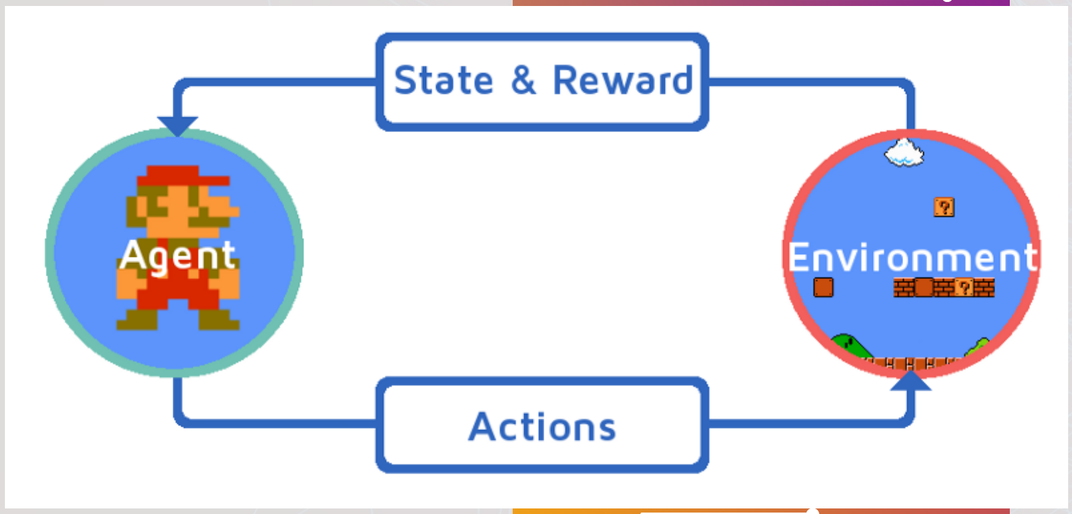
CFI

Reinforcement learning

COMPUTER VISION AND INTELLIGENCE

Table of content

About RL



Reinforcement Learning-RL is the science of decision-making. It is about learning the optimal behavior in an environment to obtain the maximum reward without a supervisor. The quality of actions is determined by the immediate reward they return and the delayed reward they might fetch. Since the agent learns without any supervisor or previous data, RL is a robust algorithm.

The Applications

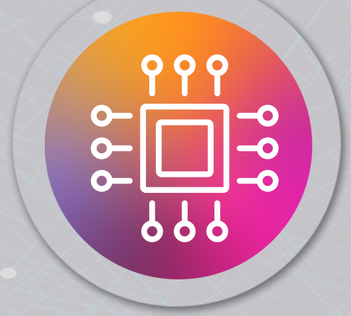
atleast some of them.....

Gaming

Using RL agents were able to learn various games such Go from scratch or any of the games from Atari

Industry

In Industry Automation, learning robots are used to perform tasks. For eg Google's Deepmind that is used to cool their data center



Autonomous Driving

Various deep RL papers have suggested the use of deep RL f for tasks of self driving such as path optimization, motion planning, dynamic pathing, etc.

Finance

Supervised learning models can predict future sales and stock prices but can not decide what to with them. RL agents on the other hands can decide whether to hold or sell. IBM has one such agent.

Basic Terminologies

So that nothing is gibberish later up.....

History

History is the sequence of actions taken and their corresponding observations and rewards up to a finite time 't' or till a terminal state is reached.

$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

Action

Anything done by the agent in the environment is referred to as a action. For example- an agent learning a maze moving up, down, right or left is an action. An agent learning space shooter in Atari moving left or right or shooting might be an action.

Rewards

Rewards are a scalar feedback signal from the environment for every agent's action. It indicates how well the agent is doing, and agents tend to maximize the cumulative reward.

Some more.....

State

A state is the summary of previous information used to determine what happens next. It is a function of the history of the environment.

$$S_t = f(H_t)$$

Environment State S_t^e

Environment's private representation of the state. It is usually not visible to agent. Even if visible, not all information in this state is useful for implementing an algorithm.

Agent State S_t^a

It is the agents internal representation of the state. It is whatever information the agent picks up and is useful for implementing an algorithm.

Information State

It contains all the useful information from the history.

Just a little more....

Policy

A policy is the agent's behavior. it is a map from the state to the action.

