# [Lecture-1] Reinforcement Learning- David Silver

Notes by Ritobrata Ghosh

Distributed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) License

Notes are meant to be read with the Lectures and the Slides. Reading these notes in a standalone manner is not recommended.

---

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0388ff33-cc30-4da7-8711-150fdad10122/intro_RL.pdf

Course Slides [Provided under Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) Lisence]

---

## Difference Between RL and Supervised Learning

The agent can actually change the environment. This is not true for Supervised Learning.

The order of data in SL doesn't matter, but it is crucial in RL.

Rewards are long term things, as opposed to the use of loss function.

---

## Rewards

Rewards are scalar feedback signals.

$R \in \mathbb{R}$

Reward Hypothesis: All goals can be described by the maximization of expected cumulative reward.

---

The agent can only see the history-

"

There are other things in the environment, but the agent can only see the history.

The agent selects action based on history → The environment gives the reward.

---

History is not practical. It is absurdly long, and it is inefficient to look back every time the agent needs to pick an action.

So, we use State.

The environment state $S_t^e$ is the environment's private representation. But the agent is not privy to the entirety of the environment's state.

The Agent only sees the observation and Reward.

So, as the agent lack a complete picture of the environment, the agent builds and stores information about the environment in itself. It is the agent state $S_t^a$. The agent uses this agent state to pick the next action.

It is in our hands to pick what informations are stored in the agent.

This state is any function of history:

"

We get to build the $f$.

---

## Markov State

A Markov state holds all important information from history.

A state is a Markov state, iff:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, ..., S_t]$$

The probability of $S_{t+1}$ happening given the current state equals the probability of the same state given all of history in Markov states. In other words, *you can throw away the history*. The whole history does not give you any more information than the current state that you are in.

> 🤖 ***The future is independent of the past given the present.***

Once the state is known, the history can be thrown away. The current state is sufficient statistics of the future (this includes future rewards).

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

Rewards are included in history.

Example: If you are flying a helicopter, then if you know the wind speed, the angular velocity, the altitude, fuel, etc. then that information is good enough to take the next step. What the helicopter did 1 hour ago- we no longer need that information.

(For non-Markov states, you need to remember history to determine relevant information. If you do not have information about the state such as angular velocity, fuel, etc., you need to examine and calculate those from history.)

---

The environment state $S_t^e$ is a Markov state.

The whole history, if we have it, is also Markov.

It is possible to formalize a Markov state always.

---

What we believe will happen next depends on our representation of state.

---

## Fully Observable Environments

When we have a fully observable environment, the agent directly observes the environment state. (This is what we see in gridworlds).

$$O_t = S_t^a = S_t^e$$

This is **M**arkov **D**ecision **P**rocess.

---

We have to develop and design states such that the problem becomes an MDP. This way, the problem becomes much more solvable.

---

## Partially Observable Environments

- A trading agent can only see current prices, and not future prices.
- Pocker playing agents cannot see opponent's cards.
- A chess engine does not know what future moves the opponent will play.

**In these cases, the agent state does not equal the environment state.**

**The agent must construct** $S_t^a$.

This can be done in few ways-

- Complete history is seen as a state: $S_t = H_t$
- Beliefs as environment state: $S_t^a = (\mathbb{P}[S_t^e = s_1], ..., \mathbb{P}[S_t^e = s_t])$, we don't know what's going on with the environment, so we maintain a list/vector of probability distribution for the environment state based on where the agent thinks it is. This is a Bayesian approach. We take actions based on the probability of an agent state being as close as possible to the environment state.
- Using RNNs: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_a)$. We have two weight matrices to weigh the previous state and current observations and form a linear combination of them, to put importance on previous and current observations. And have a nonlinear activation function to decide the current state.

---

Up until this point, we only talk about the problem setting. Not how to solve it. From this point, we get involved in looking into the agent.

---

An RL agent may include one/more of these components:

1. Policy: how an agent picks his action. This decides what action to take in a given state.
2. Value function: how good it is to be in a state and/or pick an action. Estimating how well we are doing in a particular situation.
3. Model: The agent's estimation of the environment. As we will see later, it is two things- the state transition function (how state $S$ transits to $S'$), and the reward function (how much reward is awarded in a given state and action).

---

## Policy

Policy is the agent's behaviour.

It is a map from state to action.

Policy can either be deterministic (always choosing the same policy in a particular state): $\boxed{a = \pi(s)}$, or stochastic (action sampled from a distribution given a state): $\boxed{\pi(a|s) = \mathbb{P}[A = a|S = s]}$.

---

## Value Function

A prediction of expected future reward.

Used to evaluate goodness/badness of states and to select between actions (we can compare between two actions wrt the Value function, to choose which one to take).

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...|S_t = s]$$

(We choose action based on our policy $\pi$).

We control $\gamma$ to choose how far into the future we choose to look and consider.

---

## Model

The model predicts what the environment will do next. It is the model's own estimation of the environment.

It has two parts-

1. **Transition**: Predicts the dymanics of the environment- how one state transits to the next- how the state changes into a new one.
   $\boxed{\mathcal{P}^a_{ss'} = \mathbb{P}[S' = s' | S = s, A = a]}$
2. **Rewards**: Predicts the next immediate reward. $\boxed{\mathcal{R}^a_s = \mathbb{E}[R | S = s, A = a]}$

Model is optional.

---

Types of RL agents

- Value based- No explicit policy, Value function present. We work with values.
- Policy based: No value stored, we work with policy.
- Actor Critic: We have both values and policies. And we both to make both better.

---

- Model-free: No model of the environment present. Training is done solely on the basis of plicy and/or value function.
- Model-based:  A model of the environment is created. Policy and/or value functions also used.

---

Learning and Planning are two parts of RL. They are interconnected to solve the RL problem. But they are different problem setups.

**We need to learn first in order to plan.**

When we have a proper idea of the environment, we can use planning methods such as lookahead to plan for better actions.

---

- Exploration finds out more information about the environment.
- Exploitation exploits knowledge of the environment to maximize reward.

---

**Prediction**: Evaluate the future given a policy. (How good the agent will do if I follow a policy?)
**Control**: Optimize the future, by finding the best policy. (Now that I know which policy leads to what rewards, which policy should I take to maximize rewards?)

In order to begin solving the Control problem, you need to solve the Prediction problem.