Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

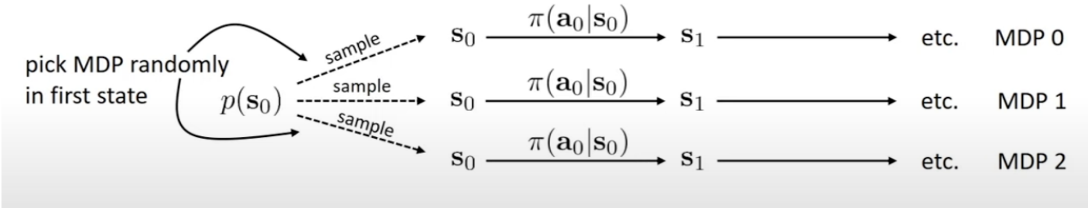
Lecture 28 Summary

Summarized By: Benyamin Naderi

Multi-Task Reinforcement Learning

Multi-task RL can be formulated as single-task RL in a joint MDP framework:

- **Approach:**
  - Randomly sample an MDP at the start of each episode
  - Begin from the initial state distribution $p(s_0)$
  - Execute policy $\pi(a_0|s_0)$ across all tasks



The policy learns to handle multiple tasks by treating them as different initial state distributions within a unified MDP.

Contextual Policies

- **Standard Policy:**

$$\pi_0(\mathbf{a}|\mathbf{s})$$

- **Contextual Policy:**

$$\pi_0(\mathbf{a}|\mathbf{s}, \omega)$$

where ω represents task context (e.g., "do dishes" vs. "laundry")

- **Formal Representation:**
  - Augmented state space:

$$\hat{\mathbf{s}} = \begin{bmatrix} \mathbf{s} \\ \omega \end{bmatrix}, \quad \hat{\mathcal{S}} = \mathcal{S} \times \Omega$$

  - Example contexts (ω):
    - Stack location in robotic manipulation
    - Walking direction for navigation
    - Target position for hockey puck hitting

Goal-Conditioned Policies

- **Policy Definition:**

$$\pi_\theta(a|s, g)$$

- **Reward Specifications:**

- Exact goal achievement:

$$r(s, a, g) = \delta(s = g)$$

- Approximate goal region (ε-tolerance):

$$r(s, a, g) = \delta(\|s - g\| \leq \varepsilon)$$

- **Advantages:**
  - Eliminates manual reward engineering for each task
  - Enables zero-shot transfer to novel goals
- **Challenges:**
  - Training difficulties in practice
  - Limited to goal-reaching tasks

A comparison on Learning paradigm

- **Standard Learning:**

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta, D^{tr})$$

(Single task, single dataset)

- **Meta-Learning:**

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}(\underbrace{f_\theta(D_i^{tr})}_{\phi_i}, D_i^{ts})$$

(Learn adaptation procedure)

- **Standard RL:**

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

(Single MDP)

- **Meta-RL:**

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} \mathbb{E}_{\pi_{\underbrace{f_\theta(\mathcal{M}_i)}_{\phi_i}}}[R(\tau)]$$

(Learn MDP adaptation)

- Meta methods optimize adaptation capability (f_θ) rather than direct solutions
- Sum over tasks appears in the outer optimization loop
- Task-specific parameters (ϕ_i) are generated on-demand via f_θ

Meta-Reinforcement Learning

- **Objective:**

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

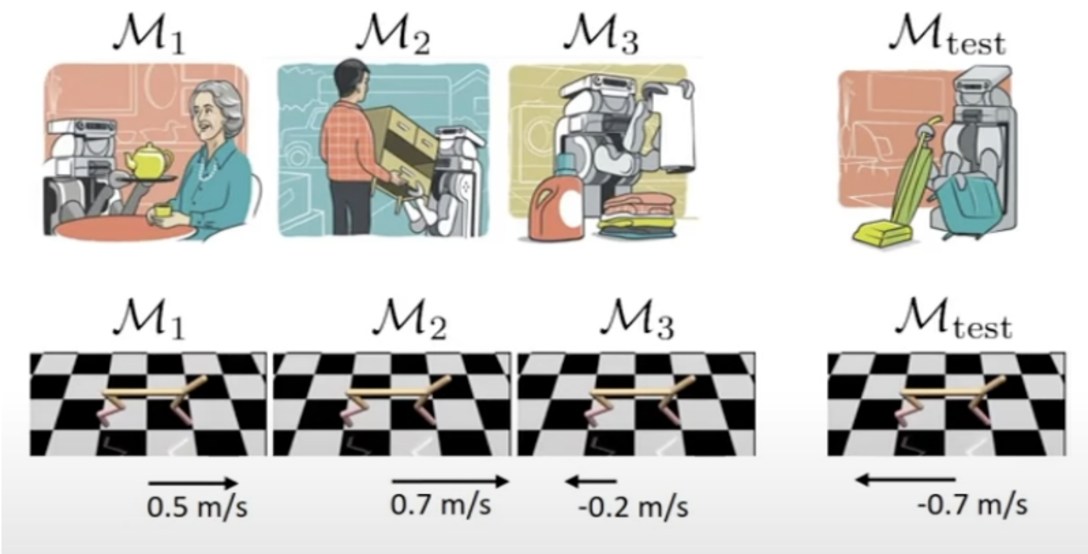
where $\phi_i = f_\theta(\mathcal{M}_i)$ (task-specific policy parameters)

