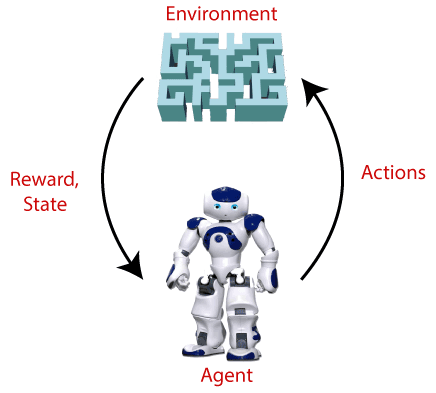
Reinforcement Learning

What is 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.](https://www.javatpoint.com/supervised-machine-learning)
* Since there is no labeled 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."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
* It is a core part of [Artificial intelligence](https://www.javatpoint.com/artificial-intelligence-tutorial), and all [AI agent](https://www.javatpoint.com/agents-in-ai) works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
* **Example:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
* The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.
* The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



Terms used in 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 (a).

Key Features of Reinforcement Learning

* In RL, the agent is not instructed about the environment and what actions need to be taken.
* It is based on the hit and trial process.
* The agent takes the next action and changes states according to the feedback of the previous action.
* The agent may get a delayed reward.
* The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

Approaches to implement Reinforcement Learning

There are mainly three ways to implement reinforcement-learning in ML, which are:

1. **Value-based:**  
   The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.
2. **Policy-based:**  
   Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward.  
   The policy-based approach has mainly two types of policy:
   * **Deterministic:** The same action is produced by the policy (π) at any state.
   * **Stochastic:** In this policy, probability determines the produced action.
3. **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

**1) 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 behavior 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:

urly wages by $1 for US workers

**For deterministic policy: a = π(s) For stochastic policy: π(a | s) = P[At =a | St = s]**

**2) 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.

**3) 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.

**4) Model:** The last element of reinforcement learning is the model, which mimics the behavior 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.

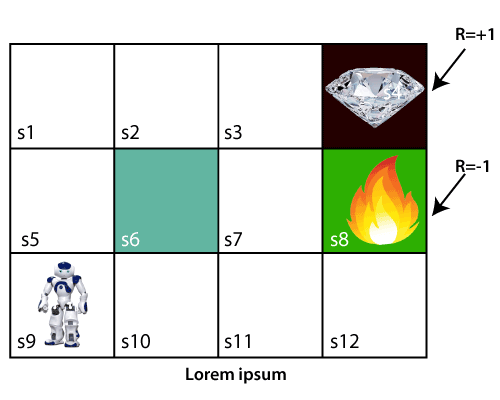
The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems **with the help of the model** are termed as the **model-based approach**. Comparatively, an approach **without using a model** is called a **model-free approach**.

How does Reinforcement Learning Work?

To understand the working process of the RL, we need to consider two main things:

* **Environment:** It can be anything such as a room, maze, football ground, etc.
* **Agent:** An intelligent agent such as AI robot.

Let's take an example of a maze environment that the agent needs to explore. Consider the below image:

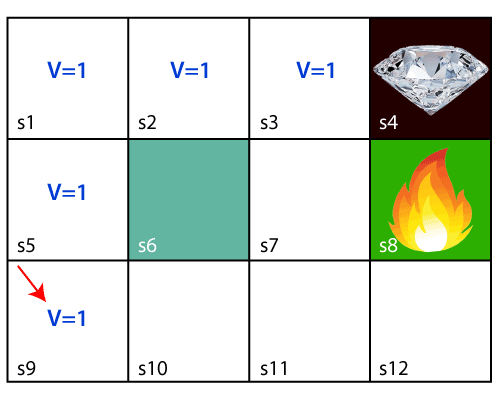


In the above image, the agent is at the very first block of the maze. The maze is consisting of an S6 block, which is a **wall**, S8 a **fire pit**, and S4 a **diamond block.**

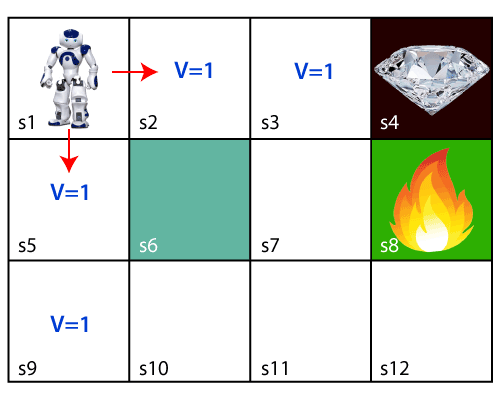
The agent cannot cross the S6 block, as it is a solid wall. If the agent reaches the S4 block, then get the **+1 reward;**if it reaches the fire pit, then gets **-1 reward point**. It can take four actions**: move up, move down, move left, and move right.**

The agent can take any path to reach to the final point, but he needs to make it in possible fewer steps. Suppose the agent considers the path **S9-S5-S1-S2-S3**, so he will get the +1-reward point.