Deterministic policy: $a = \pi(s)$

Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

Value Function

Value function is a prediction of the future rewards. It is used to evaluate the goodness/badness of the states and therefore used in selecting the next action.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s\right]$$

Model

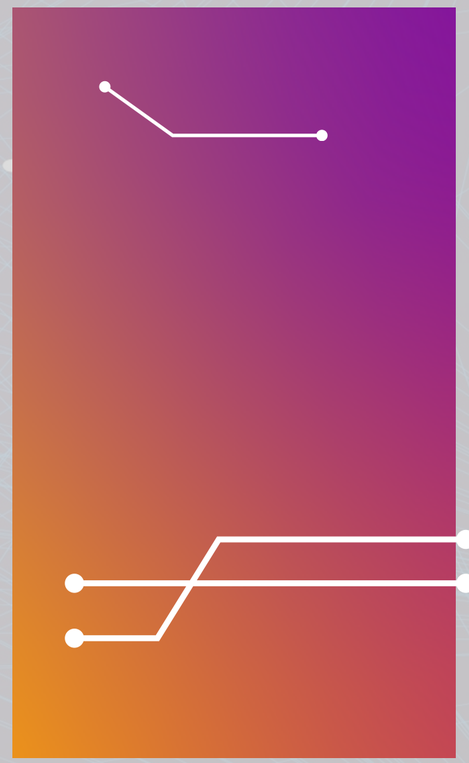
A model predicts what the environment will do next. There are 2 general types of models:

1. Transitions- predicts the next state given the previous ones.
2. Rewards- predicts the immediate reward.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Everything Markov

One of the most fundamental concepts that is crucial to learning how to make an RL model, from a very crude one to a complicated one.

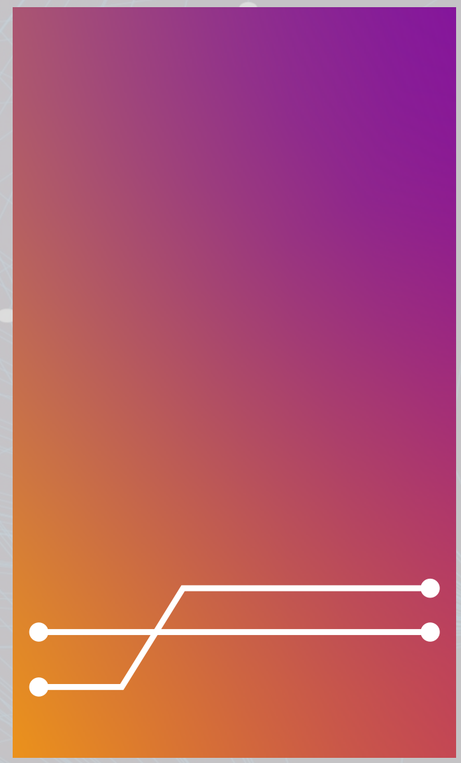


Markov Property

A state is markov if and only if the probability of the next state is only dependent on the current state and is same as the probability with respect to all the previous states.
In other terms, the future is independent of the history.

A state S_t is Markov if and only if

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



Markov Decision Process-MDP's

Markov decision process formally describes the environment for RL when the environment is fully observable i.e the current state completely characterises the process.

The state transition matrix $\mathcal{P}$ defines the transition probabilities from state S to all the successor states S'.

$$\mathcal{P} = \text{from} \begin{matrix} \\ \\ \\ \end{matrix} \overset{\text{to}}{\begin{bmatrix} \mathcal{P}_{11} & \cdots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \cdots & \mathcal{P}_{nn} \end{bmatrix}}$$

The state transition probability is the probability of going to state S' from current state S

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

The MDP is a tuple of a set of states S and the state transition matrix $\mathcal{P}$.

A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$

Markov Reward Process

A Markov reward process is a MDP with values.

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

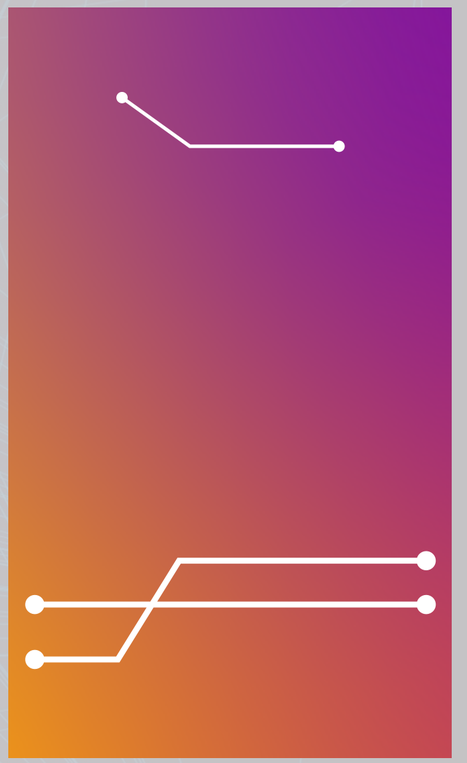
- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

A reward function for the state S at time 't' is expectation of the rewards at time 't+1'. Expectation in mathematical terms is the average outcome of a series of events with the odds being repeated.

