Reinforcement Learning

RL is learning to take an action in a situation in order to maximize reward.

Assumption is there is some agent which is trying to accomplish a task. There is an enviornment in which the agent is operating.

Enviornment can be a maze where the agent is trying to find a way out of maze. It could be an agent trying to figure out the best way to manage investment portfolio. Here the enviornment is Stock market. Another example is the agent trying to decide among the pick up requests in cab-service scenario;

* An agent is an abstraction trying to behave optimally in an environment.

Difference between RL and SL

1. In SL there is no interaction with the environment and given a dataset we are required to predict the target. In RL we deal with processes where the agent actively interacts with the envisonment.
2. RL is an active learning where the agent learns only by interacting. While SL is passive learning where the agent learns only by extracting features from a given dataset.
3. In SL, there is a teacher which tells us whether the result for a given observation is correct or not. And the model can be improved by minimising errors term. In RL there is no teacher. The environment acts only as a critic, where it tells how good or bad the action is by giving rewards. It doesn't tell whether the action taken is the ultimate best or not

Note:

- The agent is any robot that is trying to learn the task.
- Environment is the world around it that gives it the feedback.

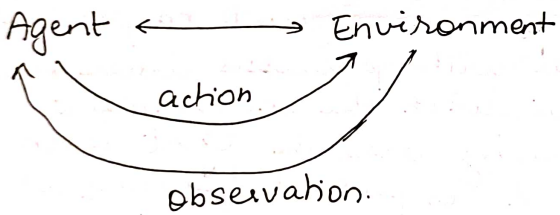
The agent doesn't have any control on the environment. The agent chooses the action but has no control over the consequences of the action.

ex in cab-service environment, when agent decides to pick a person from a particular place & drops him to another place; it is given that at the end of this action the person is dropped at the destination place. Here the agent can not wish that he will be at his house at the end of the action.

* Agent directly can not control the environment.

→ Humanoid Robot example – Read paper

Agent Environment Interaction



Agent observes the environment, then takes an action which he thinks is good. After taking the action, the enviornment tells him how good that action was in the form of numerical rewards; and presents a new observation to the agent.

Eg A student learning to maximise grades in his training He has goades of the exam that happened two weeks back. He observes the subject in which he scored lower. And then studies (action) only for those subjects After a week he goes through the exams again. His grades improved and now only in one subject his marks

are little less than the average. So, the action of studying had a positive consequence as his marks increased. Apart from this, he observes the subject in which he scored low marks - that becomes his new observation.

Reward, here is increase in marks. Reward only tells how well we are carrying out task. It does not guarantee that this is the best action.

There are two types of tasks

Reward — property of environment (weak signal)

Continuous

- tasks that do not have a definite end

eg. learning to walk, controlling a chemical plant, driving a car, portfolio management in a firm

Episodic

- tasks that have a definite end.

eg. games; since at the end of the game the agent either wins or loses

State Vectors

When we make a series of observations about the environment we try to find out the properties of the environment to plan/decide next course of action.

The set of things that we observe about the environment is called State of the environment.

State is the representation of the environment at any point of in time. The environment will give all the signals but how relevant those signals are for the agent to take an action is what we have to decide.

State vector is a list of features that help the agent to take an action. For each RL problem, state vector would be different.

state is the representation of the environment. Perhaps the environment would have a lot of things but the state that we want to take will determine which parameters in the environment really matters to us.

State vector is a collection of observations that a agent believes is adequate for the agent to carry out the task in which he is interested in.

Reward is in the form of number.

Examples of State Vectors, Actions & Rewards

1. Investment Management Portfolio
   State vector: amount of cash, % investments in mutual funds, equity, FDs, etc
   Action: Sell/Buy/Hold
   Reward: Profit/Loss

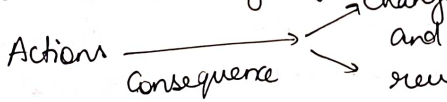
2. Cab Driver Problem
   State Vector: location, time, day
   Action: Service a request
   Reward: Money earned from a trip

Objectives Of RL Agent

- The objective of episodic tasks is to find out sequence of actions that will make majority of episodes successful.
- The objective of continuous tasks is to break it into multiple episodes & then find out actions that maximise the average rewards earned from those episodes
  eg managing portfolio of a client, we can break it down into episodes means we can have checkpoints like after every month the agent will give report as a result of the transaction. Objective is to maximize average returns

Actions & Policy

Action is something I do to interact with the environment
It results in change of state and/or reward.



