Chapter 2: Mathematical Foundations of RL

Components of RL:

→ The 2 core components in RL are the agent and the environment

→ The agent makes the decisions and is the entity that is "learning" from trial and error to solve the problem

→ The environment is the representation of the problem, (ex: The air is the environment for a helicopter learning to fly)

Environment goes through internal state change as a consequence of agent's action

State: It is the information used to determine what happens next. 2 states exists:

1) Environment State (S^e_t) is the environments private representation
   → It is the data used by the environment used to pick the next observation/Reward
   → Not entirely visible to the agent, and even if visible, may contain irrelevant information.

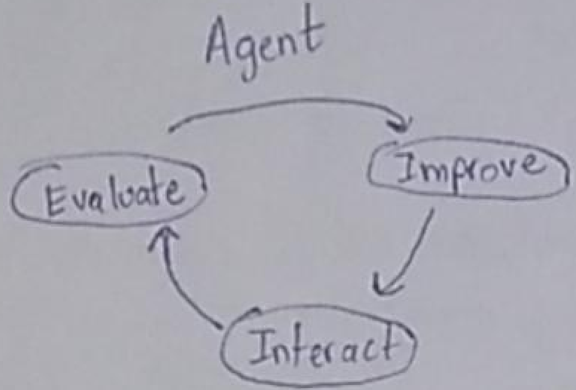
2) <u>Agent State</u> (S_t^a) is the agent's internal representation

→ It is the information the agent uses to pick the next action.

→ It can be any function of <u>History</u>, i.e $S_t^a = F(H_t)$

(By History, we mean the sequence of observations, rewards, and actions upto time t)

The agent goes through 3 internal steps:



Agents will be discussed properly in later chapters.
For now, we will look at environments closely.

The environment:

→ Most real-world decision-making problems can be expressed as RL-environments.

→ A mathematical framework called Markov Decision Process (MDP) will be used to model the problem and MDP's are the most common way to represent decision making processes in RL.

→ The "assumption" that RL makes is that every environment has an MDP working under it.

→ The environment is represented by a set of variables related to the problem. The combination of all possible values ~~of~~ that these variables take is the state space. A state is a specific set of values the variables take at any given time. (S^e_t).

→ Agents, often, do not have access to the actual environment's state, but the agent does observe the environment. The set of variables the agent perceives at any given time is called an observation

→ The combination of all possible values these variables can take is the observation space

('State' and 'observation' are usually used interchangeably)

→ At every state, the environment allows ~~a set of actions~~ the agent to perform ~~an action~~ certain actions.

The set of all actions in all states is the "action space".

(Usually the actions allowed by the environment are the same for each state)

→ The agent performs an action and this may influence the environment and therefore the state of the environment may change. The function that governs this state change is the transition function

→ After the transition, the agent takes in a new observation. The environment may also provide a reward signal as a response to this transition. The function responsible for sending a reward is called the reward function.

→ The set of transition and reward function is called the model of the environment.

→ The environment commonly has a well-defined task. (ex. winning the game in chess, learning how to walk for a robot, etc)

→ The goal of this task is defined through the reward signal. The reward signal can be dense, sparse or anything in between. When we model environments, reward signals are the way to train your agent the way you want

→ The denser the reward signal, the faster the agent will learn.

→ The interaction between agent and environment typically go on for several cycles. Each cycle is a time step.

→ It is a unit of time and can be anything from a millisecond to a day or any other periods of time.

→ At each time step, the agent takes in an observation, takes an action, and receives a new observation and maybe a reward.

→ The set of the observation, the action, the reward and the new observation is called an experience tuple.

→ Tasks in the environment may or may not have a natural ending. Tasks that do (ex. chess ends in checkmate) are called episodic tasks. Tasks that do not (ex. walking) are called continuing tasks.

→ The sequence of timesteps from the beginning to the end of an episodic task is called an episode. Agents usually take several time steps and episodes to learn to solve a task.

→ The sum of rewards collected in an episode is called the return. Our goal is to maximise the return.

