DS JULY 2022 Batch
Module 25 : Reinforcement Learning

Topics

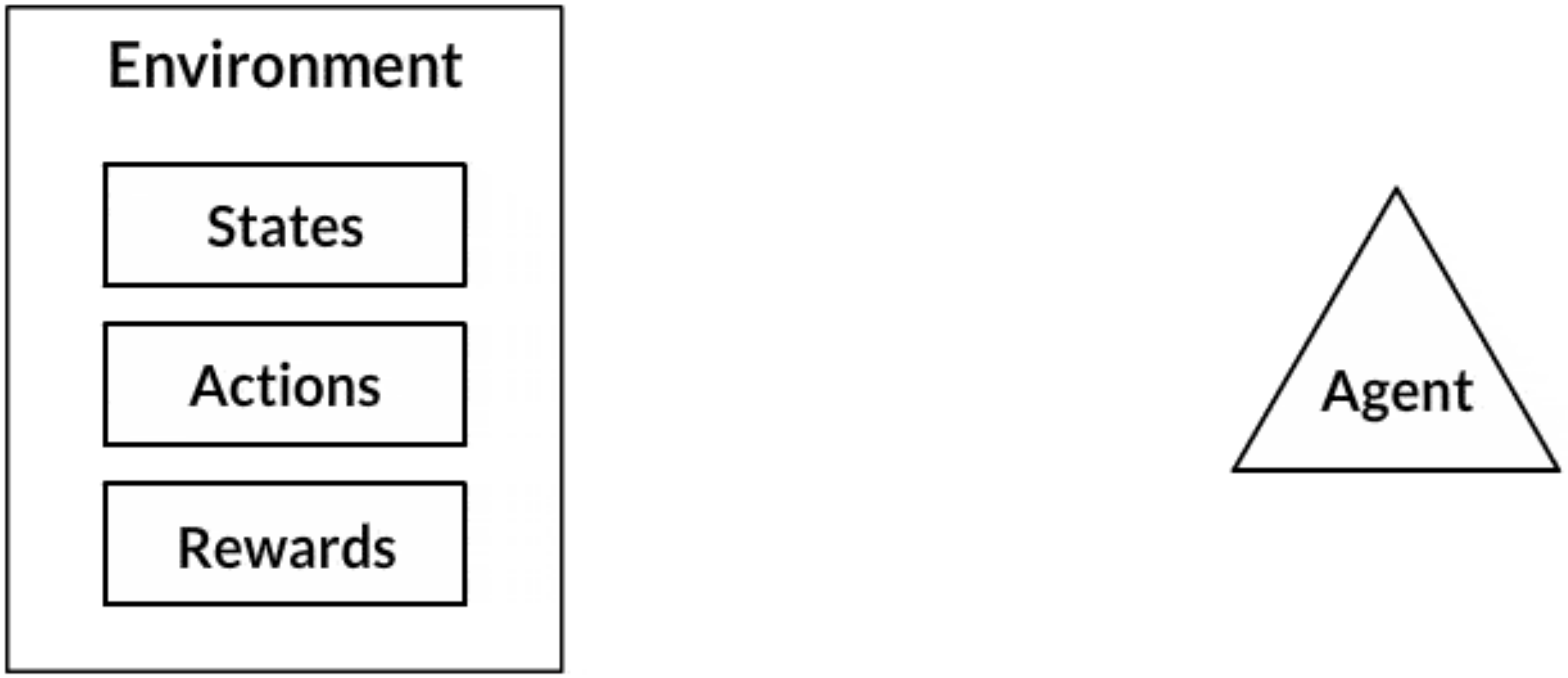
- Reinforcement Learning Introduction and Environments
- Different Algorithms Q learning, Deep Q
- OpenAI Gym tutorial

Reinforcement Learning

Reinforcement Learning

- Reinforcement Learning is a **feedback-based Machine learning** technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labelled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that **"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."**

