Reinforcement Learning-1

Photo by Franck V. on Unsplash

#REINFORCEDSERIES

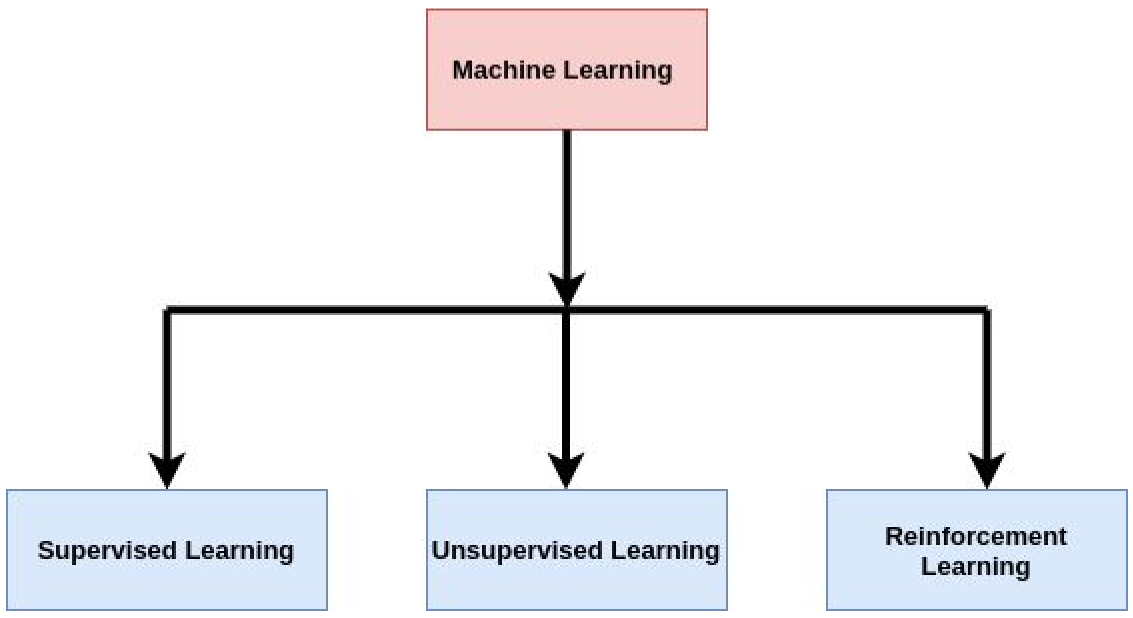
Understanding Reinforcement Learning- I

What is Reinforcement Learning?

“Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.”

Reinforcement learning currently has turned out to be a very important topic of research and has found important applications in several domains. In the reinforced series we are going to take a look at all the concepts of Reinforcement Learning and understand the basic principle behind their working, with applications. This is the first article of the series, that aims to give an introduction to the topic and explain the basic terminologies associated with reinforcement learning.

Divisions of Machine Learning



Machine Learning mostly contains three types of problems, as shown in the above figure.

Why Reinforcement Learning?

Let’s explore the types to find out.

Supervised Learning learns from a set of labeled examples. From the instances and the labels, supervised learning models try to find the correlation among the features, used to describe an instance, and learn how each feature contributes to the label corresponding to an instance. On receiving an unseen instance, the goal of supervised learning is to label the instance based on its feature correctly.

Unsupervised learning deals with data instances only. This approach tries to group data and form clusters based on the similarity of features. If two instances have similar features and placed in close proximity in feature space, there are high chances the two instances will belong to the same cluster. On getting an unseen instance, the algorithm will try to find, to which cluster the instance should belong based on its feature.

From our above discussion, we can get an intuition, both the processes are single instance-single prediction based processes. So, they can be called single decision processes.

Having explored the other two topics, it is evident that any of the two algorithm types can not be used to reach a goal state, of multiple decision processes like a game. To play a game, we need to make multiple choices and predictions during the course of the game to achieve success, so they can be called a multiple decision processes. This is where we need the third type of algorithm called reinforcement learning algorithms. The class of algorithm is based on decision-making chains which let such algorithms to support multiple decision processes.

The whole basic idea of reinforcement learning is based on Markov processes. So, let's study them in detail.