→ Continuing tasks are usually converted to episodic tasks by limiting the number of time steps the agent can interact with the environment for.

Markov Decision Processes (MDP's)

→ First, in MDP's the states are fully observable, i.e S_t^e and observation are the same.

→ If the agent cannot fully see the internal state, we have a Partially Observable MDP (or POMDP).

Defⁿ: A state S_t is Markov $\iff P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \ldots, S_t]$

→ This means that, when we are in a Markov state, the future is independent of ~~the~~ our past and depends only on the present.

→ If action is included, then the definition changes slightly:

S_t is Markov $\iff P[S_{t+1} | S_t, A_t] = P[S_{t+1} | S_t, A_t, S_{t-1}, A_{t-1}, \ldots]$

→ RL agents operate under the assumption that each state it sees is Markov, so it is necessary the agent is fed the right number of "useful variables".

ex: Agent trying to learn to land a spacecraft must be given the spacecraft's velocity and acceleration ~~and~~ along with its location. Only location will not be sufficient.

→ However, the more the variables you feed the agent, the longer it takes for the agent to train. On the other hand, if you feed too few variables, it is likely the information is not sufficient.

Notation:

→ State space of MDP will be denoted S^+.

→ A subset of S^+ denoted S^i will be the set of all states the agent starts training. S^i is the set of initial states.

(A probability distribution is usually fixed on S^i for the agent to start each episode)

→ let $\underline{S}$ denote the set of all ~~terminal states~~ non terminal states.

→ A terminal state is a unique state from which the agent can't leave if it enters, i.e all transitions from a terminal state lead to itself.

→ In general, the set of actions available to us ~~in a State~~ ~~s~~, is dependent on the state we are in. So, let A denote the function that takes state s as input and returns the set of actions we can take in that state.

So, in state s, we have A(s) actions available to us.

→ When we are modelling the environment, we know the actions available in each state. Agents can select from these actions deterministically or stochastically.

Transition Function:

→ The way the environment changes as a response to actions is referred to as the state-transition probabilities, or simply the transition function, denoted $T(s, a, s')$.

→ $T(s, a, s')$ represents the probability of going to state s' when we take action a at state s.

→ Obviously, given a state s and an action a, ~~~~ we will land in one of the states of S^+, which, in terms of probabilities is $\sum_{s' \in S^+} P[s'/s,a] = 1 \;\; \forall\, s \in S$ and $\forall\, a \in A(s)$

Reward Signal: Denoted R.

→ R is a function that maps a transition tuple s,a,s' to a scalar. In other words, R is giving a signal of goodness to transitions.

→ Most problems have atleast one positive signal. (In general, positive signal is for "good transitions" and negative signal for non favourable ones)

ex. Winning a chess match

→ However, reward can be negative and can be seen as a cost, punishment or penalty.

→ The reward function, in the most general form is denoted $R(s,a,s')$ but we can also make it $R(s,a)$ or even $R(s)$ according to our needs, as sometimes we might need to reward just the state and sometimes a state action pair.

In terms of expectations we get

$$r(s,a) = E\left[R_t \mid S_{t-1} = s, A_{t-1} = a\right]$$

or $$r(s,a,s') = E\left[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\right]$$

Discount:

→ Recall that our main goal in a state is to maximise the total reward we get in the future, however this can be tricky in the case of continuing ~~rewards~~ tasks, as continuing tasks go on indefinitely.

→ For this reason, we want to give a higher weightage to rewards we obtain in the near future and we achieve this with the discount factor denoted γ.

→ γ is a positive real less than one, i.e $0 < \gamma < 1$

→ γ is a hyperparameter that we have to tune

→ let G_t denote the return of some episode starting from time step t.

· Then, if we do not use a discount factor

$$G_t = R_{t+1} + R_{t+2} + \dots + R_{t+n}$$

→ If discount factor is also taken into consideration,

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n}$$

→ It can be written as an infinite sum also, as

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$ (Note that it converges whenever R_ts are bounded)

→ A nice recursive relation also arises :

$$G_t = R_{t+1} + \gamma G_{t+1}.$$