What Is Reinforcement Learning?

In machine learning, a common drawback is the vast amount of data that models need to train. The more complex a model, the more data it may require. Even after all this, the data we get may not be reliable. It may have false or missing values or may be collected from untrustworthy sources.

Reinforcement Learning overcomes the problem of data acquisition by almost completely removing the need for data!

Reinforcement learning is a branch of Machine Learning that trains a model to come to an optimum solution for a problem by taking decisions by itself.

It consists of:

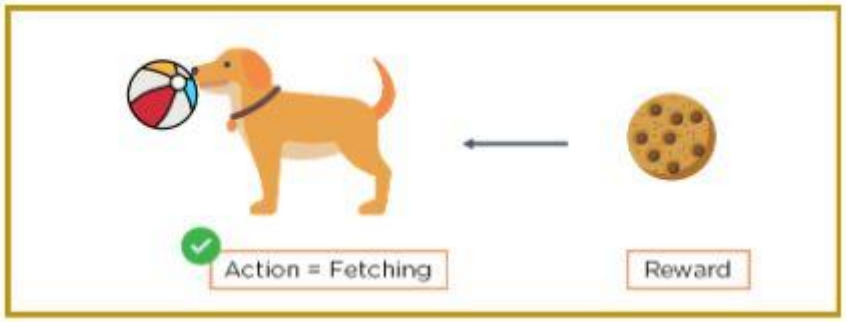
- An Environment, which an agent will interact with, to learn to reach a goal or perform an action.
- A Reward if the action performed by the model is bringing us closer to the goal/is leading to the goal. This is done to train the model in the right direction.
- A negative reward if it performs an action that will not lead to the goal to prevent it from learning in the wrong direction.

Reinforcement learning requires a machine learning model to learn from the problem and come up with the most optimal solution by itself. This means that we also arrive at fast and unique solutions which the programmer might not even have thought of.

Consider the image below. You can see a dog in a room that has to perform an action, which is fetching. The dog is the agent; the room is the environment it has to work in, and the action to be performed is fetching.

Figure 1: Agent, Action, and Environment

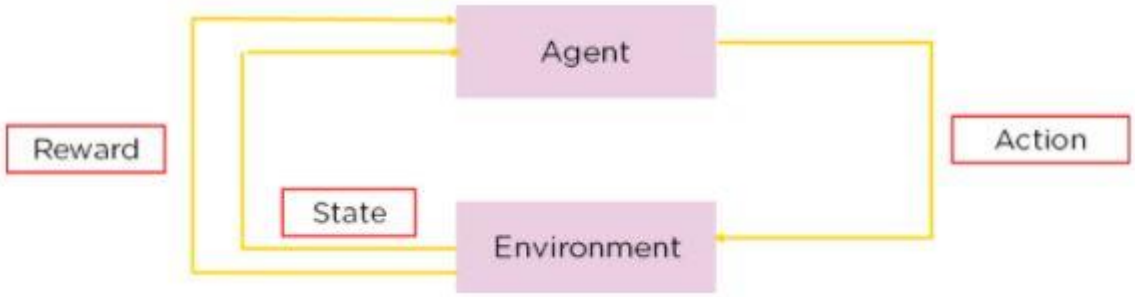
If the correct action is performed, we will reward the agent. If it performs the wrong action, we will not give it any reward or give it a negative reward, like a scolding.



Agent performing an action

What Is Q-Learning?

Q-Learning is a Reinforcement learning policy that will find the next best action, given a current state. It chooses this action at random and aims to maximize the reward.



Components of Q-Learning

Q-learning is a machine learning approach that enables a model to iteratively learn and improve over time by taking the correct action. Q-learning is a type of reinforcement learning.

Q-learning is a model-free, off-policy reinforcement learning that will find the best course of action, given the current state of the agent. Depending on where the agent is in the environment, it will decide the next action to be taken.

The objective of the model is to find the best course of action given its current state. To do this, it may come up with rules of its own or it may operate outside the policy given to it to follow. This means that there is no actual need for a policy, hence we call it off-policy.

Model-free means that the agent uses predictions of the environment's expected response to move forward. It does not use the reward system to learn, but rather, trial and error.

An example of Q-learning is an Advertisement recommendation system. In a normal ad recommendation system, the ads you get are based on your previous purchases or websites you may have visited. If you've bought a TV, you will get recommended TVs of different brands.

Ad Recommendation System

Using Q-learning, we can optimize the ad recommendation system to recommend products that are frequently bought together. The reward will be if the user clicks on the suggested product.

Ad Recommendation System with Q-Learning

Important Terms in Q-Learning

1. States: The State, S, represents the current position of an agent in an environment.
2. Action: The Action, A, is the step taken by the agent when it is in a particular state.
3. Rewards: For every action, the agent will get a positive or negative reward.
4. Episodes: When an agent ends up in a terminating state and can't take a new action.
5. Q-Values: Used to determine how good an Action, A, taken at a particular state, S, is. Q (A, S).
6. Temporal Difference: A formula used to find the Q-Value by using the value of current state and action and previous state and action.

How does Q-learning work?

Q-learning models operate in an iterative process that involves multiple components working together to help train a model. The iterative process involves the agent learning by exploring the environment and updating the model as the exploration continues.