The agent will try to remember the preceding steps that it has taken to reach the final step. To memorize the steps, it assigns 1 value to each previous step. Consider the below step:



Now, the agent has successfully stored the previous steps assigning the 1 value to each previous block. But what will the agent do if he starts moving from the block, which has 1 value block on both sides? Consider the below diagram:



It will be a difficult condition for the agent whether he should go up or down as each block has the same value. So, the above approach is not suitable for the agent to reach the destination. Hence to solve the problem, we will use the **Bellman equation**, which is the main concept behind reinforcement learning.

Types of Reinforcement learning

There are mainly two types of reinforcement learning, which are:

* **Positive Reinforcement**
* **Negative Reinforcement**

**Positive Reinforcement:**

The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior.

This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

**Negative Reinforcement:**

The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition.

It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior.

How to represent the agent state?

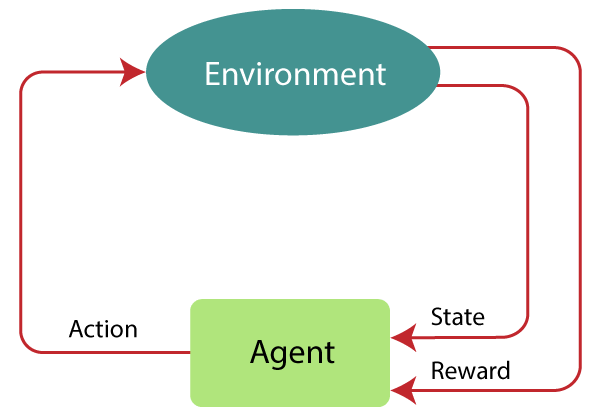
We can represent the agent state using the **Markov State** that contains all the required information from the history. The State St is Markov state if it follows the given condition:

P[St+1 | St ] = P[St +1 | S1,......, St]

The Markov state follows the **Markov property**, which says that the future is independent of the past and can only be defined with the present. The RL works on fully observable environments, where the agent can observe the environment and act for the new state. The complete process is known as Markov Decision process, which is explained below:

Markov Decision Process

Markov Decision Process or MDP, is used to **formalize the reinforcement learning problems**. If the environment is completely observable, then its dynamic can be modeled as a **Markov Process**. In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.



MDP is used to describe the environment for the RL, and almost all the RL problem can be formalized using MDP.

MDP contains a tuple of four elements (S, A, Pa, Ra):

* A set of finite States S
* A set of finite Actions A
* Rewards received after transitioning from state S to state S', due to action a.
* Probability Pa.

MDP uses **Markov property**, and to better understand the MDP, we need to learn about it.

Markov Property:

It says that ***"If the agent is present in the current state S1, performs an action a1 and move to the state s2, then the state transition from s1 to s2 only depends on the current state and future action and states do not depend on past actions, rewards, or states."***

Or, in other words, as per Markov Property, the current state transition does not depend on any past action or state. Hence, MDP is an RL problem that satisfies the Markov property. Such as in a **Chess game, the players only focus on the current state and do not need to remember past actions or states**.

**Finite MDP:**

A finite MDP is when there are finite states, finite rewards, and finite actions. In RL, we consider only the finite MDP.

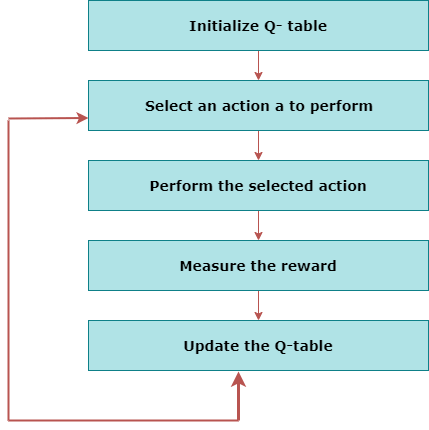
Markov Process:

Markov Process is a memoryless process with a sequence of random states S1, S2, ....., St that uses the Markov Property. Markov process is also known as Markov chain, which is a tuple (S, P) on state S and transition function P. These two components (S and P) can define the dynamics of the system.