Gamma is the discount factor. It is the present value of the future i.e how much we prefer future rewards to current rewards. It is usually between 0 to 1, 0 being extremely "short sighted" and 1 being "far sighted".

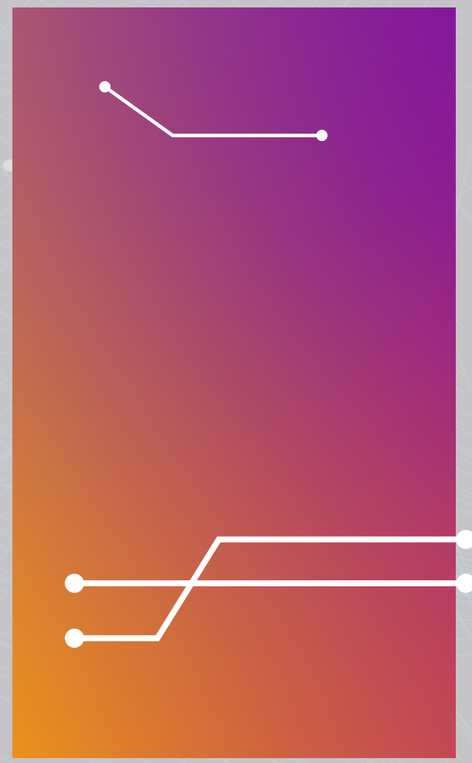
Why Discount?

One of the simplest reasons to discount the rewards is because it is mathematically convenient. It also avoids infinite returns in a Markov process cycle. One of the main reason is because the future environment is uncertain and therefore discounting helps us account for that uncertainity.



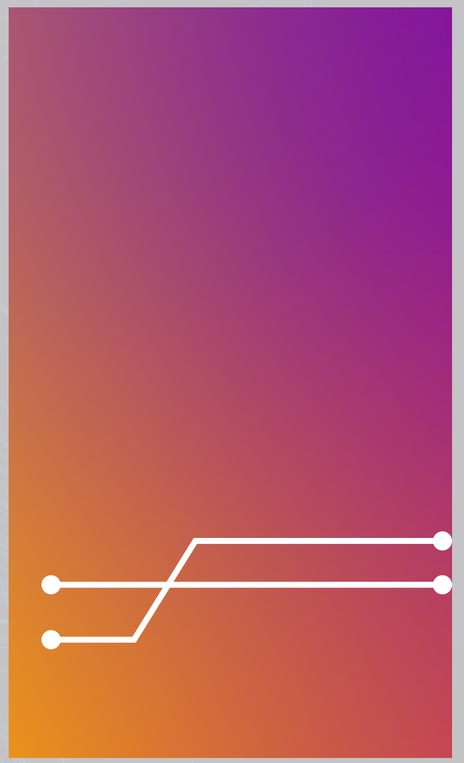
Q-Learning

Up ahead, we will see a basic learning table look-up Q-learning implemented in a primary environment, but before that .....



What is Q-Learning

Q-learning is the best example of off-policy learning. Here we use a behaviour policy to compute the actual policy to evaluate the value function or the action-value function.
In Q-learning we choose an action using our actual behaviour policy and also consider an alternative action based on the target policy that has been learnt till now and update the action value function.



Off-policy learning

Off policy learning is where we use a behaviour policy to evaluate the target policy. We sample from the behaviour policy.

$$\{S_1, A_1, R_2, ..., S_T\} \sim \mu$$

This has some advantages. For instance we can learn from other agents/humans. We can reuse the experience generated from previous policies instead of wasting them.

The Learning algorithm

We allow both the target policy and the behaviour policy to improve. The behaviour policy is epsilon greedy with respect to the action value function. The target policy is purely greedy with respect to the action value function.

$$\pi(S_{t+1}) = \underset{a'}{\text{argmax}}\ Q(S_{t+1}, a')$$

The final Q-learning algorithm comes out to be :

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$

The final Q-learning algorithm converges to the optimal action-value function. :