Agent needs to learn about the environment before it behaves in an optimal manner. Learning means that the agent interacts with the environment by trying out different actions & understanding their consequences.

Memory of action and consequences is called knowledge-base or history for an RL agent.

Now the agent can look up in its knowledge base and see which action leads to the best consequence when in a given state. We can also build a mathematical model of the environment rather than storing all possible pairs.

This works good where there are very few states and actions. In more realistic situations, it is difficult to explore all possible states and actions and therefore makes it difficult to build a knowledge base or a model.

Policy - is a set of rules which helps the agent decide the action that it should take in a given state such that the agent can maximise its rewards in the long run. There are two types of policies:

1) Deterministic policy : $\pi(s) \rightarrow$ action (given state what action agent will take) (maximise reward)

2) Probabilistic policy: $\pi(a|s) \rightarrow$ distribution of over actions given a state / prob of taking action from state

* The objective of agent is to learn these policies eg for a novice doing investment portfolio management, the policy could be: whenever the stock price reaches a certain threshold, he will sell all the stocks. This is deterministic case. He has fixed the action for a state.

A probalistic policy could be: whenever the stock prices reaches a certain threshold, sell the stock 60% of the time, retain 35% of stocks and rest buy the shock.

* Policy takes into account both the state in which the agent will be after taking an action & also the reward he will get as a result of the action. By observing these over a period of time in the history, we arrive at a bunch of rules that will maximise the rewards in the long run.

Exploration & Exploitation (Reason why stochastic policy is better)

↓

choosing the actions that we have chosen so far that are giving best rewards. This works fine when we have actually explored the entire state and action space of your problem. This is actually unrealistic.

This is practically not possible (there will be a better way out).

→ ex travelling from home to office using same route but one day because of jam you explored new route & found better way to reach to office.

Exploration-Exploitation Tradeoff:

Exploiting an action is fine if you have exhaustively explored all the actions from a given state. But, this is generally not the case in real-life situations. In most scenarios, we would have explored only a small fraction of all possible actions. What if there exists an action that can get you a lottery? Wouldn't you go exploring more? But at the same time you also don't want to lose out on the benefits of the current action, in case you don't find good options while exploring.

So to handle this problem, a small window of exploration is set.

→ Exploitation refers to agent's tendency to make decisions based on its current knowledge of the environment. For ex an agent that is exploiting its knowledge would make decisions that are likely to result in high rewards based on experiences it has had so far.

Exploration refers to agent's tendency to try new things & explore the environment in order to gain new knowledge. eg in agent that is exploring would take actions that it hasn't taken before, even if they might result in lower rewards, in order to learn more about the environment

used to map patient vital signs (eg blood pressure, heart rate) to a probability distribution over possible treatment options, in finance can be used to map a set of economic indicators to a probability distribution over different investment strategies.

Markov State

Markov assumption :-

Given the current environment state & the actions taken, the future can be predicted. It is independent of what happened in the past.

1. Investment Management Portfolio
   Markov state : (current portfolio, market trend)
2. Process Control System
   Markov state : (current absolute values of pressure & temp, rates of change of pressure & temp over time)
3. Cab Driver Problem
   Markov state : (location, timestamp)

→ The action leads to a change in state & possibly generates a reward. One bruteforce way to learn a policy is to actually remember all the possible pairs of state, action & reward. But that is often not feasible. For example, in a game of Chess, this set may comprise of million possible combinations. Therefore in the RL problem, we make a markovian assumption. The markov assumption states that the current state contains all the necessary information about all the past states the agent was in and all the past actions the agent took. It assumes that the current state is sufficient for taking the next action.

We can consider markov state as some func^n of knowledge base that captures all relevant info from the knowledge base. And once 'markov state' is known, the knowledge base can be thrown away.

Markovian assumption

Chess game - Lets make an assumption that the policy is figuring out the best move to make given the current board position. Assump^n that we are making is that it really doesn't matter how the 2 players arrived at this board pos^n. We are not looking at the sequence of moves for both the players that led to this specific board position. Policy is based on only the current board pos^n. We don't look at the history because we believe that how we arrived at this pos^n doesn't matter to us, we can figure out the evolution of the game by looking at the board pos^n as it stands now.