Reinforcement Learning Algorithms

Reinforcement learning algorithms are mainly used in AI applications and gaming applications. The main used algorithms are:

* **Q-Learning:**
  + Q-learning is an **Off policy RL algorithm**, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions.
  + It learns the value function Q (S, a), which means how good to take action "**a**" at a particular state "**s**."
  + The below flowchart explains the working of Q- learning:



* **State Action Reward State action (SARSA):**
  + SARSA stands for **State Action Reward State action**, which is an **on-policy** temporal difference learning method. The on-policy control method selects the action for each state while learning using a specific policy.
  + The goal of SARSA is to calculate the **Q π (s, a) for the selected current policy π and all pairs of (s-a).**
  + The main difference between Q-learning and SARSA algorithms is that **unlike Q-learning, the maximum reward for the next state is not required for updating the Q-value in the table.**
  + In SARSA, new action and reward are selected using the same policy, which has determined the original action.
  + The SARSA is named because it uses the quintuple **Q(s, a, r, s', a').** Where,  
     s: original state

a: Original action

r: reward observed while following the states s' and a': New state, action pair.

* **Deep Q Neural Network (DQN):**
  + As the name suggests, DQN is a **Q-learning using Neural networks**.
  + For a big state space environment, it will be a challenging and complex task to define and update a Q-table.
  + To solve such an issue, we can use a DQN algorithm. Where, instead of defining a Q-table, neural network approximates the Q-values for each action and state.

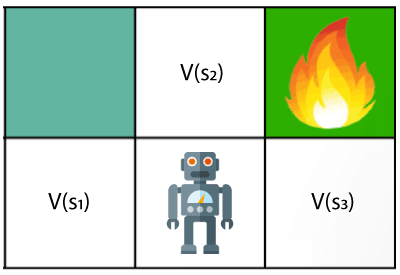
Now, we will expand the Q-learning.

Q-Learning Explanation:

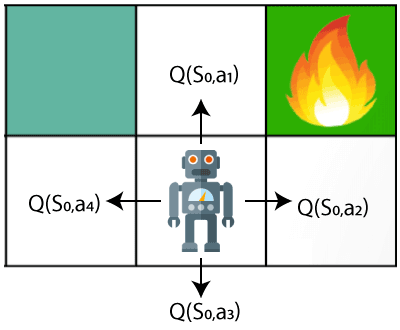
* Q-learning is a popular model-free reinforcement learning algorithm based on the Bellman equation.
* **The main objective of Q-learning is to learn the policy which can inform the agent that what actions should be taken for maximizing the reward under what circumstances.**
* It is an **off-policy RL** that attempts to find the best action to take at a current state.
* The goal of the agent in Q-learning is to maximize the value of Q.
* The value of Q-learning can be derived from the Bellman equation. Consider the Bellman equation given below:

Q-Learning Explanation

In the equation, we have various components, including reward, discount factor (γ), probability, and end states s'. But there is no any Q-value is given so first consider the below image:



In the above image, we can see there is an agent who has three values options, V(s1), V(s2), V(s3). As this is MDP, so agent only cares for the current state and the future state. The agent can go to any direction (Up, Left, or Right), so he needs to decide where to go for the optimal path. Here agent will take a move as per probability bases and changes the state. But if we want some exact moves, so for this, we need to make some changes in terms of Q-value. Consider the below image:



Q- represents the quality of the actions at each state. So instead of using a value at each state, we will use a pair of state and action, i.e., Q(s, a). Q-value specifies that which action is more lubricative than others, and according to the best Q-value, the agent takes his next move. The Bellman equation can be used for deriving the Q-value.

To perform any action, the agent will get a reward R(s, a), and also he will end up on a certain state, so the Q -value equation will be:

Q-Learning Explanation

Hence, we can say that, ***V(s) = max [Q(s, a)]***

Q-Learning Explanation

**The above formula is used to estimate the Q-values in Q-Learning.**

**What is 'Q' in Q-learning?**

The Q stands for **quality** in **Q-learning**, which means it specifies the quality of an action taken by the agent.

