Meta Reinforcement Learning

Chelsea Finn
Why are humans so good at RL?

People have prior experience.

People have an existing representation of the world.

Can we learn a representation under which RL is fast?

**Key idea:** Explicitly optimize for such a representation

“Learn how to reinforcement learn”
Outline

Meta-RL Problem Formulation & Examples
Method Classes: Recurrent Models, Gradient-Based Models
Challenges & Latest Developments
The Meta-Learning Problem

Supervised Learning:
Inputs: $x$  
Outputs: $y$  
Data: $\{(x, y)_i\}$  

$$y = f(x; \theta)$$

Meta Supervised Learning:
Inputs: $D_{\text{train}}$  
Outputs: $y_{\text{test}}$  
Data: $\{D_i\}$  

$$y_{\text{test}} = f(D_{\text{train}}, x_{\text{test}}; \theta)$$

Why is this view useful?  
Reduces the problem to the design & optimization of $f$.  

Finn & Levine. Meta-learning and Universality: Deep Representation… ICLR 2018
Example: Few-Shot Classification

Given 1 example of 5 classes:

Classify new examples

training data $D_{\text{train}}$

test set $X_{\text{test}}$

meta-training

$\mathcal{T}_1$

$\mathcal{T}_2$

training classes

diagram adapted from Ravi & Larochelle ’17
Meta-RL Example: Maze Navigation

Given a small amount of experience
Learn to solve the task

By learning how to learn many other tasks:

...
The Meta Reinforcement Learning Problem

Reinforcement Learning:
- Inputs: \( x, S_t \)
- Outputs: \( y, a_t \)
- Data: \( \{(x, y)_i\} \)

\[ y = f(x; \theta) \]
\[ a_t = \pi(s_t; \theta) \]

Meta Reinforcement Learning:
- Inputs: \( D_{\text{train}} \)
- Outputs: \( a_t \)
- Data: \( \{D_i\} \)

\[ a_t = f(D_{\text{train}}, s_t; \theta) \]

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

Finn. Learning to Learn with Gradients. PhD Thesis 2018
Given a small amount of experience

Learn to solve the task

By learning how to learn many other tasks:

\[ \mathcal{D}_{\text{train}} \]

\[ s_t \rightarrow a_t \]

\[ \{ \mathcal{D}_i \} \sim \{ \mathcal{T}_i \} \]

meta-training tasks

diagram adapted from Duan et al. ‘17
The Meta Reinforcement Learning Problem

Meta Reinforcement Learning:

**Episodic Variant**

Inputs: $D_{train}$

$k$ rollouts from $\pi$

$s_t$  \[ a_t = f(D_{train}, s_t; \theta) \]

**Online Variant**

Inputs: $D_{train}$

$1 \ldots k$ timesteps from $\pi$

$s_t$  \[ a_t = f(D_{train}, s_t; \theta) \]
Outline

Meta-RL Problem Formulation & Examples

Method Classes: Recurrent Models, Gradient-Based Models

Challenges & Latest Developments
Design of $f$?

Recurrent network
(LSTM, NTM, Conv)

$a_t = f(D_{\text{train}}, s_t; \theta)$


Difference compared to using a recurrent policy?
Hidden state maintained across episodes within a task!
Case Study: Simple Neural Attentive Meta-Learner
Mishra et al, ICLR ’18

A SIMPLE NEURAL ATTENTIVE META-LEARNER

Nikhil Mishra *† Mostafa Rohaninejad* Xi Chen† Pieter Abbeel†
UC Berkeley, Department of Electrical Engineering and Computer Science
Embodied Intelligence
{nmishra, rohaninejadm, c.xi, pabbeel}@berkeley.edu
Simple Neural Attentive Meta-Learner

Mishra et al, ICLR ‘18

interleave 1D convolutions and attention
Simple Neural Attentive Meta-Learner
Mishra et al, ICLR ‘18

**Experiment**: Learning to visually navigate a maze
- train on 1000 small mazes
- test on held-out small mazes and large mazes
Simple Neural Attentive Meta-Learner
Mishra et al, ICLR ’18