Markov Process

Markov Assumption:

Markov assumption states that the probability P of an event X at time t+1 is dependent only on the behavior of the event X at time t and independent of the behavior of X on time t=0,1….t-1 i.e, P(X (t+1)) is dependent only on X(t).

So, mathematically:

$$P(\,X(t+1)\mid X(t)\,) = P(\,X(t+1)\mid H(t)\,)$$

Where $H(t) = \{X(0),\ X(1),\ X(2), \ldots\ldots\ldots\ldots\ldots X(t)\}$

H(t) represents the history of the steps of the process. The equation represents the probability of X(t+1) given X(t) is equal to the probability of X(t+1) given all the history of al the steps of the process.

Markov Assumption presents a memoryless approach.

Any process or system that satisfies Markov assumptions is termed as Markov Process.

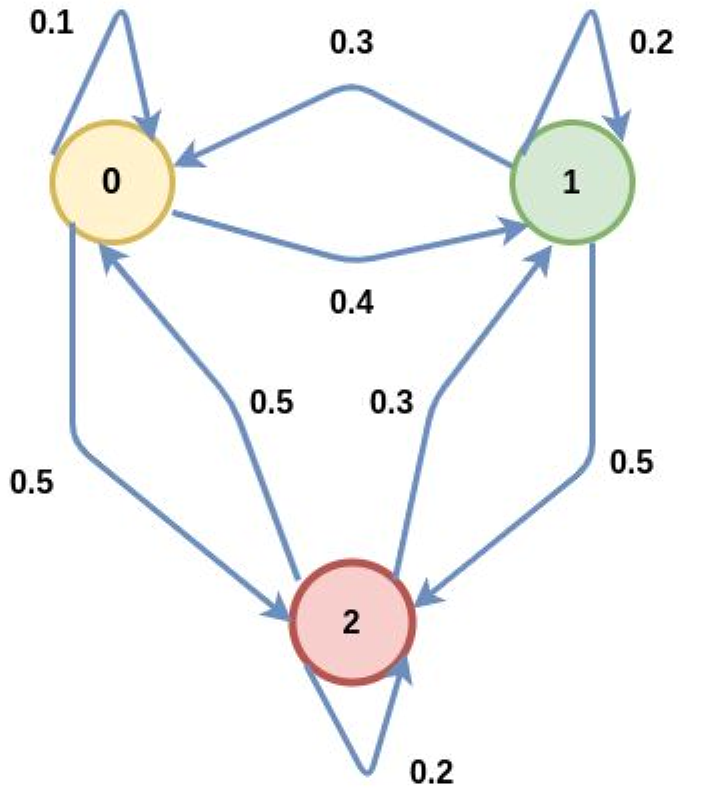
Markov Chain: ***It is a stochastic or probability-based model that consists of a sequence of events. The probability of each event occurring depends on only the state attained by the previous event.*** When we apply Markov property to a random sequential process a Markov chain is obtained.

A Markov process or Markov Chain is represented by a tuple (S, P).

- **S represents the set of states and**

- **P represents the state transition probability matrix. P(s',s) gives the probability that the process will achieve the state s' at t+1 if the process was at state s at time t.**

For instance:



If the above diagram describes a Markov process.

S-> [0,1,2]
P-> [[0.1, 0.4, 0.5], [0.3, 0.2, 0.5], [0.5, 0.3, 0.2]]

P is given as

$$[\,[\,P(X(t+1)=0\,|P(X(t=0))\ \ P(X(t+1)=1\,|P(X(t=0))\ P(X(t+1)=2\,|P(X(t=0))\,]$$
$$[\,P(X(t+1)=0\,|P(X(t=1))\ \ P(X(t+1)=1\,|P(X(t=1))\ P(X(t+1)=2\,|P(X(t=1))\,]$$
$$[\,P(X(t+1)=0\,|P(X(t=2))\ \ P(X(t+1)=1\,|P(X(t=2))\ P(X(t+1)=2\,|P(X(t=2))\,]\,]$$

Given:
P(X0)= 0 ->0.3
P(X0)= 1 ->0.4

P(X0)= 2 ->0.3

The above-given equation state the probability that the process will start from 0 i.e, the state at t=0 is state 0 is 0.3, and so on.

P is a matrix, it is horizontally stacked due to editor constraints.
For state 0, it reaches state 0 with 0.1 probability, state 1 with 0.4 probability, and so on.