- **Assumption:**

$$\mathcal{M}_i \sim p(\mathcal{M})$$

(Tasks drawn from some distribution)

- **Meta-Testing:**
  - Sample new MDP $\mathcal{M}_{\text{test}} \sim p(\mathcal{M})$
  - Adapt policy: $\phi_{\text{test}} = f_\theta(\mathcal{M}_{\text{test}})$
- **Example (Velocity Adaptation):**



Contextual Policies & Meta-Learning

- **Meta-Learning Objective:**

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

where $\phi_i = f_\theta(\mathcal{M}_i)$ learns task-specific parameters

- **Contextual Policy:**

$$\pi_\theta(a_t | s_t, s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1})$$

  - Infers latent context (z_t or ϕ_i) from interaction history
  - Key difference: Meta-RL infers context, multi-task RL receives it

- **Equivalence:**
  - ϕ_i (task parameters) $\approx z_t$ (latent context)
  - Both capture task essence
- Meta-learning's f_θ automates what multi-task RL manually specifies
- Contextual policies generalize across tasks via history conditioning
- Practical implementations often use RNNs/LSTMs to encode history

Meta-RL with Recurrent Policies

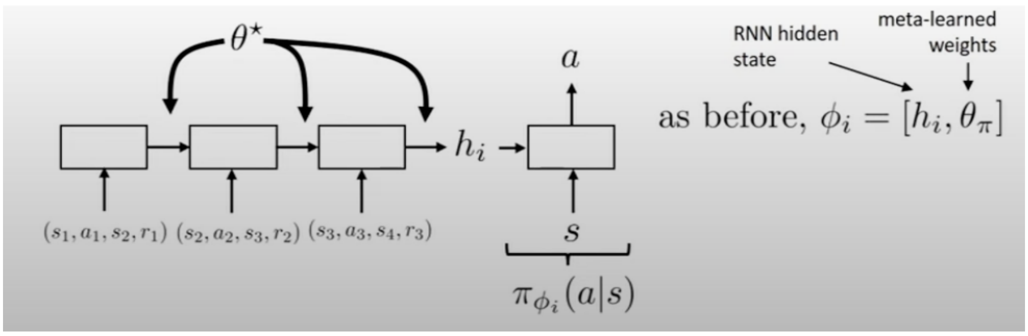
- **Objective:**

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

  where $\phi_i = f_\theta(\mathcal{M}_i)$ encodes task-specific adaptation
  - RNN processes trajectory history:

$$\tau_t = (s_1, a_1, r_1, ..., s_t)$$

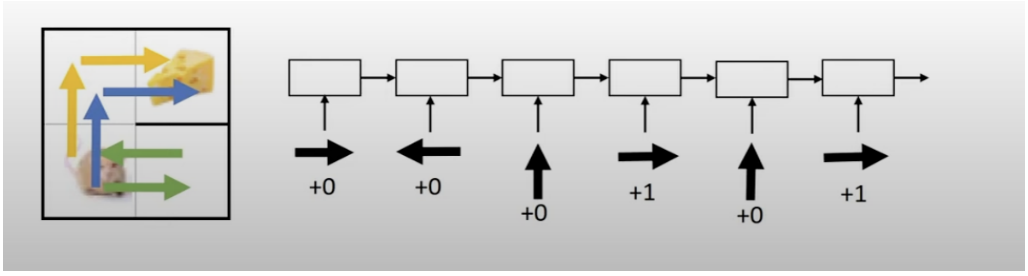
  - Hidden state h_t captures task information $\mathcal{M}_i$
  - Policy becomes $\pi_{\phi_i}(a|s) = \pi_\theta(a|s, h_t)$
- **Implementation Challenges:**
  - How to design $f_\theta(\mathcal{M}_i)$:
    1. Learn from experience $\{(s_k, a_k, s_{k+1}, r_k)\}_{k=1}^T$
    2. Control exploration during adaptation (unique to RL)