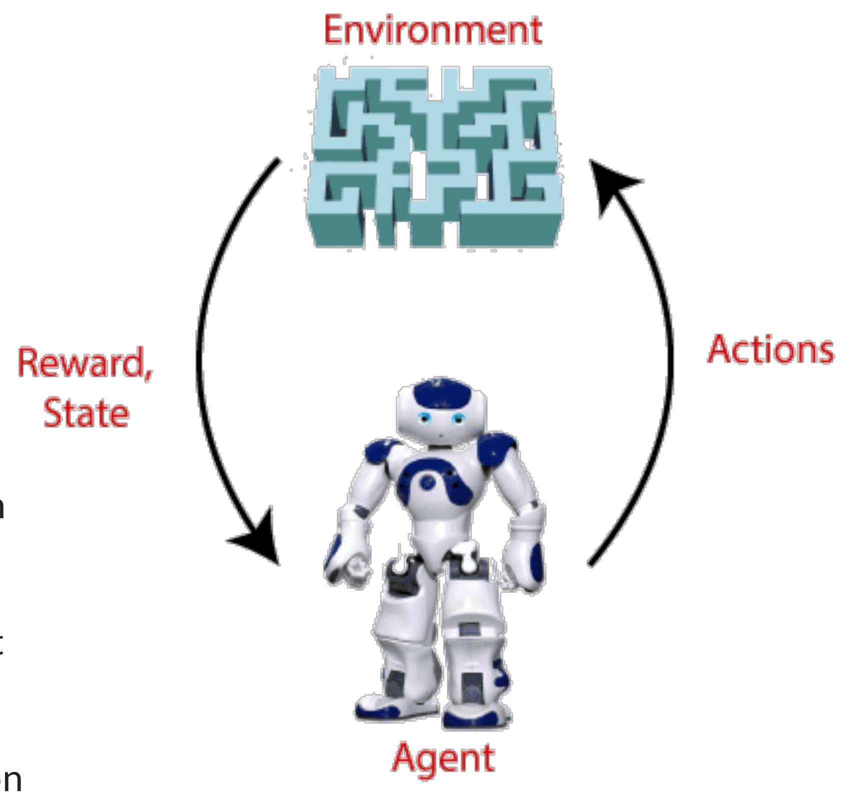
Reinforcement Learning



Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.

  Action(): Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action



RL : Elements

- **Policy:**
  - A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states.
  - A policy is the core element of the RL as it alone can define the behaviour of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:
  - For deterministic policy: $a = \pi(s)$

    For stochastic policy: $\pi(a \mid s) = P[A_t = a \mid S_t = s]$
- **Reward Signal:**
  - The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a reward signal. These rewards are given according to the good and bad actions taken by the agent.
  - The agent's main objective is to maximize the total number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

RL : Elements

- **Value Function:**
  - The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the immediate signal for each good and bad action, whereas a value function specifies the good state and action for the future.
  - The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.
- **Model:**
  - The last element of reinforcement learning is the model, which mimics the behaviour of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.
  - The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems with the help of the model are termed as the model-based approach. Comparatively, an approach without using a model is called a model-free approach.

RL : Types

- **Positive Reinforcement :** Positive reinforcement is defined as when an event, occurs due to specific behaviour, increases the strength and frequency of the behaviour. It has a positive impact on behaviour.
- Advantages
  - – Maximizes the performance of an action
  - – Sustain change for a longer period

  Disadvantage
  - – Excess reinforcement can lead to an overload of states which would minimize the results.

- **Negative Reinforcement:** Negative Reinforcement is represented as the strengthening of a behaviour. In other ways, when a negative condition is barred or avoided, it tries to stop this action in the future.
- Advantages
  - – Maximized behaviour
  - – Provide a decent to minimum standard of performance
- Disadvantage
  - – It just limits itself enough to meet up a minimum behaviour

RL : Working

- Refer Doc

Different Algorithms

RL : 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.
- 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.