→ Simultaneous games played by grand master, 100 odd people sitting with board in front of them, then this grand master is moving from one board to another & making the move. The move that the grandmaster is making is dependent on what the person sees the board at that time. There is something else also when the grand master is making a move; its not only the board pos^n the grand master is also trying to assess what the intent of the opponent was. The board pos^n alone is not making the point, we are also trying to figure out the strategy of the opponent that leads to the current board pos^n

ex

① Investment Scenario - Given portfolio & market cond^n we need to decide whether to buy or sell the stock. Here we are a little bit worry about the history of the market, we are interested in the trend. This can be turned into markov scenario; include trend as a part of state vector

② Cab driver - pure markov assump^n, holds naturally.

Deterministic Policy $\pi(s) \rightarrow a$

Given a particular state, a deterministic policy will always produce the same action. Used in situations where the outcome of an action is fully determined by the state of the system, & optimal action is known with certainity.

eg In healthcare, a deterministic policy can be used to map patients vital signs to a specific treatment plan, in Robotics a deterministic policy can be used to map sensor readings (eg the position of an object in the robot's field of view) to specific action (eg move robot's arm to pick the object), in Traffic control system, a deterministic policy could be used to map the current traffic conditions (eg the number of cars on road) to specific actions (eg change the timing of traffic lights to reduce congestion), chess game.

Stochastic / Probabilistic Policy $\pi(a|s)$

maps probability distribution over actions. Given a particular state, a stochastic policy will output a set of probabilities for each possible action, indicating the likelihood of each action being taken. Used in situations where the outcome of an action is uncertain or there are multiple optimal actions.

eg in Robotics, a stochastic policy can be used to map sensor readings (eg the position of an object in robot's field of view) to a probability distribution over actions (eg move robot's arm to pick up the object with certain probability, move robot's arm to push object away with another probability), in weather forecasting a stochastic policy could be used to map current weather cond^{s} (eg temperature, humidity, speed, wind) to a probability distribution over possible future weather states (eg Sunny, rainy, Cloudy), in healthcare could be

Deterministic Policy $\pi(s) \rightarrow a$

Value Func^n.

→ To formally arrive at RL prob.
few more concepts needs to be known.

we know Policy: $\pi(a|s)$ → distribution of actions given state.

Value of state: $v(s)$ → how valuable the state is

Lets takes an example - In a cab service enviornment the cab service at 8 pm; what is the best place a cab be in so that the cab is assured of getting a passanger for a long trip. The answer might be airport There is an inherit value associated with being at airport at 8 pm. When we are talking about valuable state, we are not talking about the action. In all MDPs we can associate an intrinsic value with the state. Irrespective of what action we take at the particular moment, the state will still be valuable. That is the value function is all about.
In investment, there may be an inheritant value associated with some stocks, some stocks may be very valuable irrespective of whether we are buying or selling the stocks.

→ Maze example.

this state is more valuable

Optimal Policy (Best)

A policy π^* is called optimal only if $\forall \pi : \pi^* \geq \pi$

For every possible policy π^* is atleast as good as π.

$\pi \geq \pi'$ (comparing 2 policies)

Policy π Policy π'

π^* → for all the given policy, this is the optimal policy

Model of the Environment

Given: model of the environment

Find the optimal policy π^*

Policy is for a particular task to be carried out, rewards will also be specific to this task

We are looking for something which will tell us about the consequences of taking an action. The agent can measure the consequence of the action in the form of reward. This is always going to be stochastic, we can't guarantee exactly the reward & state after taking an action given a state.

We can define model as:

Probability of finding at state s' & reaping some reward r conditioned on taking an action a from a particular state s

$$p(s', r \mid s, a)$$

- s': observed consequence
- s: current state
- a: action

The agent is taking an action given a state & the environment will throw agent in a new state & give a reward. The agent has no control over the consequences, the agent has control on the action that he takes.

→ This is the model of the environment.

