What is compositional Reinforcement Learning ?

Abstract

1 What is Reinforcement Learning ?

Reinforcement Learning (RL) is a methodology aiming at training an *agent* to interact with an *environment*. What does that mean ?

1.1 Reinforcement Learning is not supervised or unsupervised Machine Learning

A good way to understand what RL is about is to see how it contrasts with the other two main machine learning (ML) frameworks that are supervised ML and unsupervised ML. In both the supervised and unsupervised frameworks, data is readily available as a whole for the *model* to train on. This means that a model can be trained in parallel on all the datapoints. For a given model one gets some estimation of a global error or *loss* on the entire dataset. Training in these settings corresponds to searching for models minimizing the global loss, ie. training happens on all the data at a time.

ML	RL
datapoint	state
loss	reward
model	policy
model application	action

Table 1: Concepts in ML and corresponding RL concepts

In contrast to this, RL describes situations where the data is only accessible through the iterated interaction of an agent with an environment. In this context, we do not just have a *model* meant to be applied to some data to get some loss, we have a *policy* that tells us to take some *action* at a current *state* to get both a *reward* and a new state. This complicates things on different levels.

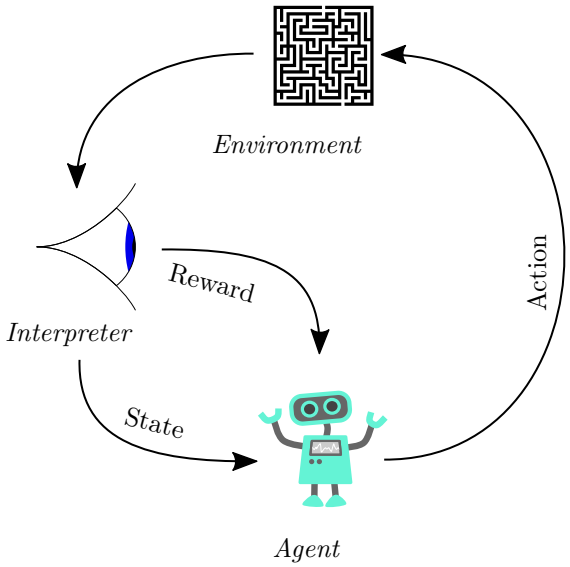

Figure 1: Typical framing of an RL scenario

1.2 What makes RL difficult ?

First, data acquisition and loss observation have been intertwined: taking an action means both receiving a new state and a reward. This leads to a tension between *exploration* (taking new actions) and *exploitation* (taking the action with the best reward seen so far).

Second, the training can't happen in parallel on all the data at a time anymore, since it has become inherently sequential.

This leads to the problem of the *sparsity of the reward function*. Imagine that you want an agent to reach some goal state in a big environment. If you only add a reward when the agent takes an action reaching the goal, you run into the risk of it never taking one of these rewarding actions by trying take actions randomly. On the other hand if you reward the agent with small rewards for taking intermediate steps towards a goal, it could get stuck on only reaching these, satisfying itself with the small rewards.

This means that the reward can give very little information on the actual environment in which you are in, and optimization can get stuck in suboptimal policy. This is not inherently a problem of RL since ML problems where the loss function has complicated dynamics have similar difficulties.

1.3 How do we make RL problems manageable ?

MDPs are models of RL. We want to decompose MDPs to make them manageable. The tool of choice to study (de)composability is category theory.

Hierarchical RL is another approach.

2 What is RL ?

Reinforcement learning is about learning how to choose the best possible actions in different situations to earn the highest possible reward. The agent isn't told exactly what to do but has to figure it out by trying different actions and seeing what works. What makes reinforcement learning interesting is that actions can affect not just the immediate reward but also future situations and rewards. So, the agent improves over time by experimenting and learning from obtained rewards.

In reinforcement learning, a learning agent needs to interact with its environment and choose actions based on the situation it's in. The agent should also have a goal, where completing the task leads to a reward. Any problem that fits this kind of setup can be approached using reinforcement learning methods.

In interactive problems where an agent has to work with an environment, it's often hard to provide perfect examples of what the agent should do in every possible situation. In unknown situation where one would expect learning to be most beneficial, an agent must be able to learn from its own experience.

All reinforcement learning agents have a clear goal and receive rewards for taking actions while interacting with the environment. Reinforcement learning has a fully interactive setup where the agent learns by trying to achieve a goal and earning rewards. Sometimes, the agent interacts directly with the main system and indirectly with the system's environment.

RL frequently involves breaking down a problem into smaller subtasks and combining the learned behaviors from these subtasks. Compositionality in RL has the potential to create modular subtask units that interfere with other system capabilities.

Imagine teaching a robot to grab a ball and move it to another place. Or teaching a robot to clean the clothes and then rinse it and dry it. At first these tasks seem to be unrelated, but both are compositional problems where smaller tasks combine into complex ones.

When I think of these subtasks and their compositions I would like to think of it as the puzzle. Where puzzle pieces are our subtasks and fixing appropriate pieces together is like the composition to obtain the full puzzle or the final combined task you may call it.

However, building compositional models means we need to understand what are the minimum set of assumptions needed to make sure the compositional property holds.

Then there are questions like, How can we formally describe what happens when reinforcement learning agents interact with environments and with each other? How can we rigorously express the composition of such systems?