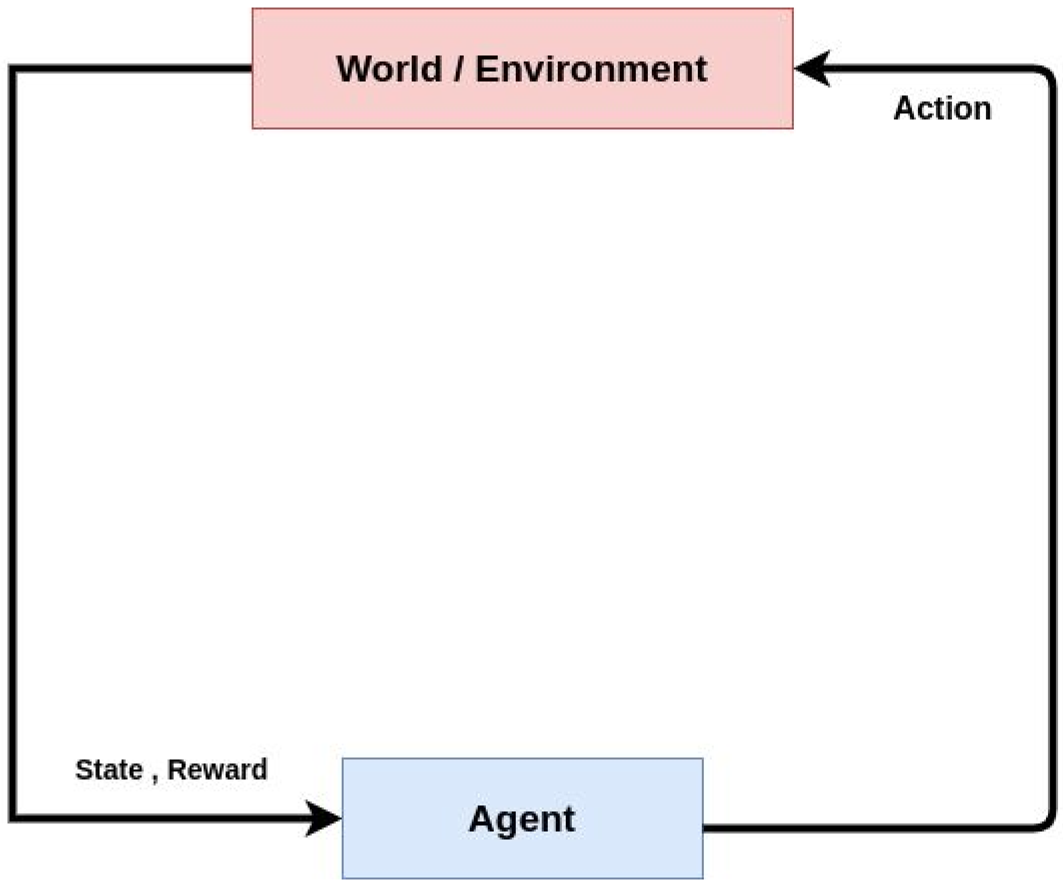
If we pick a sequence, say 210. According to Markov Process, and chain rule,

$$P(210) = P(X0 = 2).\ P((X1 = 1) \mid (X0 = 2)).\ P((X2 = 0) \mid P(X1 = 1))$$

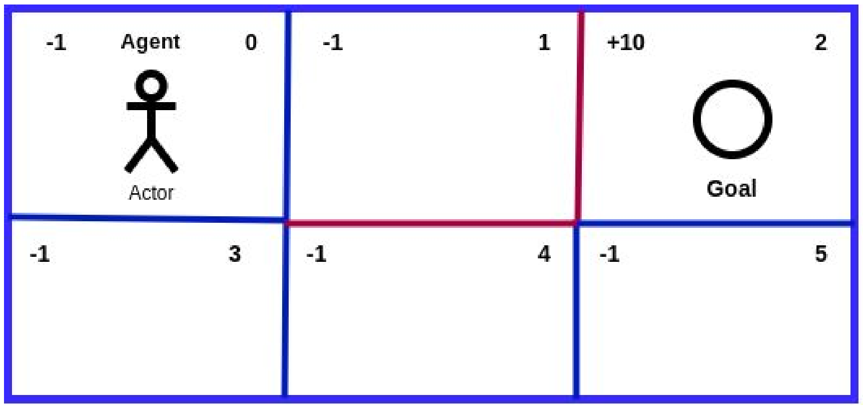
So, P(210) = 0.3 x 0.3 x 0.3= 0.027

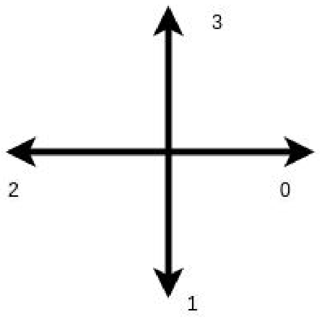
We will return back to this in the future shortly.