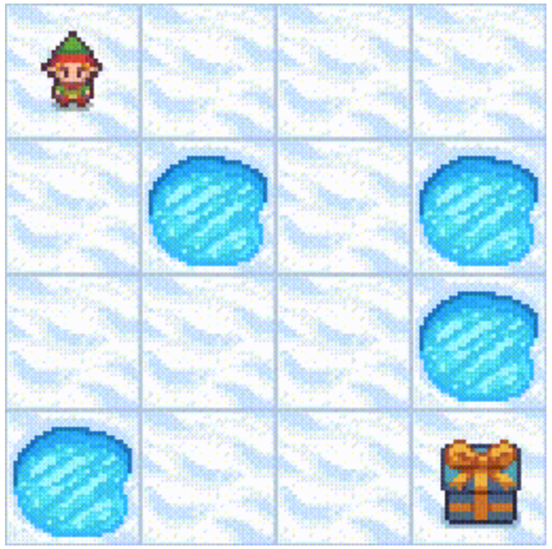
RL : Q- learning

- **Key Terminologies in Q-learning**

- **States(s)**: the current position of the agent in the environment.
- **Action(a):** a step taken by the agent in a particular state.
- **Rewards:** for every action, the agent receives a reward and penalty.
- **Episodes:** the end of the stage, where agents can't take new action. It happens when the agent has achieved the goal or failed.
- **Q(St+1, a):** expected optimal Q-value of doing the action in a particular state.
- **Q(St, At):** it is the current estimation of Q(St+1, a).
- **Q-Table:** the agent maintains the Q-table of sets of states and actions.
- **Temporal Differences(TD):** used to estimate the expected value of Q(St+1, a) by using the current state and action and previous state and action.

RL : Q- learning

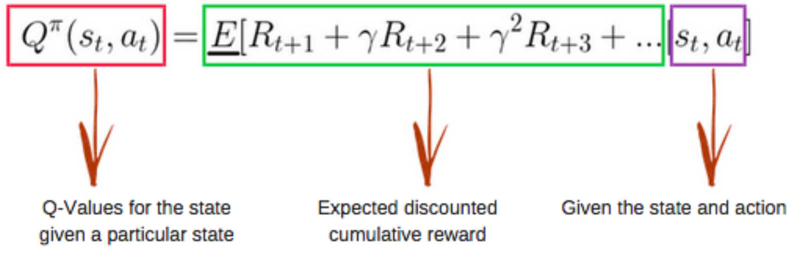
- **Q-learning : Working**
- Lets learn how Q-learning works by using the example of a frozen lake. In this environment, the agent must cross the frozen lake from the start to the goal, without falling into the holes. The best strategy is to reach goals by taking the shortest path.



RL : Q- learning

- **Q-Table :** The agent will use a Q-table to take the best possible action based on the expected reward for each state in the environment. In simple words, a Q-table is a data structure of sets of actions and states, and we use the Q-learning algorithm to update the values in the table.

- **Q-Function :** The Q-function uses the Bellman equation and takes state(s) and action(a) as input. The equation simplifies the state values and state-action value calculation.

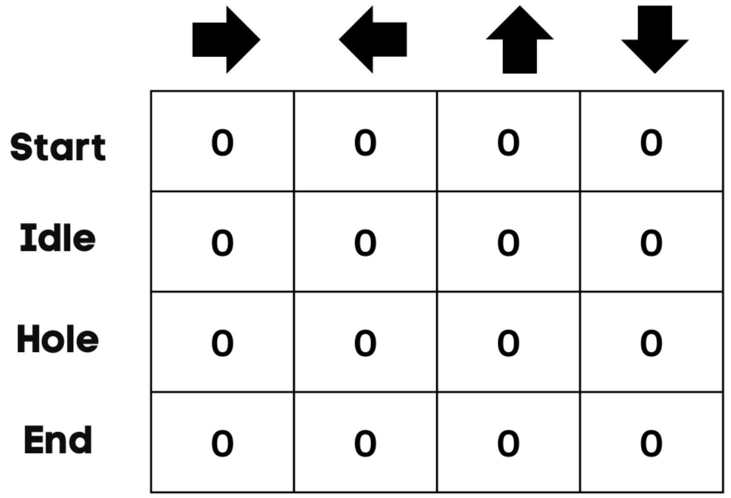


$$Q^{\pi}(s_t, a_t) = \underline{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid s_t, a_t]$$

Q-Values for the state given a particular state
Expected discounted cumulative reward
Given the state and action

RL : Q- learning

- **Initialize Q-Table**
- We will first initialize the Q-table. We will build the table with columns based on the number of actions and rows based on the number of states.
- In our example, the character can move up, down, left, and right. We have four possible actions and four states(start, Idle, wrong path, and end). You can also consider the wrong path for falling into the hole. We will initialize the Q-Table with values at 0.



