Meta Reinforcement Learning
Adaptable Models & Policies

CS 224R
Reminders

Homework 3 due **Wednesday**

Project milestone due **next Wednesday**
Plan for Today

Meta-RL problem statement
Black-box meta-RL methods <- comes up in HW4
Optimization-based meta-RL methods

Next time: Learning to explore. <- part of HW4

Lecture goals:
- Understand the meta-RL problem statement & set-up
- Understand the basics of black-box meta RL algorithms
- Understand the basics & challenges of optimization-based meta RL algorithms
Problem Settings

Multi-Task Learning
Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning
Solve target task $\mathcal{T}_b$ after solving source task $\mathcal{T}_a$ by \textit{transferring} knowledge learned from $\mathcal{T}_a$

The Meta-Learning Problem
Given data from $\mathcal{T}_1, \cdots, \mathcal{T}_n$, quickly solve new task $\mathcal{T}_{\text{test}}$

\textbf{In all settings}: tasks must share structure.

A reinforcement learning task:

$$\mathcal{T}_i \triangleq \{ S_i, A_i, p_i(s_1), p_i(s'|s, a), r_i(s, a) \}$$

Meta-reinforcement learning = meta-learning with RL tasks
The Meta-Learning Problem

Supervised Learning:

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

$\mathbf{y} = g_{\phi}(\mathbf{x})$

Meta Supervised Learning:

Inputs: $\mathcal{D}^{tr}, \mathbf{x}^{ts}$  
Outputs: $\mathbf{y}^{ts}$  
Data: $\{\mathcal{D}_i\}$

$\mathbf{y}^{ts} = f_{\theta}(\mathcal{D}^{tr}, \mathbf{x}^{ts})$

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

The Meta Reinforcement Learning Problem

Reinforcement Learning:

Inputs: $x, S_t$  
Outputs: $y, a_t$

$y = g_\phi(x)$  
$a_t = \pi(s_t; \theta)$

Data: $\{(x, y)_i\}$  
$\{(s_t, a_t, r_t, s_{t+1})\}$

Meta Reinforcement Learning:

Inputs: $D^{tr}, S_t$  
Outputs: $a_t$

$k$ rollouts from $\pi$

$a_t = f_\theta(D^{tr}, s_t)$

Data: $\{D_i\}$

dataset of datasets collected for each task

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

Meta-RL Example: Maze Navigation

**Goal:**

Collect small amount of experience in new MDP

\[ D_{tr} \sim \pi^{exp} \]

Learn policy that solves that MDP

\[ D_{tr} \rightarrow \pi^{task} \]

diagram adapted from Duan et al. ‘17
Meta-RL Example: Maze Navigation

**Meta-Train Time:**
Learn how to efficiently explore & solve many MDPs:

```
Meta-train $\pi_{\text{exp}}, \pi_{\text{task}}$ ...
```

**Meta-Test Time:**
Collect small amount of experience in new MDP

```
Collect $\mathcal{D}_{\text{tr}} \sim \pi_{\text{exp}}$
```

Learn policy that solves that MDP

```
$\mathcal{D}_{\text{tr}} \rightarrow \pi_{\text{task}}$
```

**Key assumption:** Meta-training & meta-testing MDPs come from same distribution.

(so that we can expect generalization)

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

Meta Reinforcement Learning:

**Episodic Variant**

Inputs: \( \mathcal{D}_{\text{train}} \) \( S_t \)  
\( \pi \) \( a_t \) = \( f_\theta(\mathcal{D}^{\text{tr}}, s_t) \)

**Online Variant**

Inputs: \( \mathcal{D}_{\text{train}} \) \( S_t \)  
\( \pi \) \( a_t \) = \( f_\theta(\mathcal{D}^{\text{tr}}, s_t) \)

Note: exploration policy \( \pi \) and adaptation policy \( f_\theta \) need not be the same.
Plan for Today

Meta-RL problem statement

**Black-box meta-RL methods**

Optimization-based meta-RL methods
Black-Box Meta-RL: Overview

Black-box network
(LSTM, NTM, Conv, …)

\[ a_t = f_\theta(D^{tr}, s_t) \]

**Question:** Why don’t we need to pass in the actions \( a_{t-1} \) with the support set?

**Question:** How is this different from simply doing RL with a recurrent policy?

- Reward is passed as input (& trained across multiple MDPs)
- Hidden state maintained
  - across episodes within a task!