Reinforcement Learning Setting



We all know reinforcement learning is used for building games. So, let’s understand this with respect to a game. Let’s go with the most common one, say Grid World.





Say, the actor in the 1st box has to reach the 3rd box which has a ball he needs. So, the third box is the goal state. The red boundaries are blocks. I.e, he/she can't pass through those boundaries. Here the man is called an **Agent.** It moves from its initial position, to achieve its final goal. Now, when we play a game, the game offers several obstacles at several stages. The entire setting on which the game is played is called the **Environment.** To move from the initial position to the final or goal position, the agent has to show a movement at every stage of the game. These movements are called **Actions.** Now, in this case, we can only take 4 actions, similarly, every game typically has a definite set of actions, which we can take at each stage of the game. Each of the boxes or stages are termed as the **States of the Environment.** The states show the position of an agent with respect to the environment.

The agent keeps taking actions and changing states until it reaches the final state or goal state. But the question is, how would an agent know which state is the goal state, and which action sequence it should consider to reach the goal state to keep the number of steps to reach the goal optimally. As an indicator of the facts, each stage or state of an environment provides a **Reward** to the agent. The goal state in our case gives a reward of +10 and others give a reward of -1.

Reinforcement learning aims to train the agent to reach the goal state from any initial state. This is done by "Positively reinforcing" an action at a particular state that helps the agent to reach the goal state by providing the agent a higher reward on taking that action on that particular state. It is a way of signaling the agent to take the action at that particular step. The aim of the agent is to maximize the reward while reaching the goal state to satisfy optimality.

So, the agent starts from an initial state of the environment. It takes action given the state. With each action taken by the agent, the environment reacts to the action with a new state which may or may not be equivalent to the previous state and a reward. This describes the whole Reinforcement learning setting.

Types of Reinforcement Learning Agents

Model-Based: In this type, the agent tries to 'model the environment'. The agent captures the features and behaviors of the environment and its states. Using the observed, the agent tries to replicate the original environment and tries to create a virtual model with similar behaviors as the original one. The agent instead of interacting with the original environment interacts with the model, to stimulate future movements, actions, and responses. They are fast because they actually do not interact with the environment, so they do not need to wait for the response of the environment. They are risky because if the behaviors are wrongly interpreted during observations, the whole model will be wrongly created.

Model-Free: In this type, the agent directly interacts with the environment directly. They have policies and value functions. Though this type of learning is slow still reliable. The agent in search learning takes action, collects rewards, positive or negative, and updates the choice of states using reward function.

Here we will be talking about Model-Based Learning.

Components of Reinforcement Learning:

The basic components of Reinforcement Learning are:

1. **Model**: Representation of the world or environment.
2. **Policy**: It is kind of a mapping from states to actions. It tells which action should the agent take given a particular state.
3. **Value Function:** We have talked about Rewards and States. The value function is a function of the reward of the current state and also the reward of the future steps. It shows how important or valuable a state of the environment is to the agent. It is a measure of the maximum reward an agent can get in the future from a given state. More the value function, the more favorable the state. One thing to notice is the value function is the property of the state of the environment and not the agent. It just serves as a reference or guidance to the agent.