	→	←	↑	↓
Start	0	0	0	0
Idle	0	0	0	0
Hole	0	0	0	0
End	0	0	0	0

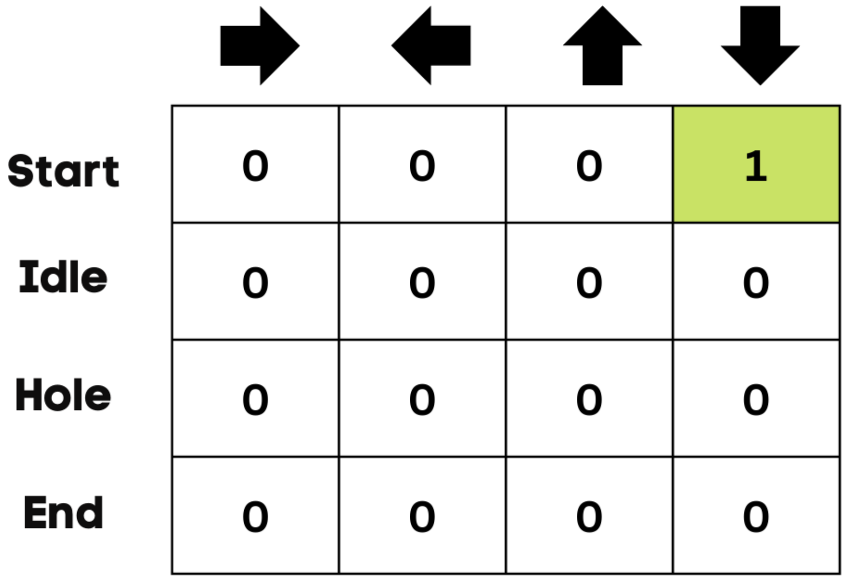
RL : Q- learning

- **Choose an Action**
- The second step is quite simple. At the start, the agent will choose to take the random action(down or right), and on the second run, it will use an updated Q-Table to select the action.

- **Perform an Action**
- Choosing an action and performing the action will repeat multiple times until the training loop stops. The first action and state are selected using the Q-Table. In our case, all values of the Q-Table are zero.
- Then, the agent will move down and update the Q-Table using the Bellman equation. With every move, we will be updating values in the Q-Table and also using it for determining the best course of action.

RL : Q- learning

- Initially, the agent is in exploration mode and chooses a random action to explore the environment. The Epsilon Greedy Strategy is a simple method to balance exploration and exploitation. The epsilon stands for the probability of choosing to explore and exploits when there are smaller chances of exploring.
- At the start, the epsilon rate is higher, meaning the agent is in exploration mode. While exploring the environment, the epsilon decreases, and agents start to exploit the environment. During exploration, with every iteration, the agent becomes more confident in estimating Q-values