**Experiment:** Learning to visually navigate a maze
- train on 1000 small mazes
- test on held-out small mazes and large mazes

<table>
<thead>
<tr>
<th>Method</th>
<th>Small Maze</th>
<th>Large Maze</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Episode 1</td>
<td>Episode 2</td>
</tr>
<tr>
<td>Random</td>
<td>188.6 ± 3.5</td>
<td>187.7 ± 3.5</td>
</tr>
<tr>
<td>LSTM</td>
<td>52.4 ± 1.3</td>
<td>39.1 ± 0.9</td>
</tr>
<tr>
<td>SNAIL (ours)</td>
<td><strong>50.3 ± 0.3</strong></td>
<td><strong>34.8 ± 0.2</strong></td>
</tr>
</tbody>
</table>

Table 5: Average time to find the goal on each episode
Digression: Connection to Contextual Policies

contextual policy: $\pi_\theta(a|s, \omega)$

$\omega$: stack location
$\omega$: walking direction

Contextual policy with experience as context.

What about goal-conditioned policies / value functions?

- rewards are a strict generalization of goals
- meta-RL objective is to adapt new tasks vs. generalize to new goals
  (k-shot vs. 0-shot)
Design of $f$?

Recurrent network (LSTM, NTM, Conv)

$y_{test} = f(D_{train}, x_{test}; \theta)$


+ general & expressive
+ a variety of design choices in architecture

-- complex model for complex task of learning
-- impractical data requirements
**Key idea:** Train over many tasks, to learn parameter vector $\theta$ that transfers

\[
\phi_j \leftarrow \theta - \alpha \nabla_{\theta} L_{\text{train}}^j(\theta)
\]

**Fine-tuning**
\[
\text{[test-time]}
\]

**Our method**
\[
\min_{\theta} \sum_{\text{task } i} L_{\text{test}}^i \left( \theta - \alpha \nabla_{\theta} L_{\text{train}}^i(\theta) \right)
\]

Can we learn a representation under which RL is fast?

Fast Adaptation in Reinforcement Learning

Fast Adaptation in Reinforcement Learning

**Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.** ICML 2017

Finn, Abbeel, Levine.
Design of $f$?

**Recurrence network**

$y_{\text{test}} = f(D_{\text{train}}, x_{\text{test}}; \theta)$

network implements the "learned learning procedure"

**Does it converge?**
- Sort of?

**What does it converge to?**
- Who knows...

**What to do if not good enough?**
- Nothing

---

**MAML**

$y_{\text{test}} = f(x_{\text{test}}; \theta - \alpha \nabla_{\theta} L(D_{\text{train}}))$

Does it converge?
- Yes (it’s gradient descent...)

What does it converge to?
- A local optimum (it’s gradient descent...)

What to do if not good enough?
- Keep taking gradient steps (it’s gradient descent..)

Does this structure come at a cost?
Outline

Meta-RL Problem Formulation & Examples
Method Classes: Recurrent Models, Gradient-Based Models
Challenges & Latest Developments
Wishlist in a Meta-RL algorithm

<table>
<thead>
<tr>
<th></th>
<th>(RL^2, \text{SNAIL})</th>
<th>MAML-PG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistent</td>
<td>(\times)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>Expressive</td>
<td>(\checkmark)</td>
<td>(\times)</td>
</tr>
<tr>
<td>Structured Exploration*</td>
<td>(\sim)</td>
<td>(\sim)</td>
</tr>
<tr>
<td>Efficient &amp; off-policy</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
</tbody>
</table>

*Initial work in this direction: Gupta et al. NIPS ’18, Stadie et al. NIPS ’18
Model-based RL already achieves zero-shot adaptation to new rewards.

**Goal:** learn to adapt model quickly to new environments (online system ID)
Goal: learn to adapt model quickly to new environments

motor malfunction

gradual terrain change

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments through Meta-RL
Goal: learn to adapt model quickly to new environments

tasks are temporal slices of experience

The task distribution is more-or-less free!