Components of a Model:

We have learned model copies of reflects the original environment. Until now we have seen for the environment part, it only provides the next state and reward obtained. So, the model also must contain those pieces of information. The model has two main components:

1. **Transition Dynamics**: It provides information about the probabilities with which an agent reaches from one state to another.

$$P(S(t+1) = s' \mid S(t) = s,\ A(t) = a\ )$$

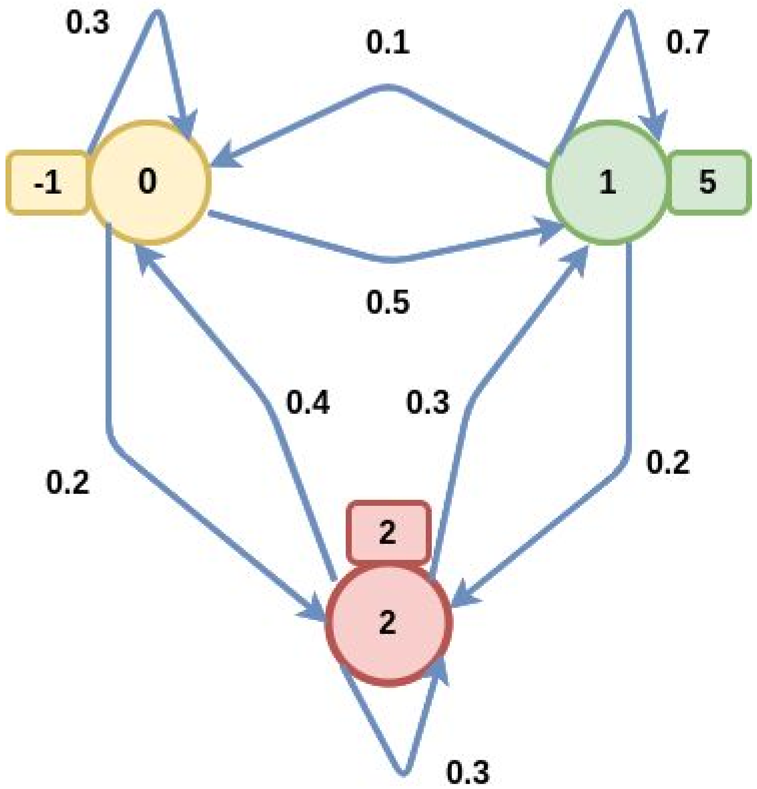
The equation gives the probability of reaching state s' at t+1 after taking action at state s at time t.

2. **Reward Dynamics:** It provides information about the rewards an agent receives when it takes action at a state.

$$R(\ S(t) = s,\ A(t) = a) \ = E\ (\ r(t)\ |\ S(t) = s, A(t) = a)$$

The reward for taking action 'a' on state s is the mathematical expectation of the same. The mathematical expectation is given as the probabilistic sum of rewards that can be obtained in the future if we take a particular action "a" at the state "s".

If we check by example,



If we consider this to be our environment,

Our transition dynamics looks like:

[0.1,0.4,0.5],
[0.3,0.2,0.5],
[0.5,0.3,0.2]

And our reward dynamics looks like: **[2,3,5]**

Policies

We have seen policy as a mapping from states to actions. It determines which action to take at which state of the environment.

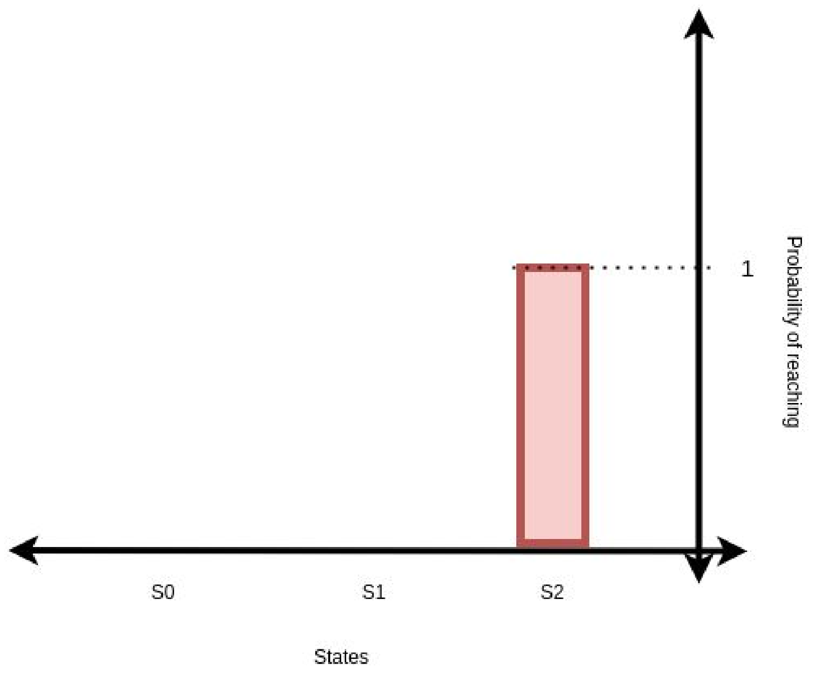
$PI : S -> A$

Pi is the policy, S is the set of states and A is the set of Actions.

