Hierarchical RL and Skill Discovery

CS 224R
Reminders

Today: Project milestone due

Wednesday next week: Homework 4 due
The Plan

Information-theoretic concepts

Skill discovery

Using discovered skills

Hierarchical RL

Key learning goals:

• Understand the concept of a skill and basic algorithms in this space
• Overview of hierarchical RL algorithms
Recall: RL so far

We knew what we wanted
Short-horizon behaviors
Well defined tasks/rewards
Why Skill Discovery?

What if we want to discover interesting behaviors?

[The construction of movement by the spinal cord, Tresch et al., 1999]

[Postural hand synergies for tool use, Santello, et al., 1998]
Why Skill Discovery? More practical version

Coming up with tasks is tricky...

Task ideas for a tabletop manipulation scenario

[Meta-World, Yu, Quillen, He, Julian, et al., 2019]
Why Hierarchical RL?

Performing tasks at various levels of abstractions

Bake a cheesecake
Buy ingredients
Go to the store
Walk to the door
Take a step
Contract muscle X

Exploration
The Plan

Information-theoretic concepts

Skill discovery

Using discovered skills

Hierarchical RL
Entropy

\[ p(x) \] distribution (e.g., over observations \( x \))

\[ H(p(x)) = -\mathbb{E}_{x \sim p(x)}[\log p(x)] \]

entropy – how “broad” \( p(x) \) is

Slide adapted from Sergey Levine
KL-divergence

**Distance between two distributions**

\[
\mathbb{D}_{KL}(q||p) = \mathbb{E}_q \left[ \log \frac{q(x)}{p(x)} \right] = \mathbb{E}_q \log q(x) - \mathbb{E}_q \log p(x) = -\mathbb{E}_q \log p(x) - \mathcal{H}(q(x))
\]
Mutual information

\[ I(x; y) = D_{KL}(p(x, y) \| p(x)p(y)) \]
\[ = E_{(x,y) \sim p(x,y)} \left[ \log \frac{p(x, y)}{p(x)p(y)} \right] \]
\[ = H(p(y)) - H(p(y|x)) = H(p(x)) - H(p(x|y)) \]

High MI?

x- it rains tomorrow, y – streets are wet tomorrow

x- it rains tomorrow, y – we find life on Mars tomorrow
Mutual information

\[ I(x; y) = D_{KL}(p(x, y) \parallel p(x)p(y)) \]

\[ = E_{(x,y) \sim p(x,y)} \left[ \log \frac{p(x, y)}{p(x)p(y)} \right] \]

\[ = H(p(y)) - H(p(y | x)) = H(p(x)) - H(p(x | y)) \]

example of mutual information: “empowerment” (Polani et al.)

\[ I(s_{t+1}; a_t) = H(s_{t+1}) - H(s_{t+1} | a_t) \]
The Plan

Information-theoretic concepts

Skill discovery

Using discovered skills

Hierarchical RL
Soft Q-learning

Objective:
\[
\sum_{t} E(s_t, a_t) \sim q [r(s_t, a_t) + \mathcal{H}(q(a_t|s_t))]
\]

Q-learning

1. collect dataset \(\{(s_i, a_i, s'_i, r_i)\}\)
2. set \(y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q(\phi)(s'_i, a'_i)\)
3. set \(\phi \leftarrow \arg \min_{\phi} \frac{1}{K} \sum_i \|Q(\phi)(s_i, a_i) - y_i\|^2\)
\[
\pi(a|s) = \arg \max_a Q(\phi)(s,a)
\]

Soft Q-learning

1. collect dataset \(\{(s_i, a_i, s'_i, r_i)\}\)
2. set \(y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q(\phi)(s'_i, a'_i)\)
3. set \(\phi \leftarrow \arg \min_{\phi} \frac{1}{K} \sum_i \|Q(\phi)(s_i, a_i) - y_i\|^2\)
\[
\pi(a|s) = \arg \max_a Q(\phi)(s,a) \propto \exp \left( \lambda_t(s) A_t(a) \right)
\]
Soft Q-learning

\[ \pi(a_t|s_t) = \mathcal{N}(\mu(s_t), \Sigma) \]

\[ Q(s_t, a_t) \]

\[ \pi(a_t|s_t) \propto \exp Q(s_t, a_t) \]