Q-table:

A Q-table or matrix is created while performing the Q-learning. The table follows the state and action pair, i.e., [s, a], and initializes the values to zero. After each action, the table is updated, and the q-values are stored within the table.

The RL agent uses this Q-table as a reference table to select the best action based on the q-values.

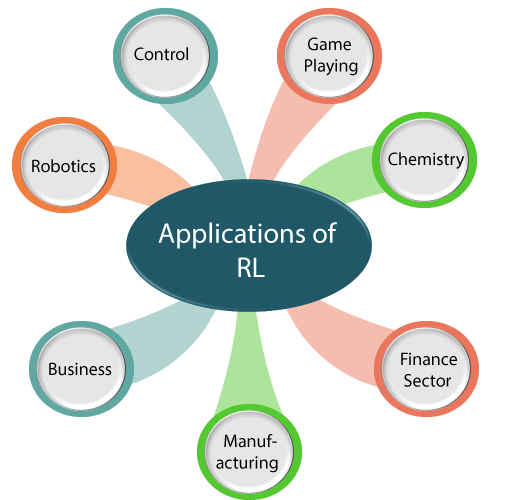
Difference between Reinforcement Learning and Supervised Learning

The Reinforcement Learning and Supervised Learning both are the part of machine learning, but both types of learnings are far opposite to each other. The RL agents interact with the environment, explore it, take action, and get rewarded. Whereas supervised learning algorithms learn from the labeled dataset and, on the basis of the training, predict the output.

The difference table between RL and Supervised learning is given below:

|  |  |
| --- | --- |
| **Reinforcement Learning** | **Supervised Learning** |
| RL works by interacting with the environment. | Supervised learning works on the existing dataset. |
| The RL algorithm works like the human brain works when making some decisions. | Supervised Learning works as when a human learns things in the supervision of a guide. |
| There is no labeled dataset is present | The labeled dataset is present. |
| No previous training is provided to the learning agent. | Training is provided to the algorithm so that it can predict the output. |
| RL helps to take decisions sequentially. | In Supervised learning, decisions are made when input is given. |

Reinforcement Learning Applications



1. **Robotics:**
   1. RL is used in **Robot navigation, Robo-soccer, walking, juggling**, etc.
2. **Control:**
   1. RL can be used for **adaptive control** such as Factory processes, admission control in telecommunication, and Helicopter pilot is an example of reinforcement learning.
3. **Game Playing:**
   1. RL can be used in **Game playing** such as tic-tac-toe, chess, etc.
4. **Chemistry:**
   1. RL can be used for optimizing the chemical reactions.
5. **Business:**
   1. RL is now used for business strategy planning.
6. **Manufacturing:**
   1. In various automobile manufacturing companies, the robots use deep reinforcement learning to pick goods and put them in some containers.
7. **Finance Sector:**
   1. The RL is currently used in the finance sector for evaluating trading strategies.

# What are the main challenges of reinforcement learning, and how to overcome them?

As popular as it may be, RL does not come without its challenges. Analytics India Magazine has noted some common RL challenges and ways to overcome them.

### ****Sample efficiency****

One of the major challenges with RL is efficiently learning with limited samples. Sample efficiency denotes an algorithm making the most of the given sample. Essentially, it is also the amount of experience the algorithm has to generate during training to reach efficient performance. The challenge is it takes the RL system a considerable amount of time to be efficient. For instance, [DeepMind’s](https://analyticsindiamag.com/a-historical-tale-of-deepminds-games/) AlphaGoZero played five million Go games before beating the world champion in it.

[A research paper by Gen Li Princeton](https://arxiv.org/pdf/2105.08024.pdf) et al. described this as, “Given that the state space and the action space could both be unprecedentedly enormous, it is often infeasible to request a sample size exceeding the fundamental limit set forth by the ambient dimension in the tabular setting. As a result, the quest for sample efficiency cannot be achieved in general without exploiting proper low-complexity structures underlying the problem of interest.” [An IEEE paper](https://ieeexplore.ieee.org/document/9655259/figures#figures) introduced the ‘safe set algorithm’ as a solution. The algorithm monitors and modifies control and evaluates RL in a clustered dynamic environment, challenging existing RLs.