Now, let’s consider two cases.

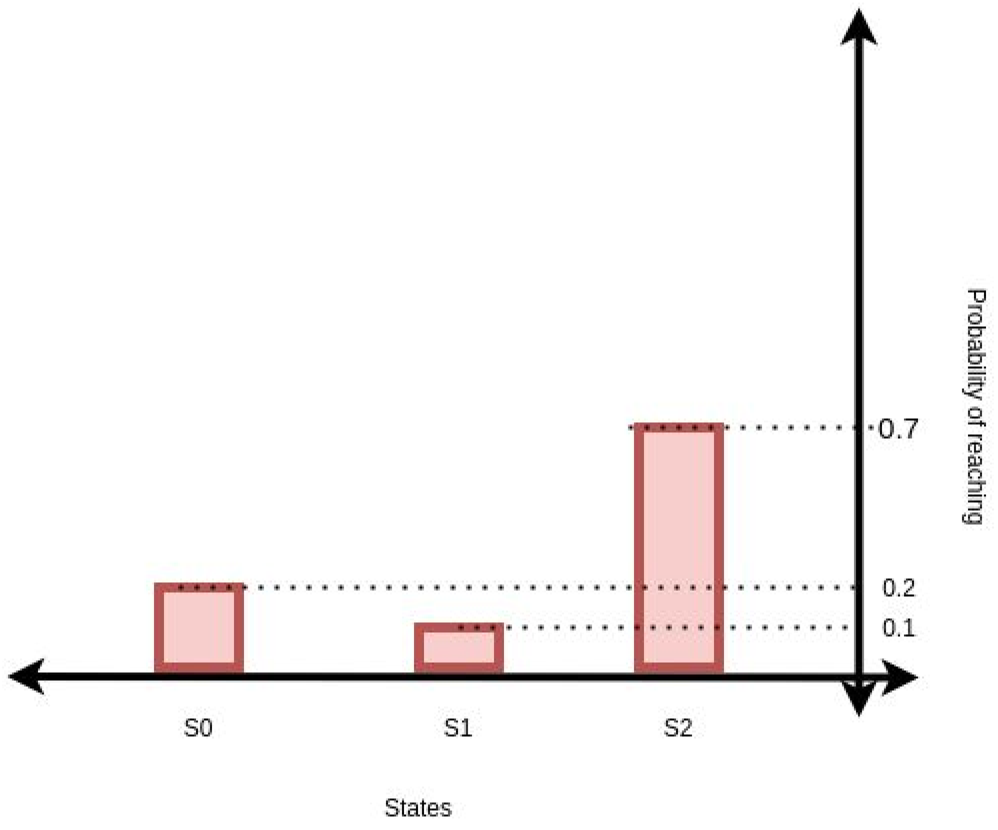
1. Say, the agent takes an action 1 at a state s1. The action should take the agent to state s2 according to the policy. The agent on executing the action at that state reaches state s2 on a confirmed basis i.e, with probability 1. Such a type of policy is called Deterministic policy.

   On transition, the agent reaches the desired state with a 100% probability.



2. Now, the second category states, on taking action 1 at state s1, the agent reaches state 2 with a probability of say 0.7. It reaches state s1 with probability 0.1 and state s0 with a

probability of 0.2. So, we are reaching the next step on a probabilistic distribution basis and not on a confirmed basis. Such a policy is called Stochastic Policy.