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments through Meta-RL
Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments through Meta-RL

Meta-learning using either:
- Dynamics RNN (ReBAL)
- MAML (GrBAL)

ONLINE ADAPTATION

Store recent history
\( \{ s_{t-M:t}, a_{t-M:t} \} \)

Take action \( a_t \) via MPC

\( (a_t, s_{t+1}) \)

Model Adaptation

\( \theta' = u_{\psi}(s_{t-M:t}, a_{t-M:t}, \theta_*) \)

Meta-train a prior \( \theta_* \)
Roach Robot

Meta-train on variable terrains  

Meta-test with slope, missing leg, payload, calibration errors

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments through Meta-RL
Roach Robot

Meta-train on variable terrains  Meta-test with slope, missing leg, payload, calibration errors

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments through Meta-RL
Roach Robot

Meta-train on variable terrains  Meta-test with slope, **missing leg**, payload, calibration errors

GrBAL  Model-Based RL (no adaptation)

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments through Meta-RL
# Wishlist in a Meta-RL algorithm

<table>
<thead>
<tr>
<th>Feature</th>
<th>RL^2, SNAIL</th>
<th>MAML-PG</th>
<th>ReBAL, GrBAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistent</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Expressive</td>
<td>✓</td>
<td>✗</td>
<td>~</td>
</tr>
<tr>
<td>Structured Exploration</td>
<td>~</td>
<td>~</td>
<td>✗</td>
</tr>
<tr>
<td>Efficient &amp; off-policy</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Still *lots* of room for exploration & improvement!

One more important challenge: where do the tasks come from?

So far: manual defined distribution of tasks. (and corresponding rewards)
Unsupervised Meta-RL

General Recipe


reward function


Abhishek Gupta  Ben Eysenbach  Sergey Levine
Random Task Proposals

- Use randomly initialize discriminators for reward functions

\[ R(s, z) = \log p_D(z | s) \]

\( D \rightarrow \) randomly initialized network

- Important: Random functions over state space, **not** random policies

Random policy – exponential
Random reward – polynomial
Diversity-Driven Proposals

Environment

Action  \rightarrow  State

Policy(Agent)

Skill (z)  \rightarrow  Predict Skill

Discriminator(D)

\begin{align*}
R(s, z) &= \log p_D(z|s)
\end{align*}

- Policy $\rightarrow$ visit states which are discriminable
- Discriminator $\rightarrow$ predict skill from state

Task Reward for UML: $R(s, z) = \log p_D(z|s)$

Eysenbach, Gupta, Ibarz, Levine. Diversity is All You Need.
Examples of Acquired Tasks

Eysenbach, Gupta, Ibarz, Levine. Diversity is All You Need.
Does it work?

Meta-test performance with rewards

**Takeaway:** Relatively **simple** mechanisms for proposing tasks work surprisingly well.

Key References

**Recurrent Models:** Yan Duan et al. ’17 (RL²), Jane Wang et al. CogSci ’17 (Learning to Reinforcement Learn), Nikhil Mishra et al. ICLR ’18 (SNAIL)

**Model-Agnostic Meta-Learning:** Finn et al. ICML ’17

Further Reading

**Value-based:** Sung et al. arXiv ’17 (*Meta-critic networks*)

**Exploration:** Gupta et al. NIPS ’18, Stadie et al. NIPS ’18

**Unsupervised:** Gupta et al. arXiv ’18 (*Unsupervised Meta-Learning for Reinforcement Learning*)

**Model-based:** Nagabandi*, Clavera* et al. arXiv ’18

**Evolutionary Strategies:** Houthooft et al. NIPS ’18

**Cognitive Science:** Wang et al. Nature Neuroscience ’18 (*PFC as a Meta-RL System*)
Takeaway

Don’t reinforcement learn from scratch. Meta-RL provides an approach to do this.

Open Problems

- a meta-RL algorithm that is both consistent and expressive
- meta-RL algorithms that learn to explore in a structured way
- constructing task distributions for meta-RL in an automated way
- meta-RL algorithms that can incorporate off-policy data effectively

Collaborators