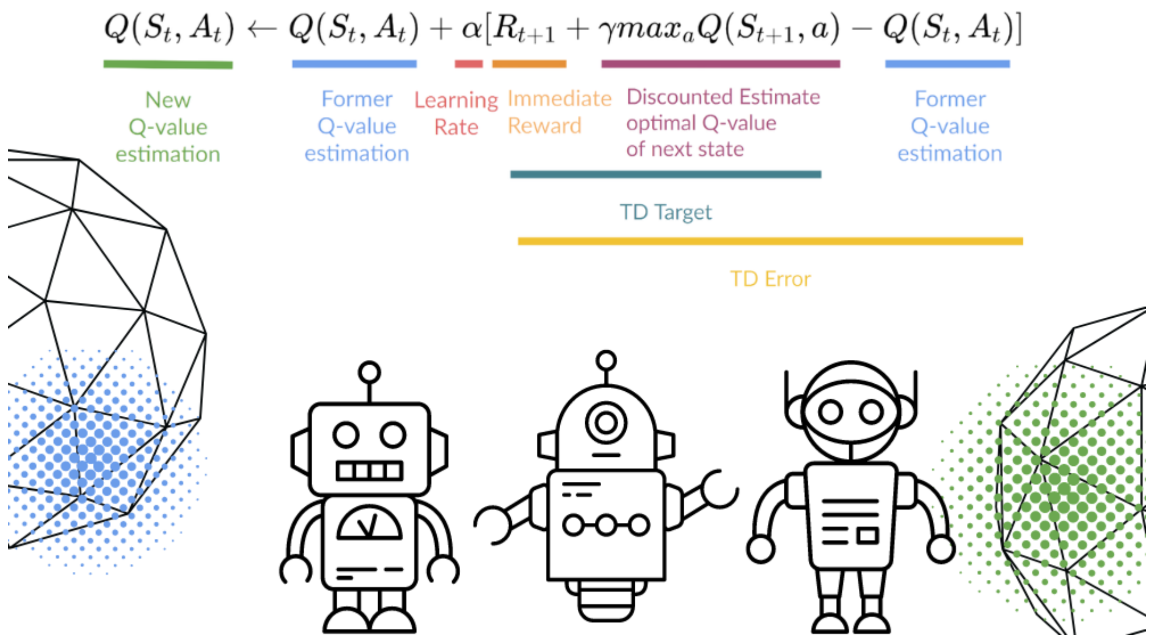
	→	←	↑	↓
Start	0	0	0	1
Idle	0	0	0	0
Hole	0	0	0	0
End	0	0	0	0

RL : Q- learning

- In the frozen lake example, the agent is unaware of the environment, so it takes random action (move down) to start. As we can see in the above image, the Q-Table is updated using the Bellman equation.
- Measuring the Rewards
  - After taking the action, we will measure the outcome and the reward.
  - The reward for reaching the goal is +1
  - The reward for taking the wrong path (falling into the hole) is 0
  - The reward for Idle or moving on the frozen lake is also 0.

RL : Q- learning

- **Update Q-Table**
- We will update the function Q(St, At) using the equation. It uses the previous episode's estimated Q-values, learning rate, and Temporal Differences error. Temporal Differences error is calculated using Immediate reward, the discounted maximum expected future reward, and the former estimation Q-value.
- The process is repeated multiple times until the Q-Table is updated and the Q-value function is maximized.



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

New Q-value estimation
Former Q-value estimation
Learning Rate
Immediate Reward
Discounted Estimate optimal Q-value of next state
Former Q-value estimation
TD Target
TD Error

RL : Q- learning

- At the start, the agent is exploring the environment to update the Q-table. And when the Q-Table is ready, the agent will start exploiting and start taking better decisions.

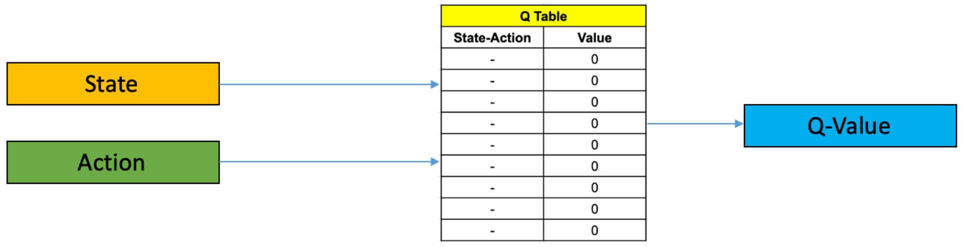
	→	←	↑	↓
Start	0	1	0	0
Idle	2	0	0	3
Hole	0	2	0	0
End	1	0	0	0

- In the case of a frozen lake, the agent will learn to take the shortest path to reach the goal and avoid jumping into the holes.