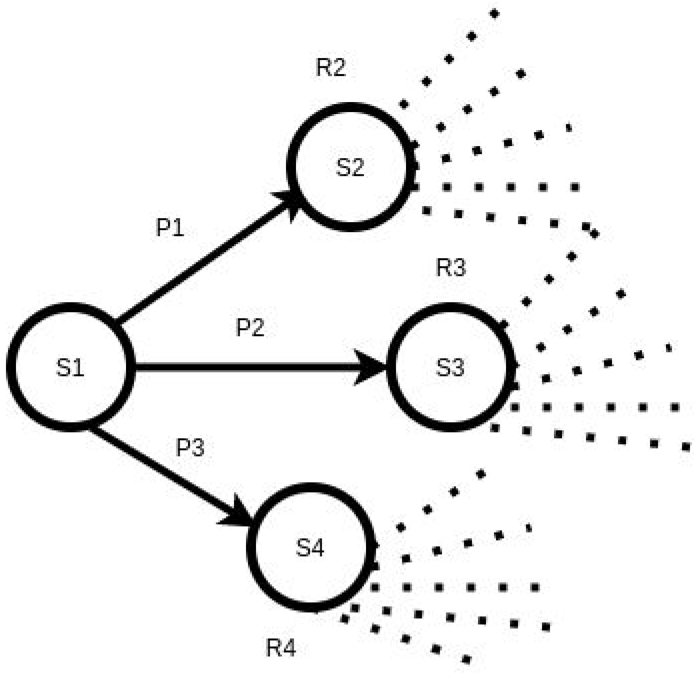


On transition, the agent reaches the desired state with a 70% probability. Such a policy represents a slip concept or reinforcement learning.

One thing to note is that the rewards achieved depends on the sequence of actions taken on states, i.e, the state-action mapping or policy. If the policy changes, everything will change.

Value Function

The value function is the expected discounted sum of future rewards for a particular policy. Mathematical Expectation is the probabilistic summation.



P1, P2 are the probabilities to reach S2, S3, S4 respectively. R2, R3, R4 are the already calculated Expectation of future rewards at S2, S3, and S4.

$$E(S1) = P1.R2 + P2.R3 + P3.S4$$

It is a discounted sum of expectation of future rewards.

So, it is given by:

$$V(PI) = E(PI)\ [\ R(t) + gamma * R(t+1) + gamma^2\ * R(t+2)\ + gamma^3\ *\ R(t+3)............ \mid S(t) = s\ ]$$

Where PI is the policy and gamma is the discount factor, R(t+1) is the reward at the state reached at time t+1 according to the policy pi.

Now, why the discount factor?

Let's see without a discount factor.

$$V(PI) = E(PI)\ [\ R(t) + R(t+1) + R(t+2)\ +\ R(t+3)............ \mid S(t) = s\ ]$$

If we talk about an infinite process, then the value explodes and also grows infinite. To protect this we use a discounted factor. The discounted factor is a value less than 1. So, as we go on increasing the powers of the factor, the value actually keeps on decreasing. After some time units, the rewards at that time step become insignificant. For example, if we consider gamma=

0.9. The rewards at and after (t+10) lose significance at time label t. So, we don’t face the infinity problem anymore.

The discount factor also serves as the weight balance between the current rewards and future rewards. Elaborately speaking, if the discount factor is 0.9 we focus on 10 future time step’s reward if it is 0.99 we focus on 50 future time step’s reward. So, it is key to a tradeoff that controls future rewards.

The value function of a state determines how good is a particular state of the environment is for the agent, or it shows how much reward can the agent while reaching the goal state on following that particular policy. The more the value function, the better the state, because the more the reward the agent will get.

Value functions are also used for comparing policies. The more the value functions for the states the better the policy.

Exploration and Exploitation

The agent has to reach the goal state from any initial state according to the theories of reinforcement learning. So, to do that, the agent needs to move around and observe the environment, i.e, it must know how the environment reacts to action at a given state. To accomplish this the agent takes random actions outside its policies and notes the response of the environment. This is called **Exploration**.

Now, if the agent keeps on exploring the agent will never obtain the optimality for reaching the goal state. For this, it will need to follow the policies established by the previous observations of the environment. So, here the agent is actually using the policies that are obtained by previously gathered experiences or gained knowledge. This is called **Exploitation**.

If the agent keeps on exploiting, the agent will never be able to know the entire state space. So, there is a tradeoff between exploitation and exploration. To achieve optimality we need to have both in equal amounts.

Evaluation and Control

Evaluation is the process of determining the value functions of the states of a particular policy.

Control is the optimization of value functions to find the optimal policy.

Conclusion

We have taken a look at the underlying principle of Reinforcement Learning and also seen the basic associated terminologies. Please check the full reinforced series to get a complete idea.

Thanks for reading!!!!.