\( D^{tr} \) training/support set
gets larger over time
Black-Box Meta-RL: Algorithm

1. Sample task $\mathcal{T}_i$
2. Roll-out policy $\pi(a \mid s, \mathcal{D}_i^{tr})$ for $N$ episodes (under dynamics $p_i(s' \mid s, a)$ and reward $r_i(s, a)$)
3. Store sequence in replay buffer for task $\mathcal{T}_i$.
4. Update policy to maximize discounted return for all tasks.
Black-Box Meta-RL: Algorithm

Meta-Training

1. Sample task $\mathcal{T}_i$
2. Roll-out policy $\pi(a | s, \mathcal{D}^\text{tr}_i)$ for $N$ episodes (under dynamics $p_i(s'|s,a)$ and reward $r_i(s,a)$)
3. Store sequence in replay buffer for task $\mathcal{T}_i$.
4. Update policy to maximize discounted return for all tasks.

Meta-Test Time

1. Sample new task $\mathcal{T}_j$
2. Roll-out policy $\pi(a | s, \mathcal{D}^\text{tr}_j)$ for up to $N$ episodes
RNN architecture

TRPO/A3C (on-policy)


Attention + 1D conv

TRPO (on-policy)

Mishra, Rohaninejad, Chen, Abbeel. *A Simple Neural Attentive Meta-Learner*. ICLR 2018

Feedforward + average

SAC (off-policy)

Meta-RL Example #1

From: Mishra, Rohaninejad, Chen, Abbeel. *A Simple Neural Attentive Meta-Learner*. ICLR 2018

**Experiment:** Learning to visually navigate a maze
- train on 1000 small mazes
- test on held-out small mazes and large mazes
Meta-RL Example #1

From: Mishra, Rohaninejad, Chen, Abbeel. *A Simple Neural Attentive Meta-Learner*. ICLR 2018

**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></th>
<th>Large Maze</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Episode 1</td>
<td>Episode 2</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>
<td>420.2 ± 1.2</td>
<td>420.8 ± 1.2</td>
</tr>
<tr>
<td>LSTM</td>
<td>52.4 ± 1.3</td>
<td>39.1 ± 0.9</td>
<td>180.1 ± 6.0</td>
<td>150.6 ± 5.9</td>
</tr>
<tr>
<td>SNAIL (ours)</td>
<td>50.3 ± 0.3</td>
<td>34.8 ± 0.2</td>
<td>140.5 ± 4.2</td>
<td>105.9 ± 2.4</td>
</tr>
</tbody>
</table>

Table 5: Average time to find the goal on each episode
**Meta-RL Example #2**


**Experiment:** Continuous control problems

- different directions, velocities
- different physical dynamics

Meta-RL algos are very efficient at new tasks.

But, what about meta-training efficiency?

**Question:** Do you expect off-policy meta-RL to be more or less efficient than on-policy meta-RL?
Digression: Connection to Multi-Task Policies

Multi-task policy: $\pi_\theta(a \mid s, z_i)$

$z_i$: stack location  
$z_i$: walking direction

Multi-task policy with experience as task identifier.

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)
Black-Box Meta-RL Summary

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

**Black-box network**
(LSTM, NTM, Conv, …)

- general & expressive
- a variety of design choices in architecture
- hard to optimize
- inherits sample efficiency from outer RL optimizer
Plan for Today

Meta-RL problem statement
Black-box meta-RL methods

Optimization-based meta-RL methods
Optimization-Based Meta-Learning

Key idea: embed optimization inside the inner learning process
Fine-tuning

Universal Language Model Fine-Tuning for Text Classification. Howard, Ruder. ‘18

\[ \phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D^{tr}) \]

Fine-tuning less effective with very small datasets.
**Optimization-Based Meta-Learning**

**Fine-tuning** [test-time]

\[
\phi \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, D_{tr}^{tr})
\]

**Meta-learning**

\[
\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, D_{i}^{tr}), D_{i}^{ts})
\]

**Key idea:** Over many tasks, learn parameter vector \( \theta \) that transfers via fine-tuning

---

Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017
Optimization-Based Meta-Learning

$$\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_{i}^{tr}), \mathcal{D}_{i}^{ts})$$

- $\theta$: parameter vector being meta-learned
- $\phi_i^*$: optimal parameter vector for task $i$

Model-Agnostic Meta-Learning

Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017
Optimization-Based Meta-Learning Meta-RL

Key idea: embed optimization inside the inner learning process

Question: What should we use for the inner optimization and why?