Here are the two methods to determine the Q-value:

- **Temporal difference.** The temporal difference formula calculates the Q-value by incorporating the value of the current state and action by comparing the differences with the previous state and action.
- **Bellman's equation.** Mathematician Richard Bellman invented this equation in 1957 as a recursive formula for optimal decision-making. In the q-learning context, Bellman's equation is used to help calculate the value of a given state and assess its relative position. The state with the highest value is considered the optimal state.

Q-learning models work through trial-and-error experiences to learn the optimal behavior for a task. The Q-learning process involves modeling optimal behavior by learning an optimal *action value function* or q-function. This function represents the optimal long-term value of action *a* in state *s* and subsequently follows optimal behavior in every subsequent state.

What is a Q-table?

The Q-table includes columns and rows with lists of rewards for the best actions of each state in a specific environment. A Q-table helps an agent understand what actions are likely to lead to positive outcomes in different situations.

The table rows represent different situations the agent might encounter, and the columns represent the actions it can take. As the agent interacts with the environment

and receives feedback in the form of rewards or penalties, the values in the Q-table are updated to reflect what the model has learned.

The purpose of reinforcement learning is to gradually improve performance through the Q-table to help choose actions. With more feedback, the Q-table becomes more accurate so the agent can make better decisions and achieve optimal results.

The Q-table is directly related to the concept of the Q-function. The Q-function is a mathematical equation that looks at the current state of the environment and the action under consideration as inputs. The Q-function then generates outputs along with expected future rewards for that action in the specific state. The Q-table allows the agent to look up the expected future reward for any given state-action pair to move toward an optimized state.

How to Make a Q-Table?

While running our algorithm, we will come across various solutions and the agent will take multiple paths. How do we find out the best among them? This is done by tabulating our findings in a table called a Q-Table.

A Q-Table helps us to find the best action for each state in the environment. We use the Bellman Equation at each state to get the expected future state and reward and save it in a table to compare with other states.

Lets us create a q-table for an agent that has to learn to run, fetch and sit on command. The steps taken to construct a q-table are :

Step 1: Create an initial Q-Table with all values initialized to 0

When we initially start, the values of all states and rewards will be 0. Consider the Q-Table shown below which shows a dog simulator learning to perform actions :

Action	Fetching	Sitting	Running
Start	0	0	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	0	0
End	0	0	0

Initial Q-Table

Step 2: Choose an action and perform it. Update values in the table

This is the starting point. We have performed no other action as of yet. Let us say that we want the agent to sit initially, which it does. The table will change to:

Action	Fetching	Sitting	Running
Start	0	1	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	0	0
End	0	0	0

Q-Table after performing an action

Step 3: Get the value of the reward and calculate the value Q-Value using Bellman Equation

For the action performed, we need to calculate the value of the actual reward and the Q(S, A) value

Action	Fetching	Sitting	Running
Start	0	1	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	34	0
End	0	0	0

Updating Q-Table with Bellman Equation

Step 4: Continue the same until the table is filled or an episode ends

The agent continues taking actions and for each action, the reward and Q-value are calculated and it updates the table.

Action	Fetching	Sitting	Running
Start	5	7	10
Idle	2	5	3
Wrong Action	2	6	1
Correct Action	54	34	17
End	3	1	4

Final Q-Table at end of an episode

What is the Q-learning algorithm process?

The Q-learning algorithm process is an interactive method where the agent learns by exploring the environment and updating the Q-table based on the rewards received.

The steps involved in the Q-learning algorithm process include the following:

- **Q-table initialization.** The first step is to create the Q-table as a place to track each action in each state and the associated progress.
- **Observation.** The agent needs to observe the current state of the environment.
- **Action.** The agent chooses to act in the environment. Upon completion of the action, the model observes if the action is beneficial in the environment.
- **Update.** After the action has been taken, it's time to update the Q-table with the results.
- **Repeat.** Repeat steps 2-4 until the model reaches a termination state for a desired objective.

What are the advantages of Q-learning?

The Q-learning approach to reinforcement learning can potentially be advantageous for several reasons, including the following:

- **Model-free.** The model-free approach is the foundation of Q-learning and one of the biggest potential advantages for some uses. Rather than requiring prior knowledge about an environment, the Q-learning agent can learn about the environment as it trains. The model-free approach is particularly beneficial for scenarios where the underlying dynamics of an environment are difficult to model or completely unknown.
- **Off-policy optimization.** The model can optimize to get the best possible result without being strictly tethered to a policy that might not enable the same degree of optimization.
- **Flexibility.** The model-free, off-policy approach enables Q-learning flexibility to work across a variety of problems and environments.
- **Offline training**. A Q-learning model can be deployed on pre-collected, offline data sets.

What are the disadvantages of Q-learning?

The Q-learning approach to reinforcement model machine learning also has some disadvantages, such as the following:

- **Exploration vs. exploitation tradeoff.** It can be hard for a Q-learning model to find the right balance between trying new actions and sticking with what's already known. It's a dilemma that is commonly referred to as the exploration vs. exploitation tradeoff for reinforcement learning.
- **Curse of dimensionality.** Q-learning can potentially face a machine learning risk known as the curse of dimensionality. The curse of dimensionality is a

problem with high-dimensional data where the amount of data required to represent the distribution increases exponentially. This can lead to computational challenges and decreased accuracy.

- **Overestimation.** A Q-learning model can sometimes be too optimistic and overestimate how good a particular action or strategy is.
- **Performance.** A Q-learning model can take a long time to figure out the best method if there are several ways to approach a problem.