### ****Reproducibility issues****

“When combined with the unavailability of code and models, the result is that the approach is very difficult, if not impossible, to reproduce, study, improve upon, and extend,” said [Facebook research](https://arxiv.org/pdf/1902.04522.pdf) in their quest to reproduce DeepMind’s AlphaZero (the team succeeded ultimately).

Neural networks are opaque black boxes whose workings are mysteries to even the creators. They are also increasing in size and complexity, backed by huge datasets, computing power and hours of training. These factors make RL models very difficult to replicate.

In recent years, there’s been a growing movement in AI to counteract the so-called [reproducibility crisis](https://www.nature.com/articles/d41586-019-03895-5), a high-stakes version of the classic it-worked-on-my-machine coding problem. The crisis manifests in problems ranging from AI research that selectively reports algorithm runs to idealised results courtesy of heavy GPU firepower.

The [Leiden Institute of Advanced Computer Science](https://arxiv.org/pdf/2203.01075.pdf) paper suggests leveraging the ‘minimal traces’ concept. The idea supports “re-simulation of action sequences in deterministic RL environments”, allowing reviewers to verify, re-use, and manually inspect experimental results without needing large compute clusters. Other solutions include tracking and logging experiments, submitting code and creating a metadata repository.

Second, rather than having entrants submit their agents, which could conceivably be trained with research-lab levels of GPU wattage, they’re required to submit code trained using the organisers’ machine. Finally, they also introduce randomising elements to make sure results track across different game versions.

### ****Performing in real-life scenarios****

RL agents learn from exploration of the artificial environments. Talking about AlphaZero, [DeepMind said](https://deepmind.com/blog/article/generally-capable-agents-emerge-from-open-ended-play" \t "_blank), “Through [reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning) (RL), this single system learnt by playing round after round of games through a repetitive process of trial and error.” The agents can fail in manufactured environments, but they do not have the opportunity to fail and learn in real-life scenarios. Usually, in real environments, the agent lacks the space to observe the environment well enough to use past training data to decide on a winning strategy. This also includes the reality gap, where the agent cannot gauge the difference between the learning simulation and the real world.

General techniques used by researchers include learning by mimicking the behaviour desired, learning through accurate simulations, better algorithm design and demonstrations and, most popularly used, training the agents on a reward and punishment mechanism. Since the agent is rewarded for correct actions and punished for incorrect ones, it is trained to maximise the right ones.

### ****Sparse rewards****

The reward technique discussed in the previous point is not foolproof. Since the rewards are sparsely distributed in the environment, a possible issue is an agent not observing the situation enough to notice the reward signals and maximise specific actions. This also occurs when the environment cannot provide reward signals in time; for instance, in many situations, the agent receives a green flag only when it is close enough to the target.

Curiosity-driven methods are widely used to encourage the agent to explore the environment and learn to tackle tasks in it. The researchers in the paper ‘Curiosity-driven exploration by self-supervised prediction’ proposed an Intrinsic Curiosity Module (ICM) to support the agent in exploration and prompt it to choose actions based on reduced errors. Another approach is curriculum learning, where the agent is presented with various tasks in ascending order of complexity. This imitates the learning order of humans.

### ****Offline reinforcement****

While in real RL, the agent incrementally improves its policy with new experiences, [offline RL](https://analyticsindiamag.com/how-to-address-offline-reinforcement-learning-challenges/) works on a fixed set of logged experiences with minimal interaction with the environment. This method eliminates the need for repeated training of AI agents to scale. Still, it proposes the challenge where if the model, which is being trained with an existing dataset, takes action different from the data collection agent, one cannot determine the reward provided to the learning model.

Another issue, as suggested by [Google AI](https://ai.googleblog.com/2020/08/tackling-open-challenges-in-offline.html), is the distributional shift. This occurs when the RL algorithms must learn to make decisions that differ from the decisions taken in the dataset to improve over the historical data.