Policy gradients?
- + gradient-based!
- + on-policy (inefficient)
- - low information (esp w/ sparse rewards)

Q-learning?
- - dynamic programming (requires many steps)

Model-based RL?
- + gradient-based (model learning=supervised)
- + off-policy (data efficient)
MAML with Policy Gradients

\[
\text{MAML: } \min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_{i}^{\text{tr}}, \mathcal{D}_{i}^{\text{ts}}))
\]

\[
\text{Policy Gradient: } \nabla_{\theta} J_{i}(\theta) = E_{\tau \sim \pi_{\theta}, \mathcal{T}_{i}} \left[ \left( \sum_{t} \nabla_{\theta} \log \pi_{\theta}(a_{t}|s_{t}) \right) \left( \sum_{t} r_{i}(s_{t}, a_{t}) \right) \right]
\]

**Meta-Training**

1. Sample task \( \mathcal{F}_{i} \)
2. Collect \( \mathcal{D}_{i}^{\text{tr}} \) by rolling out \( \pi_{\theta} \)
3. Inner loop adaptation: \( \phi_{i} = \theta + \alpha \nabla_{\theta} J_{i}(\theta) \)
4. Collect \( \mathcal{D}_{i}^{\text{ts}} \) by rolling out \( \pi_{\phi_{i}} \)
5. Outer loop update: \( \theta \leftarrow \theta + \beta \sum_{\text{task } i} \nabla_{\theta} J_{i}(\phi_{i}) \)

**Meta-Test Time**

1. Sample new task \( \mathcal{F}_{j} \)
2. Collect \( \mathcal{D}_{j}^{\text{tr}} \) by rolling out \( \pi_{\theta} \)
3. Adapt policy:
   \[
   \phi_{j} = \theta + \alpha \nabla_{\theta} J_{j}(\theta)
   \]
MAML with Policy Gradients

MAML with Policy Gradients

Meta-test time:
1. Adapt model $f_\theta \rightarrow f_{\phi_t}$ to last $k$ time steps
2. Plan $a_t, \ldots, a_{t+h}$ using adapted model $f_{\phi_t}$

Meta-training:
- Tasks: windows in time
- $D_{ts}^t$: $s_{t:t+h}, a_{t:t+h}$
- $D_{tr}^t$: $s_{t-k:t}, a_{t-k:t}$

Dynamic Environments without Adaptation

Model-Based RL Only

Tries to fit single model $f(s' \mid s, a)$ to varying $p_t(s' \mid s, a)$.
Dynamic Environments without Adaptation
MAML+Model-based RL

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR’19
VelociRoACH 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 Environments through Meta-RL. ICLR ’19
VelociRoACH Robot

Meta-train on variable terrains

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

model-based RL (no adaptation)

with MAML (ours)

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR’19
Black-Box Meta-RL

+ general & expressive
+ a variety of design choices in architecture & objective
-- hard to optimize

Optimization-Based Meta-RL

+ inductive bias of optimization built in
+ easy to combine with policy gradients, model-based methods
-- policy gradients very noisy
-- hard to combine with value-based RL methods

Both: inherit sample efficiency from outer RL optimizer
Plan for Today

Meta-RL problem statement
Black-box meta-RL methods
Optimization-based meta-RL methods

Lecture goals:
- Understand the meta-RL problem statement & set-up
- Understand the basics of black-box meta RL algorithms
- Understand the basics & challenges of optimization-based meta RL algorithms
Next time

Reminders

Today: meta-RL basics

Wednesday: learning to explore via meta-RL

Homework 3 due Wednesday

Project milestone due next Wednesday