Some limitations or challenges of RL: Exploration and exploitation- One of the main challenges in reinforcement learning, which doesn't usually come up in other types of Machine Learning, is balancing exploration and exploitation. To get high rewards, the agent should choose actions that have worked well in the past. But to find those good actions in the first place, it needs to try new ones it hasn't tried before. So, the agent has to exploit what

it already knows to earn rewards, while also exploring new options to improve. It can't only explore or only exploit, doing just one won't work. The agent has to keep trying different actions and eventually focus on the ones that gives best rewards. In situations where outcomes are random, each action needs to be tried multiple times to understand how well suited it really is.

In this blogpost we will see how author of this paper develope a framework for a compositional theory of RL using a categorical point of view. This approach bridges machine learning and category theory in a way that should interest people from both fields.

Hierarchical Reinforcement Learning: Hierarchical Reinforcement Learning (HRL) is a version of reinforcement learning that adds a layered structure to how the agent learns. In standard RL, the agent learns how to choose actions based on the current situation. In HRL, the agent learns at different levels of abstraction. This hierarchical approach not only simplifies the learning process but also improves the scalability and efficiency

In HRL some decisions are about big-picture goals, while others handle smaller steps to reach those goals. In HRL, a big task is split into smaller sub-tasks, and those can be broken down even further if needed. Each level in the hierarchy focuses on a different part of the task. This makes learning easier and helps the agent become more efficient, especially when dealing with complex problems. Each level of the hierarchy focuses on solving a specific aspect of the overall task, making it easier for the agent to learn and optimize policies at different levels of abstraction. Hierarchical approach simplifies the learning process and improves the scalability and efficiency of RL algorithms.

Advantages/Challenges of HRL: HRL improves scalability by enabling the machine to learn from decomposed smaller tasks, allowing it to tackle tasks with large state-action spaces hierarchically. HRL allows transferability of information across different tasks and we don't need to learn from the scratch. Hierarchical Reinforcement Learning (HRL) has its own challenges. For example, expecting a robot to figure out the smaller steps or goals on its own is asking a lot. Also, HRL makes the system more complex, which makes it harder to understand how the final decisions are made. This added complexity can also make it difficult to use standard computational methods.

2.1 Category-theoretic RL

Here or somewhere we can say few things about RRR paper.

2.2 Compositional Reinforcement Learning from Logical Specifications

(Not sure about including things about this)

2.3 Category theory and Machine learning

Something interesting

2.4 Hierarchial RL

3 Preliminaries

3.1 Markov Decision Processes

Reinforcement learning problems such as the grid world robot are modeled using **Markov Decision Processes (MDPs)**. The key idea is representing RL systems as objects in certain category and morphisms in a category gives composition corresponds to connecting RL systems together. This paper formalizes idea of coposition using categorical semantics for RL, treating MDPs as objects in a category where compositionality is governed by pushouts and pullbacks. Let's define Markov Decision Processes categorically:

Definition 3.1 *An MDP $\mathcal{M} = (S, A, \psi, T, R)$ is a 5-tuple consisting of the following data:*

- *A measurable space S with fixed σ-algebra, called the state space of $\mathcal{M}$.*
- *A set A, called the state-action space of $\mathcal{M}$. This is the set of actions available at all different states of S.*
- *A function $\psi : A \to S$ that maps an action $a \in A$ to its assosiated state $s \in S$.*
- *A transition probability function $T : A \to \mathcal{P}_S$, where $\mathcal{P}_S$ denotes the space of probability measures on S.*
- *A reward function $R : A \to \mathbb{R}$.*

MDPs form a category MDP whose morphisms are defined as

$$m = (f, g) \colon \mathcal{M}_1 = (S_1, A_1, \psi_1, T_1, R_1) \to \mathcal{M}_2 = (S_2, A_2, \psi_2, T_2, R_2)$$

consisting of a measurable function $f \colon S_1 \to S_2$ and a function $g : A_1 \to A_2$ satisfying the following compatibility conditions:

1. The diagram

$$\begin{array}{ccc} A_1 & \xrightarrow{\ g\ } & A_2 \\ {\scriptstyle \psi_1}\downarrow & & \downarrow{\scriptstyle \psi_2} \\ S_1 & \xrightarrow[\ f\ ]{} & S_2 \end{array} \tag{1}$$

   is commutative.

2. The diagram

$$\begin{array}{ccc} A_1 & \xrightarrow{g} & A_2 \\ {\scriptstyle T_1}\downarrow & & \downarrow{\scriptstyle T_2} \\ \mathcal{P}_{S_1} & \xrightarrow[f_*]{} & \mathcal{P}_{S_2} \end{array} \tag{2}$$

   is commutative, where f_* maps a probability measure $\mu_1 \in \mathcal{P}_{S_1}$ to the pushforward measure $\mu_2 = f_*\mu_1 \in \mathcal{P}_{S_2}$ under f.

3. The diagram

$$\begin{array}{ccc} A_1 & \xrightarrow{g} & A_2 \\ & {\scriptstyle R_1}\searrow & \downarrow{\scriptstyle R_2} \\ & & \mathbb{R} \end{array} \tag{3}$$

   is commutative.

4 Pullbacks and Pushouts

4.1 Pullbacks

4.2 Pushouts/Gluing

5 Avoiding obstacles/ Putting safety conditions: Safe grid worlds

6 Zig-zag diagrams

7 Performing a compositional task: Example

8 Some observations

This paper is basically not just about reinforcement learning but it's also about how to think in terms of structure both Latent as well as Modular. The big complex systems are just smaller pieces locked together in the right way, and category theory gives us the rules for structure for how to glue/compose them properly. It's like if you understand how to connect pieces, you understand the whole thing and category theory gives you the one way to do that.

References