In most of the applications/practical problems we don't have an explicit model like this $\boxed{p(s', r \mid s, a)}$ for the environment

The agent will never be able to tell that this much will be the reward by taking an action in a given state because there could be multiple actions that the agent can take in a state. But we still assume that there is a model of this form $p(s', r \mid s, a)$, we may not know what it is but if we want to describe the behavior of the environment as responses to the action then this is the way we are going to describe it. So if we don't have an explicit model beforehand we can estimate the model $p(s', r \mid s, a)$ from observations. So agent takes

action from states & observes the consequences in the form of new state & reward & build a huge body of experiences of taking specific actions in specific states & then use this data to estimate consequences assuming that the environment has been using this model underneath.
The model could be implicit or explicit. Explicit is rare; in most cases model will be implicit. We assume that such model exists given that we need to find an optimal policy π^* either by explicit model or by implicitly inferring about the model from the observations agent makes by taking an action given a state
One broad division of RL methods is

- Model-based
  - explicit model
- model-free methods
  - → implicitly refer about the model from the observations

Basic Equations

1) State Value function
Let's start some basic equations relating:
policy:- $\pi(a|s)$ → gives distribution over possible actions given a state
state value :- $V_\pi(s)$ - value of state given policy π
Q-value :- $q_\pi(s,a)$ - state s for an action a

we need to find expected reward from state s following policy π
$R_\pi(s)$ is a random variable (holding the total rewards)

So $V_\pi(s) = E[R_\pi(s)]$

↳ it is called random because everytime when we start from state s following policy π we are not going to get the same rewards. We are talking about the long term reward. This reward will keep changing. We are interested in expected reward if we start from state s

If we denote actual reward we get as a random variable if we take an action a from state s we see that by definition

$$q_\pi(s,a) = \underbrace{E}_{\text{expected}}\left[R_\pi(s,a)\right]$$

So $$R_\pi(s) = \sum_{a\sim\pi(a|s)} \pi(a|s)\, R_\pi(s,a)$$

↑ pick a from this distribution.

For each of these actions there is a probability with which we pick an action. Whenever we pick an action we get a ~~If we~~ reward starting from the ~~old~~ state s and taking action a.

If we take expectation on both sides.

$$\boxed{V_\pi(s) = \sum_{a\sim\pi(a|s)} \pi(a|s)\, q_\pi(s,a)} \rightarrow \text{value func}^n \text{ associated with state } s$$

2) Action Value function

We need to create an equation for action-value func^n $q_\pi(s,a)$

Let's say ~~from state s and~~ that we start from state s and take an action a under the policy π and end up in a state s' and got an immediate reward r. The q-value is the expected total reward starting from the state s and take an action a (under the policy π). We can write:

$$q_\pi(s,a) = r + V_\pi(s')$$

where $V_\pi(s')$ is the expected future rewards we receive if we start from state s'.

But we don't know how the conditions will be at s', so typically we won't value our future rewards as much as current rewards. That is why we discount the future rewards with a factor of γ.

$$q_\pi(s,a) = r + \gamma V_\pi(s') = r + \gamma\left(\sum_a \pi(a|s')\, q_\pi(s',a)\right)$$

optimal Policy

An agent is trying to find the opt.

This is a nested equation, where $q_\pi(s,a)$ can be re-written as $r' + \gamma V_\pi(s'')$. So the discount factor will keep on multiplying as we keep expanding the terms.

The discount factor determines the importance of future rewards i.e. how much we value the future rewards

- A factor of 0 would mean that we don't care about future rewards at all and only the immediate reward matters. In such case $q_\pi(s,a) = r$
- A factor of 1 would mean that we ~~have~~ value the immediate and future rewards equally.

$$q_\pi(s,a) = r + V_\pi(s')$$

Another important feature of having a discount factor is for continuous tasks. When total rewards are calculated for continuous tasks the total reward may not converge. So the discount factor will keep the total rewards bounded.

To calculate the expected value of $q_\pi(s,a)$ we need to have multiple episodes to account for all possible combinations of s' and r.

The same state-action pair can yield very different (successive state-reward) pairs. We need to average over all these combinations to get a sense of value of performing an action in a particular state.

Assume that the model of the environment is $p(s', r | s, a)$.

So the action-value function will be the expected reward over all possible combinations of s' and r.

To calculate this expected reward, we take the weighted average of the reward over all possible s' and r. Thus,

$$\boxed{q_\pi(s,a) = \sum_{s'} \sum_{r} p(s', r | s, a)\,(r + \gamma V_\pi(s'))}$$