### Comparision Between Supervised Learning and Reinforcement Learning

Below is the Top 7 comparison between Supervised Learning and Reinforcement Learning:

### Key Differences between Supervised Learning and Reinforcement Learning

Below is the difference between Supervised Learning and Reinforcement Learning:

1. Supervised Learning has two main tasks called Regression and Classification whereas Reinforcement Learning has different tasks such as exploitation or exploration, Markov’s decision processes, Policy Learning, Deep Learning and value learning.
2. Supervised Learning analyses the training data and produces a generalized formula, In Reinforcement Learning basic reinforcement is defined in the model Markov’s Decision process.
3. In Supervised Learning, each example will have a pair of input objects and an output with desired values whereas in Reinforcement Learning Markov’s Decision process means the agent interacts with the environment in discrete steps i.e., agent makes an observation for every time period “t” and receives a reward for every observation and finally, the goal is to collect as many rewards as possible to make more observations.
4. In Supervised Learning, different numbers of algorithms exist with advantages and disadvantages that suit the system requirement. In Reinforcement Learning, Markov’s decision process provides a mathematical framework for modeling and decision making situations.
5. The most used learning algorithms for both Supervised learning and Reinforcement learning are linear regression, logistic regression, decision trees, Bayes Algorithm, Support Vector Machines, and Decision trees, etc., those which can be applied in different scenarios.
6. In Supervised Learning, the goal is to learn the general formula from the given examples by analyzing the given inputs and outputs of a function. In Reinforcement Learning, the goal is in such way like controlling mechanism like control theory, gaming theory, etc., for example, driving a vehicle or playing gaming against another player, etc.,
7. In Supervised learning both input and output will be available for decision making where the learner will be trained on many examples or sample data given whereas in reinforcement learning sequential decision making happens and the next input depends on the decision of the learner or system, examples are like playing chess against an opponent, robotic movement in an environment, gaming theory.
8. In Supervised learning, just a generalized model is needed to classify data whereas in reinforcement learning the learner interacts with the environment to extract the output or make decisions, where the single output will be available in the initial state and output, will be of many possible solutions.
9. Supervised learning means the name itself says it is highly supervised whereas the reinforcement learning is less supervised and depends on the learning agent in determining the output solutions by arriving at different possible ways in order to achieve the best possible solution.
10. Supervised learning makes prediction depending on a class type whereas reinforcement learning is trained as a learning agent where it works as a reward and action system.
11. In Supervised learning, a huge amount of data is required to train the system for arriving at a generalized formula whereas in reinforcement learning the system or learning agent itself creates data on its own to by interacting with the environment.
12. Both Supervised learning and reinforcement learning are used to create and bring some innovations like robots that reflect human behavior and works like a human and interacting more with the environment causes more growth and development to the systems performance results in more technological advancement and growth.

### Supervised Learning and Reinforcement Learning Comparison Table

Below is the comparison table between Supervised Learning and Reinforcement Learning.

|  |  |  |
| --- | --- | --- |
| **BASIS FOR**  **COMPARISON** | **Supervised Learning** | **Reinforcement learning** |
| **Definition** | Works on existing or given sample data or examples | Works on interacting with the environment |
| **Preference** | Preferred in generalized working mechanisms where routine tasks are required to be done | Preferred in the area of Artificial Intelligence |
| **Area** | Comes under the area of Machine Learning | Comes under the area of Machine Learning |
| **Platform** | Operated with interactive software systems or applications | Supports and works better in Artificial Intelligence where Human Interaction is prevalent |
| **Generality** | Many open source projects are evolving of development in this area | More useful in Artificial Intelligence |
| **Algorithm** | Many algorithms exist in using this learning | Neither supervised nor unsupervised algorithms are used |
| **Integration** | Runs on any platform or with any applications | Runs with any hardware or software devices |

### Conclusion

Supervised Learning is an [area of Machine Learning](https://www.educba.com/what-is-machine-learning/) where the analysis of generalized formula for a software system can be achieved by using the training data or examples given to the system, this can be achieved only by sample data for training the system.

Reinforcement Learning has a learning agent that interacts with the environment to observe the basic behavior of a human system in order to achieve the behavioral phenomenon. The applications include control theory, operations research, gaming theory, information theory, etc.,

The applications of supervised and reinforcement learning differ on the purpose or goal of a software system. Both Supervised Learning and Reinforcement Learning have huge advantages in the area of their applications in computer science.

The development of different new algorithms causes more development and improvement of performance and growth of machine learning that will result in sophisticated learning methods in Supervised learning as well as reinforcement learning.

From the above discussion, we can say that Reinforcement Learning is one of the most interesting and useful parts of Machine learning. In RL, the agent explores the environment by exploring it without any human intervention. It is the main learning algorithm that is used in Artificial Intelligence. But there are some cases where it should not be used, such as if you have enough data to solve the problem, then other ML algorithms can be used more efficiently. The main issue with the RL algorithm is that some of the parameters may affect the speed of the learning, such as delayed feedback.