Exploration  
Fine-tunability  
Robustness

Haarnoja et al. RL with Deep Energy-Based Policies, 2017
Learning diverse skills

\[ \pi(a|s, z) \]

Why can’t we just use MaxEnt RL

1. **action** entropy is not the same as **state** entropy
   
   agent can take very different actions, but land in similar states

2. MaxEnt policies are stochastic, but not always **controllable**

   intuitively, we want low diversity for a fixed \( z \), high diversity across \( z \)’s

**Intuition:** different **skills** should visit different **state-space regions**

Eysenbach, Gupta, Ibarz, Levine. *Diversity is All You Need.*
Diversity-promoting reward function

$$\pi(a|s, z) = \arg \max_{\pi} \sum_{z} E_{s \sim \pi(s|z)}[r(s, z)]$$

reward states that are unlikely for other $z' \neq z$

$$r(s, z) = \log p(z|s)$$

Eysenbach, Gupta, Ibarz, Levine. Diversity is All You Need. Slide adapted from Sergey Levine
Examples of learned tasks

Cheetah

Ant

Mountain car

Eysenbach, Gupta, Ibarz, Levine. Diversity is All You Need.
A connection to mutual information

\[
\pi(a|s, z) = \arg \max_{\pi} \sum_z E_{s \sim \pi(s|z)}[r(s, z)]
\]

\[
r(s, z) = \log p(z|s)
\]

\[
I(z, s) = H(z) - H(z|s)
\]

maximized by using uniform prior \(p(z)\)
minimized by maximizing \(\log p(z|s)\)

Eysenbach, Gupta, Ibarz, Levine. Diversity is All You Need.
See also: Gregor et al. Variational Intrinsic Control. 2016

Slide adapted from Sergey Levine
The Plan

Information-theoretic concepts

Skill discovery

Using discovered skills

Hierarchical RL
How to use learned skills?

How can we use the learned skills to accomplish a task?

Learn a policy that operates on $z$’s

$$\pi(\mathbf{a}|\mathbf{s}, z)$$
Results: hierarchical RL

Can we do better?

Eysenbach, Gupta, Ibarz, Levine. Diversity is All You Need.
What’s the problem?

Skills might not be particularly useful  
It’s not very easy to use the learned skills

What makes a useful skill?
What’s the problem?

Consequences are **hard** to predict.

Consequences are **easy** to predict.
Slightly different mutual information

\[ I(z, s) = H(z) - H(z|s) \]

\[ \max \mathcal{I}(s', z \mid s) = \max \left( \mathcal{H}(s' \mid s) - \mathcal{H}(s' \mid s, z) \right) \]

\[ \mathcal{I}(x; y) = D_{\text{KL}}(p(x, y) \mid\mid p(x)p(y)) \]

\[ I(s'; z \mid s) \geq \mathbb{E}_s \mathbb{E}_z \mathbb{E}_{p(s' \mid s, z)} \left[ \log \frac{q_{\phi}(s' \mid s, z)}{p(s' \mid s)} \right] \]

\[ \approx \mathbb{E}_s \mathbb{E}_z \mathbb{E}_{p(s' \mid s, z)} \left[ \log \frac{q_{\phi}(s' \mid s, z)}{\sum_{i=1}^L q_{\phi}(s' \mid s, z_i)} + \log L \right] \]

Sharma, Gu, Levine, Kumar, Hausman, DADS, 2019.
Skill-dynamics model

We are learning a skill-dynamics model \( q(s' \mid s, z) \)

compared to conventional global dynamics \( p(s' \mid s, a) \)

Skills are optimized specifically to make skill-dynamics easier to model

Sharma, Gu, Levine, Kumar, Hausman, DADS, 2019.
DADS algorithm

\[ \text{Update } q_\phi(s' | s, z) \]
\[ \text{Compute } r_z(s, a, s') \]

Algorithm 1: Dynamics-Aware Discovery of Skills (DADS)

Initialize \( \pi, q_\phi \);

while not converged do

Sample a skill \( z \sim p(z) \) every episode;
Collect new \( M \) on-policy samples;
Update \( q_\phi \) using \( K_1 \) steps of gradient descent on \( M \) transitions;
Compute \( r_z(s, a, s') \) for \( M \) transitions;
Update \( \pi \) using any RL algorithm;

end

Sharma, Gu, Levine, Kumar, Hausman, DADS, 2019.
DADS results
Using learned skills