- **Architecture**
  - $\phi_i = [h_i, \theta_\pi]$ combines:
    * RNN hidden state h_i (task memory)
    * Meta-learned weights θ_π (shared base policy)
  - Action sampling: $a_t \sim \pi_\theta(a_t|s_t, h_t)$

Input	Process	Output
(s_t, a_t, r_t, s_{t+1})	RNN update	h_{t+1}
s_t, h_t	Policy network	a_t

RNN Policy Implementation



- Directly train a recurrent policy $\pi_{\phi_i}(a|s)$ that maintains memory across episodes
- Hidden state persists between episodes to retain task information
- crucially , RNN hidden state is not reset between episodes.

Meta-RL as Optimization

- **Optimization**

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

  where $\phi_i = f_\theta(\mathcal{M}_i)$ is the adaptation rule

- **When f_θ is RL Itself:**

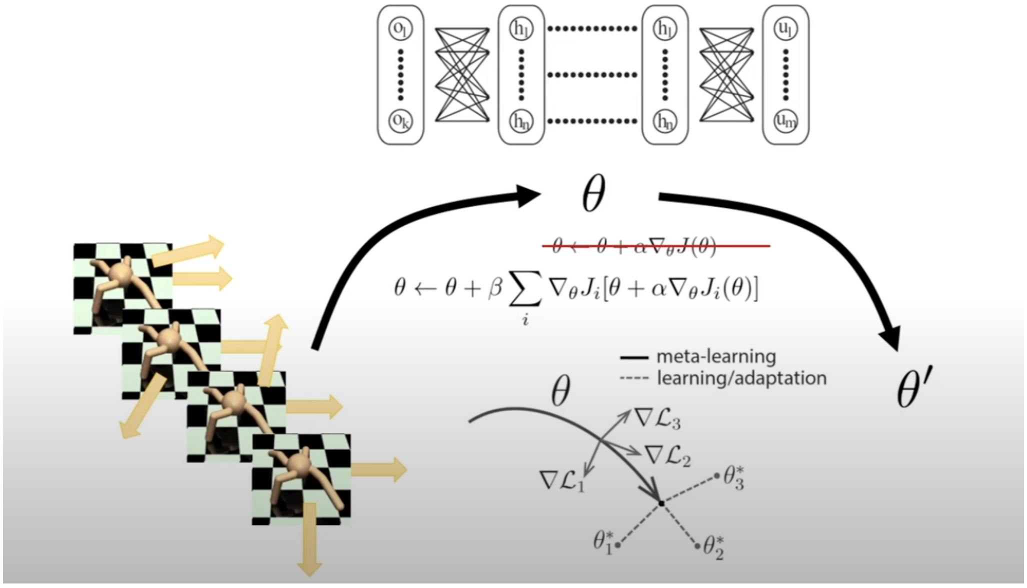
$$f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta)$$

  - Requires policy rollouts in $\mathcal{M}_i$ to estimate $\nabla_\theta \mathbb{E}[R(\tau)]$
  - Adaptation data: $\{(s_k, a_k, s_{k+1}, r_k)\}_{k=1}^{T}$

- **Connection to MAML:**
  - Outer loop: Meta-objective over tasks
  - Inner loop: Policy gradient updates

$$\theta^{k+1} \leftarrow \theta^k + \alpha \nabla_\theta J(\theta^k)$$

Standard RL	**Meta-RL**
$\theta^* = \arg\max_\theta \mathbb{E}[R(\tau)]$	$\theta^* = \arg\max_\theta \sum_i \mathbb{E}[R(\tau_i)]$
Single task optimization	Bi-level optimization



θ
$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
$\theta \leftarrow \theta + \beta \sum_i \nabla_\theta J_i[\theta + \alpha \nabla_\theta J_i(\theta)]$
meta-learning
learning/adaptation
θ'
$\nabla \mathcal{L}_1$
$\nabla \mathcal{L}_2$
$\nabla \mathcal{L}_3$
θ_1^*
θ_2^*
θ_3^*