Literature Review: Unsupervised Skill Discovery in Hierarchical Reinforcement Learning

Hamza Emin Hacıoğlu (2786978)
Burak Muammer Yıldız (2529451)

Contents

1 Introduction

Deep reinforcement learning (RL) has shown good results in many tasks like such as playing games, controlling robots, and navigating in complex environments. However, most RL methods need a reward function that tells the agent what to do. In real life, intelligent creatures can explore their environments and learn useful skills without any external supervision. When they later face specific goals, they can use these skills to solve new problems quickly.

Learning skills without using external rewards is actually really useful in practice. First, many environments have sparse rewards, meaning the agent only gets feedback when it randomly reaches the goal state. If the agent can learn useful behaviors without supervision, then it can explore better in these harsh environments. Second, for long tasks that take many steps, these learned skills can act like building blocks for hierarchical RL. In hierarchical RL, a higher level controller just needs to decide which skill to use, instead of choosing every small action at every step. This makes the problem easier to handle. Third, in many real world situations, rewards often come from humans, and that kind of feedback is both expensive and slow. Unsupervised skill learning helps reduce how much human supervision we need.

In this project, we'll look at unsupervised skill discovery methods, mainly from an information theoretic point of view. We'll focus on DIAYN and VALOR as our main methods, and then briefly cover related work. We will also discuss how the discovered skills can be used for downstream tasks and as building blocks in hierarchical control. Finally, we will briefly explain why we chose our specific environments.

2 Unsupervised Skill Discovery via Mutual Information

The core idea behind modern unsupervised skill discovery is to use mutual information as the objective. Instead of hand designing the reward functions, we want the agent to learn a set of skills that are both diverse and easy to tell apart. Here, a "skill" is a policy conditioned on a latent variable z. Different values of z should lead to clearly different behaviors.

2.1 DIAYN: Diversity Is All You Need

DIAYN, introduced by Eysenbach et al. [1], learns skills by maximizing the mutual information between the visited states and the skill label. In simple terms, it tries to learn a set of different skills such that each skill tends to bring the agent to its own characteristic set of states.

The method is built around three key ideas:

1. **Skills should control where the agent goes.** Meaning that, for a skill to be meaningful, it should influence which states the agent visits. Different skills should lead the agent to different regions of the state space so that we can tell them apart just by looking at what states they reach.
2. **We shall distinguish skills using states, not actions.** Meaning that, the goal is to identify which skill is being used from the observed states alone. Actions that don't visibly change the environment aren't helpful for an external observer. So, DIAYN focuses on how skills affect state trajectories rather than on the raw action sequences.
3. **Skills should be as random as possible while still being distinguishable.** Meaning that, to encourage exploration, the agent is pushed to act with high randomness under each skill. However this encouragement continues as long as the resulting behavior is still different enough from the other skills to be identifiable.

The objective function:

$$\mathcal{F}(\theta) = I(S; Z) + \mathcal{H}[A|S] - I(A; Z|S) \tag{1}$$

combines three pieces consisting the state variable S, the action variable A (random variables for states and actions), and the latent skill variable Z sampled from a prior $p(z)$. This can be rewritten in terms of entropies as:

$$\mathcal{F}(\theta) = \mathcal{H}[Z] - \mathcal{H}[Z|S] + \mathcal{H}[A|S, Z] \tag{2}$$

The first term $\mathcal{H}[Z]$ encourages the prior over skills to have high entropy. DIAYN fixes $p(z)$ to be uniform, so this is already maximized. The second term $-\mathcal{H}[Z|S]$ says it should be easy to infer the skill from the current state. And lastly, third term $\mathcal{H}[A|S, Z]$ says each skill should act as randomly as possible, so skills remain stochastic rather than collapsing to deterministic behaviors.

Since we cannot compute $p(z|s)$ exactly, DIAYN uses a learned discriminator $q_\phi(z|s)$ to approximate it. This gives a variational lower bound on the objective. The intrinsic reward that the agent maximizes becomes:

$$r_z(s, a) = \log q_\phi(z|s) - \log p(z) \tag{3}$$

The training works like this: At the start of each episode, we sample a skill z from the fixed prior $p(z)$. The agent then follows the policy $\pi(a|s, z)$ with that same skill for the whole episode. For each state visited, we compute a reward using the discriminator. We update the policy with Soft Actor-Critic (SAC) [9] to maximize this intrinsic reward, and we update the discriminator with supervised learning so it gets better at predicting z from the states.