Use skill-dynamics for model-based planning
Plan for skills not actions
Tasks can be learned zero-shot
Summary

- Two skill discovery algorithms that use mutual information
- Predictability can be used as a proxy for “usefulness”
- Method that optimizes for both, predictability and diversity
- Model-based planning in the skill space
- Opens new avenues such as unsupervised meta-RL
  - Gupta et al. *Unsupervised Meta-Learning for RL*, 2018
The Plan

Information-theoretic concepts

Skill discovery

Using discovered skills

Hierarchical RL
Why Hierarchical RL?

Performing tasks at various levels of abstractions

Bake a cheesecake
Buy ingredients
Go to the store
Walk to the door
Take a step

Contract muscle X
Hierarchical RL – design choices

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy
Learning Locomotor Controllers

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy

Option Critic

A Markovian option $\omega \in \Omega$ is a triple $(I_\omega, \pi_\omega, \beta_\omega)$ in which $I_\omega \subseteq S$ is an initiation set, $\pi_\omega$ is an intra-option policy, and $\beta_\omega : S \rightarrow [0, 1]$ is a termination function. We also assume that $\forall s \in S, \forall \omega \in \Omega : s \in I_\omega$ (i.e., all options are available everywhere)

- Option is a self-terminating mini-policy
- Everything trained together with policy gradient

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy

Bacon, Harb, Precup. The Option-Critic Architecture, 2016.
Relay Policy Learning

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy

Relay Policy Learning

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy

1. Collect experience $s_t, g_t, a_t, R_t, \ldots$
2. Train $\mu^\alpha$ with experience transitions $(s_t, g_t, a_t, r_t, s_{t+1}, g_{t+1})$ using $g_t$ as additional state observation and reward given by goal-conditioned function $r_t = \tau(s_t, g_t, a_t, s_{t+1}) = -\|s_t + g_t - s_{t+1}\|_2$.
3. Train $\mu^h$ on temporally-extended experience $(s_t, g_t, \sum R_{t:t+c-1}, s_{t+c})$, where $g_t$ is re-labelled high-level action to maximize probability of past low-level actions $a_{t:t+c-1}$.
4. Repeat.

Figure 2: The design and basic training of HIRO. The lower-level policy interacts directly with the environment. The higher-level policy instructs the lower-level policy via high-level actions, or goals, $g_t \in \mathbb{R}^d$, which it samples anew every $c$ steps. On intermediate steps, a fixed goal transition function $h$ determines the next step’s goal. The goal simply instructs the lower-level policy to reach specific states, which allows the lower-level policy to easily learn from prior off-policy experience.

- Goal-conditioned policies with relabeling
- Off-policy training through off-policy corrections

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy

HRL Summary

- Multiple design choices and frameworks
- Helps with exploration and temporally extended tasks
- Can be difficult to get it to work
- Seems like a natural direction for harder RL problems

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Experiments</th>
<th>Important?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H1) Temporal training</td>
<td>Figures 2, 3</td>
<td>Yes, but only for the use of multi-step rewards (n-step returns).</td>
</tr>
<tr>
<td>(H2) Temporal exploration</td>
<td>Figures 2, 4</td>
<td>Yes, and this is important even for non-hierarchical exploration.</td>
</tr>
<tr>
<td>(H3) Semantic training</td>
<td>Figure 3</td>
<td>No.</td>
</tr>
<tr>
<td>(H4) Semantic exploration</td>
<td>Figure 4</td>
<td>Yes, and this is important even for non-hierarchical exploration.</td>
</tr>
</tbody>
</table>

Design choices:
- goal-conditioned vs not
- pre-trained vs e2e
- self-terminating vs fixed rate
- on-policy vs off-policy

Recap

Key learning goals:

• Understand the concept of a skill and basic algorithms in this space
• Overview of hierarchical RL algorithms

Skill discovery/learning:

• Connected to information-theoretic measures like mutual information
• Unsupervised but difficult to use in complex environments

Hierarchical RL:

• Many different options/methods
• Designed to cope with longer-horizon tasks
• Largely unsolved
Monday – no lecture

Guest lecture – Anna Goldie on various RL applications including LLMs and chip design