Rakelly & Zhou et al. 2019
Meta-RL experimental domains

Simulated via MuJoCo (Todorov et al. 2012), tasks proposed by (Finn et al. 2017, Rothfuss et al. 2019)
ProMP (Rothfuss et al. 2019), MAML (Finn et al. 2017), RL2 (Duan et al. 2016)
ProMP (Rothfuss et al. 2019), MAML (Finn et al. 2017), RL2 (Duan et al. 2016)

20-100X more sample efficient!
two views of meta-RL

Mechanistic view

- Deep neural network model that can read in an entire dataset and make predictions for new datapoints
- Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task

Probabilistic view

- Extract prior information from a set of (meta-training) tasks that allows efficient learning of new tasks
- Learning a new task uses this prior and (small) training set to infer most likely posterior parameters
Summary

1. Perspective 1: just RNN it
   \[ \theta^* \]
   \[ (s_1, a_1, s_2, r_1) \rightarrow (s_2, a_2, s_3, r_2) \rightarrow (s_3, a_3, s_4, r_3) \rightarrow h_i \rightarrow s \]

everything needed to solve task

2. Perspective 2: bi-level optimization
   \[ f_\theta(M_i) = \theta + \alpha \nabla_\theta J_\theta(\theta) \]
   MAML for RL

3. Perspective 3: it's an inference problem!
   \[ \pi_\theta(a|s, z) \]
   \[ z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t}) \]

just perspective 1, but with stochastic hidden variables!

\[ i.e., \phi = z \]

just a particular architecture choice for these

Slide adapted from Sergey Levine and Chelsea Finn
Where do tasks come from?

Idea: generate self-supervised tasks and use them during meta-training

$$\max \ H[Z] - H[Z | S] + H[A | S, Z]$$

Point robot learns to explore different areas after the hallway

Ant learns to run in different directions, jump, and flip

Limitations

Assumption that skills shouldn’t depend on action not always valid

Distribution shift meta-train -> meta-test
How to explore efficiently in a new task?

Learn exploration strategies better...

Plain gradient meta-RL  Latent-variable model

Bias exploration with extra information...

- human-provided demo
- Robot attempt #1, w/ only demo info
- Robot attempt #2, w/ demo + reward info

Online meta-learning

Meta-training tasks are presented in a sequence rather than a batch

Finn et al. 2019
Meta-RL finds an adaptation procedure that can quickly adapt the policy to a new task.

Three main solution classes: RNN, optimization, task-belief and several learning paradigms: model-free (on and off policy), model-based, imitation learning.

Connection to goal-conditioned RL and POMDPs.

Some open problems (there are more!): better exploration, defining task distributions, meta-learning online.
References

**Recurrent meta-RL**
Fast Reinforcement Learning by Slow Reinforcement Learning, Duan et al. 2016
Memory-Based Control with Recurrent Neural Networks, Heess et al. 2015

**Optimization-based meta-RL**
Model-Agnostic Meta-Learning, Finn et al. 2017
Proximal Meta-Policy Search, Rothfuss et al. 2018

**Optimization-based meta-RL + imitation learning**
One-Shot Visual Imitation Learning via Meta-Learning, Yu et al. 2017
One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning, Yu et al. 2018

**Model-based meta-RL**
Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning, Nagabandi et al. 2019

**Off-policy meta-RL**
Soft Actor-Critic, Haarnoja et al. 2018
Control as Inference, Levine 2018.
Efficient Off-Policy Meta-RL via Probabilistic Context Variables, Rakelly et al. 2019
Open Problems
Diversity is All You Need: Learning Skills without a Reward Function, Eysenbach et al. 2018
Unsupervised Meta-learning for RL, Gupta et al. 2018
Meta-Reinforcement Learning of Structured Exploration Strategies, Gupta et al. 2018
Watch, Try, Learn, Meta-Learning from Demonstrations and Reward, Zhou et al. 2019
Online Meta-Learning, Finn et al. 2019

Slides and Figures
Some slides adapted from Meta-Learning Tutorial at ICML 2019, Finn and Levine
Robot illustrations by Matt Spangler, mattspangler.com