A key design choice in DIAYN is to keep the prior over skills fixed instead of learning it. In earlier work like Variational Intrinsic Control (VIC) [3], the prior is learned. This can cause a problem: the learned prior starts to favor skills that already look diverse, so those skills get sampled more, trained more, and the less developed skills are basically ignored. By keeping $p(z)$ fixed and uniform, DIAYN samples all skills equally, which helps it learn a broader and more diverse set of behaviors.

In experiments, DIAYN is able to learn a wide range of locomotion skills in environment such as Half Cheetah, Hopper, and Ant. Even without any external task reward, the agent discovers skills like running forward, running backward, jumping, flipping, and more. Some of these skills even end up solving standard benchmark tasks, despite never being trained directly on those task rewards.

2.2 VALOR: Variational Option Discovery

VALOR, introduced by Achiam et al. [2], takes a different approach. While DIAYN looks at single states to distinguish skills, VALOR looks at whole trajectories. The idea is to learn "dynamical modes" (for example, moving in a circle over time) rather than "goal modes" (like going to a specific position X).

The objective is:

$$\max_{\pi, D} \mathbb{E}_{c\sim G} \left[\mathbb{E}_{\tau\sim\pi, c}[\log P_D(c|\tau)] + \beta\mathcal{H}(\pi|c)\right] \tag{4}$$

where c is a context (similar to skill z in DIAYN), τ is a trajectory, P_D is the decoder probability. And, β is a coefficient that controls the strength of the entropy regularization term.

There is a natural connection to variational autoencoders (VAEs). Here:

- the context c plays the role of the "data" (what we want to renconstruct)
- the trajectory τ is like the latent representation
- the policy together with the environment acts as an encoder
- and the decoder D tries to recover the context from the trajectory.

VALOR makes two key design choices for the decoder:

1. The decoder never sees actions, it only observes states (or observations). This forces the agent to actually move and interact with the environment to indicate which context it is following. If the decoder could see actions, the agent could "cheat" by just encoding the context directly in its actions without moving.
2. The decoder does not decompose as a sum over timesteps. In DIAYN, the instrinsic reward is $\sum_t \log q(z|s_t)$. In constrast, VALOR uses a recurrent neural network (bidirectional LSTM) to process the whole trajectory at once. Thanks to that, VALOR capture dynamical patterns that only make sense over time, not just at individual states.

VALOR also introduces a curriculum like approach for training. Instead of starting with many contexts and trying to distinguish them all at once, it starts with a small number K and only increases it when the decoder is doing well on the current set. More concretely, when $P_D(c|\tau)$ becomes high enough on the existing contexts, VALOR increases the number of contexts using

$$K \leftarrow \min(1.5 \times K + 1, K_{max})$$

. This makes training more stable and allows learning many more modes since gradually expands the set of contexts.

Experiments show that VALOR can learn behaviors similar to DIAYN, but because it looks at trajectories, it can produce results which looks different in practice. For example, VALOR tends to learn movement patterns (like looping or oscillatory motions) rather than just reaching different final states.

2.3 Related Methods

Several other methods explore similar ideas:

Variational Intrinsic Control (VIC) [3] learns skills by maximizing mutual information between the skill and the final state of a trajectory. Unlike DIAYN, it learns the prior over skills and only looks at the last state. As previously discussed, DIAYN improves on this by fixing the prior and using all states, and that gives a denser reward signal and helps avoid mode collapse.

SNN4HRL [4] uses a similar information-theoretic objective but adds a task-specific proxy reward to encourage exploration toward useful states.

Contrastive Intrinsic Control (CIC) [5] also maximizes mutual information between state transitions and skills. But, it uses contrastive learning instead of a discriminator. This achieves faster adaptation on downstream tasks compared to DIAYN.

DISCERN and Skew-Fit [6, 7] view skill discovery as goal-conditioned policy learning. They still use discriminators to encourage diverse behaviors. But, they frame the problem in terms of reaching different goals rather than labeling "skills" direclty.

Information Geometry of Unsupervised RL [8] provides theoretical analysis of mutual-information-based skill discovery. It shows that, under certain assumptions these methods are optimal for minimizing regret with respect to unknown rewards functions.

Overall, these approaches share the same core idea: in order to learn diverse skills use intrinsic motivation, usually based on mutual information. They mainly differ in which parts of the trajectory they use, whether the skill prior is fixed or learned and how they balance exploration with being able to distinguish skills.

3 Skills for Downstream Tasks

The main motivation for unsupervised skill discovery is that learned skills should be useful for with downstream tasks. After the unsupervised phase, there are a few common ways to reuse

these skills.

