RL: Introduction to Hierarchical Reinforcement Learning
Andrea Mattei
You are surfing the web, when you stumble on this website
MineRL

Finds diamonds as fast as possible
First Try

Agent

States, Rewards → Actions

Environment
First Failure

In Minecraft tools have many iterations

We need generalization and transfer learning
First Failure

States are **big** and **complex**!

We need to deal with big states efficiently
First Failure

Limited training **samples**!

We need sample efficiency

The environment gives **sparse rewards**!

We need to deal with sparse rewards
What we need

• We need generalization and transfer learning
• We need to deal with big states efficiently
  • We need sample efficiency
  • We need to deal with sparse rewards
What we need

- We need generalization and transfer learning
- We need to deal with big states efficiently
  - We need sample efficiency
  - We need to deal with sparse rewards

Hierarchical Reinforcement Learning (HRL) can help us!
Hierarchical Reinforcement Learning (HRL)

In Hierarchical Reinforcement Learning the agent divides the task into sub-tasks

Collect Diamonds

Build Pickaxe

Find Wood/Iron
How HRL Help Us

- We need generalization and transfer learning
  **HRL**: sub-tasks can be reused

- We need to deal with big states efficiently
  **HRL**: sub-tasks can operate on a subset of the state space

- We need sample efficiency
  **HRL**: sub-tasks are uniquely trained

- We need to deal with sparse rewards
  **HRL**: sub-tasks can model sparse reward goals
Let’s Pick Our Tools
Markov Decision Process (MDP)

A Markov Decision Process is defined as a tuple \(< S, A, P, R, \gamma >\) where:

- \(S\) is a set of states
- \(A\) is a set of actions
- \(P\) is a state transition function defined as \(P_{s s'}^a = P[S_{t+1} = s'|S_t = s, A_t = a]\)
- \(R\) is a reward function defined as \(R_s^a = E[R_{t+1}|S_t = s, A_t = a]\)
- \(\gamma\) is a discount factor \(\gamma \in [0,1]\)
Semi-Markov Decision Process (SMDP)

SMDPs allow modelling of **continuous-time discrete-event** systems.
Options

An option $\omega$ is defined by 3 parameters $<I_\omega, \pi_\omega, \beta_\omega>$:

- $I_\omega$ is the initiation set with $I_\omega \subseteq S$
- $\pi_\omega$ is a policy
- $\beta_\omega$ is the termination conditions $\beta_\omega: S \rightarrow [0,1]$
The 2 Main Approaches of HRL

- Feudal RL
- Option-critic
Feudal Reinforcement Learning

Feudal RL -> HRL
The main idea is to have an hierarchy of managers. Like in Feudalism, managers have absolute power over their sub-managers.
Feudal Reinforcement Learning

The main idea is to have an hierarchy of managers. Like in Feudalism, managers have absolute power over their sub-managers.
Reward Hiding

«Managers must reward sub-managers for doing their bidding whether or not this satisfies the commands of the super-managers»
Information Hiding

«Managers only need to know the state of the system at the granularity of their own choices of tasks»
From the idea to the implementation

In 2017 Vezhnevets et al. proposed a Deep Reinforcement Learning implementation (FuN) of Feudal Reinforcement Learning.
FuN architecture

Manager

Worker

Goals, Rewards

States

Rewards

Actions

Environment
An in-depth look at the agent
Manager

The manager produces goals for the worker
The worker acts on the environment according to its goal.
Worker’s Reward

- Unlike the original FRL formulation, reward hiding is not strictly enforced
- The overall reward is given as:
  \[ R = R_t + \alpha R_t^I \]
- The intrinsic reward is calculated starting from the goal as:
  \[ R_t^I = \frac{1}{c} \sum_{1}^{c} d_{cos}(s_t - s_{t-1}, g_{t-1}) \]
Result on Montezuma’s Revenge

**FuN** obtained great result on Montezuma’s Revenge, a game infamous for its **sparse rewards**
Option-Critic

HRL

Option-critic
The Option-Critic architecture extends the Actor-Critic one by introducing options.
A Recap on Actor-Critic

![Diagram of Actor-Critic]

- Actor
- Critic
- States
- Rewards
- Actions
- TD Error
- Gradient
- Environment
Option-Critic

- Policy over options
- Options
- Gradients
- TD Error
- States
- Critic
- Rewards
- Actions
- Environment
Intra-Option Learning

For learning an option \( \omega \), it is necessary to learn both the option policy \( \pi_{\omega,\theta} \) and the termination function \( \beta_{\omega,\theta} \).

Intra-Option Policy Gradient Theorem:

\[
\frac{\partial Q_{\theta}(\omega, s)}{\partial \theta} = E \left[ \frac{\partial \pi_{\omega,\theta}(a|s)}{\partial \theta} Q_U(w, s, a) \right]
\]

Termination Gradient Theorem:

\[
\frac{\partial U(\omega, s)}{\partial \nu} = E \left[ - \frac{\partial \beta_{\omega,\nu}(s)}{\partial \nu} A_{\theta}(w, s) \right]
\]
Critic

The Critic model learns to approximate the state-option value function $Q_\Omega(s, \omega)$.

With it the critic estimates:

- The action-option-state value function $Q_U(w, s, a)$
- The advantage function over options $A_\Omega(w, s) = Q_\Omega(s, \omega) - V_\Omega(s)$ with $V_\Omega(s)$ as the value function
Visualizing the Options

The image is the trajectory of the original implementation of the Option-Critic playing Seaquest with 2 options.
In Feudal Reinforcement Learning, sub-tasks must be fixed by the programmer. On the contrary, in Option-Critic sub-tasks are learned automatically.
Challenges of Option-Critic

During training, options could collapse into:

• A single active option that completes the entire task
• A set options that changes at every step

Another limitation of the architecture is the assumption that options can apply everywhere
Challenges of FuN

The main challenge of FuN is the fact that the goal depends on the state. In particular, it depends on the distance function used to generate the rewards.

Superficially similar states can lead to wrong/ineffective goal generations!
Thank for your attention
I hope we will have a nice discussion!

To be continued in Edoardo’s presentation...
References

Presented Papers
• Feudal Reinforcement Learning (Dayan et al., 1993)
• FeUdal Networks for Hierarchical Reinforcement Learning (Vezhnevets et al., 2017)
• The Option-Critic Architecture (Precup et al., 2016)

Additional Resources
• Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning (Sutton et al. 1999)
• https://thegradient.pub/the-promise-of-hierarchical-reinforcement-learning/
• https://towardsdatascience.com/hierarchical-reinforcement-learning-a2cca9b76097

MineRL
• Official website: https://minerl.io/