Meta Reinforcement Learning
Adaptable Models & Policies

CS 330
Reminders

Homework 3 out, due **Monday 10/26**.

Project milestone due **Monday 11/2**.
Mid-Quarter Check In

How are you doing?

https://pollev.com/chelseafinn494
Thank you!
Mid-Quarter Survey

Some of the results

How is the difficulty of the class?
47 responses

<table>
<thead>
<tr>
<th>Level</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2.1%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.3%</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>48.9%</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>29.8%</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>14.9%</td>
</tr>
</tbody>
</table>
Mid-Quarter Survey

Some of the results

How do you feel about the lecture speed?

47 responses

<table>
<thead>
<tr>
<th>Speed</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Too slow 1</td>
<td>2</td>
<td>4.3%</td>
</tr>
<tr>
<td>Too slow 2</td>
<td>5</td>
<td>10.6%</td>
</tr>
<tr>
<td>Normal 3</td>
<td>23</td>
<td>48.9%</td>
</tr>
<tr>
<td>Too fast 4</td>
<td>13</td>
<td>27.7%</td>
</tr>
<tr>
<td>Too fast 5</td>
<td>4</td>
<td>8.5%</td>
</tr>
</tbody>
</table>
Mid-Quarter Survey

Some of the comments mentioned multiple times.

Lecture feedback:
- Additional sections on background RL, variational inference concepts
- More concrete examples / instantiations of algorithms
- Reduce some gaps between lectures and assignments

Assignment feedback:
- Provide non-colab option
- Cleaner code with more comments
- PyTorch instead of TF
- Reduce # of iterations the algorithms need to be run

Office hours feedback:
- More office hours
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. <- focus 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
Recap: The anatomy of a reinforcement learning algorithm

Last lectures: introduced model-free, model-based RL methods

- generate samples (i.e. run the policy)
- fit a model to estimate return:
  - compute \( \hat{Q} = \sum_{t' = t}^{T} \gamma^{t' - t} r_{t'} \) (MC policy gradient)
  - fit \( Q_{\phi}(s, a) \) (actor-critic, Q-learning)
  - estimate \( p(s'|s, a) \) (model-based)
- improve the policy:
  - \( \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \) (policy gradient)
  - \( \pi(s) = \arg \max Q_{\phi}(s, a) \) (Q-learning)
  - optimize \( \pi_{\theta}(a|s) \) (model-based)
Recall: 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 **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 $\mathcal{T}_{\text{test}}$

**In all settings**: tasks must share structure.

A reinforcement learning **task**:

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

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

**Supervised Learning:**

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

\( \mathbf{y} = f(\mathbf{x}; \theta) \)

**Meta Supervised Learning:**

Inputs: \( \mathcal{D}_{\text{train}} \), \( \mathbf{x}_{\text{test}} \)  
Outputs: \( \mathbf{y}_{\text{test}} \)  
Data: \( \{\mathcal{D}_i\} \)

\( \mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta) \)

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

The Meta Reinforcement Learning Problem

Reinforcement Learning:
Inputs: $x, S_t$  
Outputs: $y, a_t$  
$y = f(x; \theta)$  
$a_t = \pi(s_t; \theta)$  

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

Meta Reinforcement Learning:
Inputs: $D_{train}, S_t$  
Outputs: $a_t$  
$a_t = f(D_{train}, s_t; \theta)$  

Data: $\{D_i\}$  
(dataset of datasets collected for each task)

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

Given a small amount of experience

Learn to solve the task

\( D_{\text{train}} \)

\( s_t \rightarrow a_t \)

Meta-RL Example: Maze Navigation

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

By learning how to learn many other tasks:

\[ \mathcal{T}_1 \] \hspace{0.5cm} \mathcal{T}_2 \hspace{0.5cm} \ldots \hspace{0.5cm} \text{meta-training tasks}

Given a small amount of experience

Learn to solve the task

\[ D_{\text{train}} \]

\[ S_t \rightarrow a_t \]

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

Meta Reinforcement Learning:

**Inputs:** $D_{train} \quad s_t \quad k$ rollouts from $\pi$

**Outputs:** $a_t

**Episodic Variant**

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

**Online Variant**

$$a_t = f(D_{train}, s_t; \theta)$$
Plan for Today

Meta-RL problem statement

Black-box meta-RL methods

Optimization-based meta-RL methods
Black-box meta-RL

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

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

**Question:** How is this different from simply using a recurrent policy?
(answer in chat or by raising your hand)

Reward is passed as input (& trained across multiple MDPs)

Hidden state maintained across episodes within a task!
1. Sample task $\mathcal{T}_i$
2. Roll-out policy $\pi(a|s, \mathcal{D}_{tr})$ 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.
RNN architecture  TRPO/A3C (on-policy)

Feedforward + average  SAC (off-policy)

Attention + 1D conv  TRPO (on-policy)


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

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>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>50.3 ± 0.3</td>
<td>34.8 ± 0.2</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.

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

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

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

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

- 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**
Recap: **Optimization-Based Meta-Learning**

Key idea: embed optimization inside the inner learning process
Recap: **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?  
(in chat or by raising hand)

<table>
<thead>
<tr>
<th></th>
<th>Policy gradients?</th>
<th>Q-learning?</th>
<th>Model-based RL?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+ gradient-based!</td>
<td>- dynamic programming (requires many steps)</td>
<td>+ gradient-based (model learning=supervised)</td>
</tr>
<tr>
<td></td>
<td>- on-policy (inefficient)</td>
<td>+ off-policy (efficient)</td>
<td>+ off-policy (efficient)</td>
</tr>
</tbody>
</table>
MAML + Policy Gradients

MAML + Policy Gradients

MAML + Model-Based RL

**Online Variant**

\[ \mathcal{D}_{\text{train}} \xrightarrow{\pi} S_t \xrightarrow{k \text{ timesteps from } \pi} \mathcal{D}_{ts} \]

**Inputs:** \( \mathcal{D}_{\text{train}} \), \( S_t \)

**Outputs:** \( a_t \)

**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:**

\( \mathcal{D}_{tr}^t \)

\( s_{t-k:t}, a_{t-k:t} \)

tasks: windows in time

\( \mathcal{D}_{ts}^t \)

\( s_{t:t+h}, a_{t:t+h} \)

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR '19
Dynamic Environments without Adaptation

Model-Based RL Only

Tries to fit single model $f(s' | s, a)$ to varying $p_t(s' | s, a)$. 

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Real-World Environments. ICLR '19
Dynamic Environments without Adaptation

MAML+Model-based RL
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 Real-World Environments. ICLR ’19
Meta-train on variable terrains  Meta-test with slope, missing leg, payload, calibration errors

VelociRoACH Robot

model-based RL (no adaptation)  with MAML (ours)

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

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

Next time: Learning to explore.

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
Reminders

Today: meta-RL basics
Next Monday: learning to explore via meta-RL
Next Wednesday: Bayesian perspective on meta-RL

Homework 3 due Monday 10/26.
Project milestone due Monday 11/2.