Policy Initialization: The simplest approach is to treat the learned skills as pretrained policies. On a new task, we can evaluate each skill; then pick the one that gets the highest reward, and then fine-tune it using the task reward. In fact, this is similar to how pretrained models are used in computer vision. Experiments over there, show that this usually leads to faster learning compared to training from scratch.

Hierarchical RL: Skills can also be used as options [10] or action primitives in a hierarchical setup. A high-level (meta-)policy chooses which skill to execute for the next k steps. This effectively shortens the decision horizon and can make sparse reward problems more tractable. Using this approach, DIAYN shows good results on tasks like ant navigation and cheetah hurdle jumping.

Imitation Learning: Given an expert trajectory, we can find which skill best matches it by computing

$$\hat{z} = \arg\max_{z} \prod_{s_t \in \tau^*} q_\phi(z|s_t).$$

This allows us to imitate behaviors without needing any action labels.

However, we believe that there are some concrete limitations. Most papers show that skills help with specific tasks in the same environment where they were learned. It is still unclear how well these skills transfer to very different tasks or environments. In addition, the relationship between "more diverse skills" and better downstream performance is not fully understood.

4 Continual Learning and Transfer

Our project focuses on two less explored aspects of skill discovery: forward transfer and catastrophic forgetting.

Forward transfer refers to whether learning skills first make it faster to learn later tasks. After a skill-learning phase, can we solve a sequence of tasks more quickly because we can reuse those skills?

Catastrophic forgetting refers to whether we lose performance on behaviors we learned before, when we fine-tune them for new tasks?

Most continual RL work addresses these problems using replay buffers, regularization methods (like EWC), or architectural tricks (like separate heads for different tasks). We instead ask a different question: can learned skills themselves act as a kind of structured prior that both improves transfer and reduces forgetting?

Our hypothesis is that if skills partition the behavior space in a useful way, then new tasks might only require learning which skills to use, rather than changing the skill themselves. Keeping low-level skills fixed could naturally prevent forgetting.

5 Environment Choice

For our project, we focus on continuous control environments from DMControl (using Gymnasium wrappers). These include:

- **Cheetah**: A 2D running robot. Skills typically show different speeds and directions (forward, backward, flipping).
- **Walker**: A 2D humanoid that can walk, run, and balance. Shows diverse locomotion gaits.

We choose these environments because of several reasons. First they are the standard benchmarks used in DIAYN and VALOR papers, so we can compare our results. And skills in these

environments are visually clear: for example different gaits, speeds, and movement styles. Finally, they are complex enough to be interesting but simple enough to train in reasonable time. However, we are open to any suggestions.

Lastly, we may also use MiniGrid environments for some experiments, especially for the high-level controller and multi task settings. It is mainly because, MiniGrid provides simple discrete navigation tasks where we can more easily measure forward transfer and forgetting.

6 Summary of Review

Both VALOR and DIAYN, show that diverse skills emerge naturally from the information-theoretic objective. These skills can be used for policy initialization, hierarchical RL, and imitation learning. However, our observation is that most work focuses on showing that skills help in specific downstream tasks without systematically studying transfer and forgetting.

We hope, our project will:

1. Implement DIAYN and VALOR in DMControl environments (Cheetah, Walker, ?).
2. Analyze what kinds of skills emerge. For instance their structure, discriminability, and temporal consistency.
3. Study how these skills can be reused by a high-level controller in multi-task settings.
4. More importantly, measure whether skills speed up learning new tasks and does learning new tasks harm old skills.

We hope this will give a better sense of whether unsupervised skill discovery can actually be a practical basis for continual and hierarchical RL in practice.

References

[1] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is All You Need: Learning Skills without a Reward Function. *arXiv preprint arXiv:1802.06070*, 2018.

[2] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational Option Discovery Algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[3] K. Gregor, D. Rezende, and D. Wierstra. Variational Intrinsic Control. *arXiv preprint arXiv:1611.07507*, 2016.

[4] C. Florensa, Y. Duan, and P. Abbeel. Stochastic Neural Networks for Hierarchical Reinforcement Learning. *ICLR*, 2017.

[5] M. Laskin, et al. Contrastive Intrinsic Control for Unsupervised Skill Discovery. *arXiv preprint*, 2022.

[6] D. Warde-Farley, et al. Unsupervised Control Through Non-Parametric Discriminative Rewards. *ICLR*, 2019.

[7] V. Pong, et al. Skew-Fit: State-Covering Self-Supervised Reinforcement Learning. *ICML*, 2019.

[8] B. Eysenbach, R. Salakhutdinov, and S. Levine. The Information Geometry of Unsupervised Reinforcement Learning. *ICLR*, 2022.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *ICML*, 2018.

[10] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181-211, 1999.