Deep Q

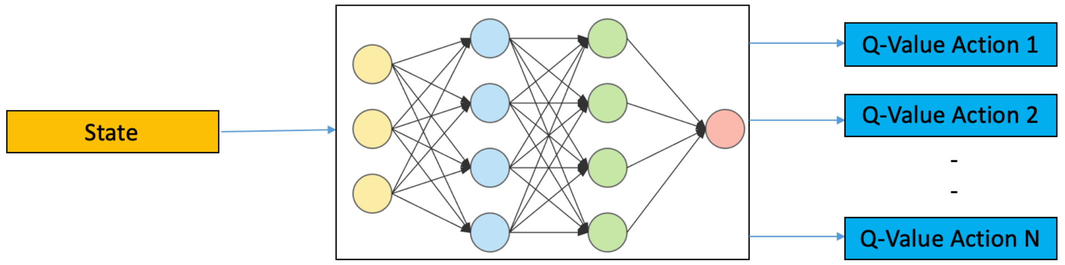
RL : Deep - Q

- The deep Q-learning model breaks the chain in order to find the optimal Q-value function.
- It determines this by combining Q-learning and a neural network.
- The uses of the deep Q-learning algorithm can be stated as finding the input and the optimal Q-value for all possible actions as the output.



Q Table	
State-Action	Value
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0

Q Learning



Deep Q Learning

RL : Deep - Q

- In deep Q-learning, past experiences are stored in memory and the future action depends on the Q-network output. It is how Q-network calculates the Q-value at state st. Similarly, the target network (neural network) determines the Q-value for state St+1 (next state) to stabilize the training.
- As an additional feature, it copies Q-value count as training dataset for each iteration of Q-value in the Q-network, thereby blocking abrupt increases in Q-value count.
- The deep Q-learning algorithm relies on neural networks and Q-learning. In this case, the neural network stores experience as a tuple in its memory with a tuple that includes <State, Next State, Action, Reward>.
- A random sample of previous data increases the stability of neural network training. As a result, deep Q-learning uses another concept to boost agents' performance – experience replay or store experience from the past.
- The target network uses experience replay to determine the Q-value while the Q-network uses it for training. Calculating the loss becomes easier when the squared difference between the target and predicted Q-values is known. The equation is given below:

$$Loss = (r + \gamma max_{a'} Q(s', a') - Q(s, a))^2$$

RL : Deep - Q

- **Why 'deep' Q-learning ?**
- The Q-learning algorithm creates a cheat sheet for agents in a simple but quite powerful manner. By doing so, the agents can determine exactly what action to take.
- Consider an environment with 10,000 states and 1,000 actions per state. Wouldn't the cheat sheet be too long? There would be a table with 10 million cells. Eventually, things would spiral out of control. Thus, it would be impossible to estimate the Q-value for new states based on previously explored states.
- Two main problems would arise:
    - As the number of states increases, the amount of memory required to save and update the table would expand.
    - It would become impossible to explore every state to create the required Q-table in the time required.
- A neural network becomes helpful in approximating the Q-value function in deep Q-learning. The state gets taken as the input and Q-values for all possible actions get generated as the output.

Deep – Q : Working

- **Initialize main and target neural networks**
- The Q-table implementation is the prime difference between Q-learning and deep Q-learning. In the latter, neural networks replace the regular Q-table.
- In neural networks, input states get mapped to (action, Q-value) pairs instead of state-action pairs to Q-values. A unique feature of deep Q-learning is that it uses two neural networks in its learning process.
- Despite having the same architecture, these networks differ in their weights. During each N-step, the weights get copied between the main and target networks. Both networks help the algorithm learn more effectively and stabilize the learning process.
- How to map states to (action, Q-value) pairs
  - An (action, Q-value) pair gets mapped between the main and target neural networks. The output nodes (representing actions) contain a floating-point value representing each action's Q-value. As the output nodes do not reflect a probability distribution, they cannot add up to 1.

Deep – Q : Working

- **Choose an action using the Epsilon-Greedy exploration strategy**
- Using the Epsilon-Greedy exploration strategy, an agent selects an action at random with probability as epsilon. It further uses its best-known action with probability equivalent to 1 epsilon.
- What are the best-known actions from a network?
  - There is a mapping between input states and output actions in both the target and main models. These output actions correspond to the predicted Q-value for the model. When the largest Q-value gets predicted for an action, that action is the best-known at that state.

- **Bellman equation to update network weight**
- The agents have to perform an action when they choose it and update the main and target networks using the Bellman equation. In order to update them, a deep Q-learning agent deploys experience replay which enables them to acquire knowledge about their environment.
- Essentially, every step involves sampling and training which is based on a batch of 4 steps based on the past experiences. After 100 such steps, the weights of the main network get transferred to the weight of the target network.

Thank you!