Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Lecture 27 Summary

Summarized By: Benyamin Naderi

1 Meta-Learning and Transfer Learning

Knowledge Sharing in Sparse-Reward Environments : In games like *Montezuma's Revenge*, sparse rewards mean the agent lacks guidance from the environment. However, leveraging prior knowledge—like needing to get a key—can drastically improve performance. Curiosity-driven exploration is useful but resource-intensive. Understanding task structure significantly helps.

How is Knowledge Stored?

- **Q-function:** Indicates good actions or states; can be reused or fine-tuned.
- **Policy:** Guides decisions. Some actions are always irrelevant.
- **Models:** Capture environment dynamics (e.g., physics).
- **Features/hidden states:** Provide useful internal representations.

What is Transfer Learning?

Transfer learning means leveraging knowledge from previous tasks to learn faster or perform better on a new task.

- In RL, each task is modeled as an **MDP (Markov Decision Process)**.
- **Source domain:** Where the original task is learned.
- **Target domain:** Where the new task is applied.

Types of Transfer

1. **Forward Transfer**
   - Train on a source task, then apply to the target.
   - Works best when tasks are similar.
2. **Multi-task Transfer**
   - Train on multiple tasks jointly.
   - Representations are shared across all tasks.
3. **Meta-learning**
   - Learn how to quickly learn new tasks.
   - Assumes adaptation will happen during testing.

Transfer Performance by "Shot"

- **0-shot:** No retraining; apply the source policy as-is.
- **1-shot:** One trial in the target domain is allowed.
- **Few-shot:** Limited target trials before evaluation.

2 Challenges in Transfer Learning

When applying transfer learning in reinforcement learning, several key challenges may arise:

- **Domain Shift:** Representations learned in the source domain might not work well in the target domain due to differences in state distributions or dynamics.
- **Difference in the MDP:** Some actions or transitions possible in the source domain may not be feasible in the target domain, requiring policy adaptation.
- **Finetuning Issues:** When pretraining and then finetuning, the finetuning process may still need exploration, but the optimal policy during finetuning could become deterministic, limiting learning.

Approaches to Address Challenges

- **Domain Adaptation:** Techniques like adversarial training or representation matching can help align source and target domains.
- **Policy Regularization:** Constraining the policy during finetuning to prevent drastic deviations from the source policy.
- **Curriculum Learning:** Gradually increasing task difficulty to bridge the gap between source and target domains.

Domain Adaptation Approaches

Domain adaptation techniques aim to align the source and target domains by learning domain-invariant representations. A common approach in computer vision involves adversarial training:

- **Adversarial Domain Adaptation:**
  - Train a feature extractor to produce domain-invariant features
  - Simultaneously train a domain classifier $D_\phi(z)$ to predict the domain from features z
  - Use gradient reversal to make the features confusing to the domain classifier
- **Invariance Assumption:** The key assumption is that domain differences are irrelevant to the task. Formally:
  - While $p(x)$ differs between domains
  - There exists some representation $z = f(x)$ where:
    * $p(y|z) = p(y|x)$ (task-relevant information preserved)
    * $p(z)$ is the same across domains (domain-invariant)

Domain Adaptation in RL for Dynamics

In reinforcement learning, transferring a policy trained in a simulator to the real world is challenging due to differences in **dynamics**. Even if the observation space remains similar (i.e., invariant features are preserved), the policy may fail if the underlying environment behaves differently.

- In the **real world**, dynamics like obstacles or friction cause the agent to behave differently.
- In a **simulator**, simplified dynamics may lead to misleading trajectories and rewards.
- To bridge this gap, a **learned reward offset** $\Delta r(s, a)$ can be added, resulting in a corrected reward:

$$\tilde{r}(s, a) = r(s, a) + \Delta r(s, a)$$

- This offset is estimated using a classifier that distinguishes real vs. simulated transitions.

However, this method is not foolproof. It may **fail** when:

- The simulator's assumptions are too far from reality (e.g., dynamics that can't be compensated for).
- The agent learns policies that exploit simulator-specific shortcuts not present in the real environment.

To adapt from simulation to real-world dynamics, we adjust the reward using a learned offset:

$$\tilde{r}(s_t, a_t) = r(s_t, a_t) + \Delta r(s_t, a_t) \tag{1}$$

The reward offset $\Delta r(s_t, a_t, s_{t+1})$ can be estimated in two ways:

$$\Delta r(s_t, a_t, s_{t+1}) = \log p_{\text{target}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{source}}(s_{t+1} \mid s_t, a_t) \tag{2}$$

Alternatively, using a domain classifier:

$$\begin{aligned} \Delta r(s_t, a_t, s_{t+1}) = {} & \log p(\text{target} \mid s_t, a_t, s_{t+1}) - \log p(\text{target} \mid s_t, a_t) \\ & - \log p(\text{source} \mid s_t, a_t, s_{t+1}) + \log p(\text{source} \mid s_t, a_t) \end{aligned} \tag{3}$$

What if We Can Also Finetune?

1. **RL tasks are generally much less diverse**
   - Learned features tend to be *less general*.
   - Policies and value functions often become *overly specialized*, limiting their transferability.
2. **Optimal policies in fully observed MDPs are deterministic**
   - *Loss of exploration* occurs as training converges.
   - *Low-entropy policies* adapt very slowly to new environments or tasks.

How to Maximize Forward Transfer?

- **Basic intuition:** The more **varied** the training domain is, the more likely the learned policy is to generalize in a **zero-shot** setting to a slightly different domain.
- **Randomization:** Applying randomness in:
  - *Dynamics*
  - *Appearance*
  - *Environmental parameters*

  is widely used in **simulation-to-real** transfer, especially in robotics.