# ARTIFICIAL INTELLIGENCE

Unit – 8 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 labeled data, so the agent is bound to learn by its experience only.

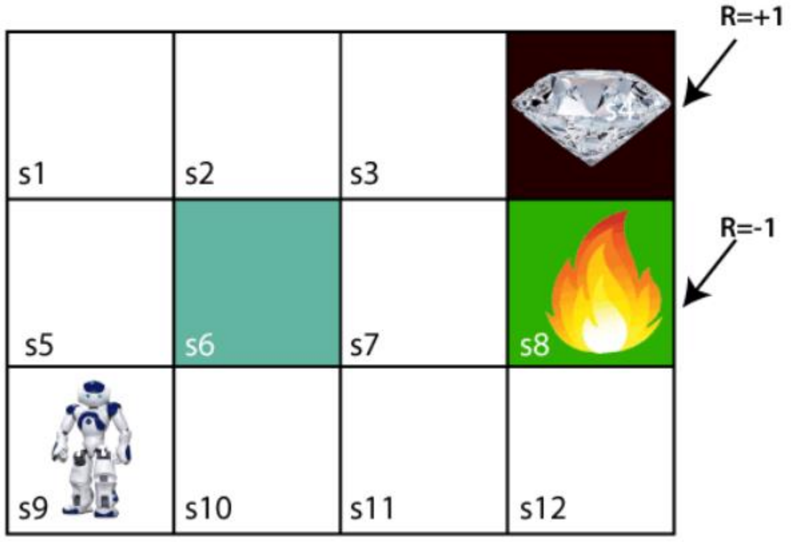
RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.

## REINFORCEMENT LEARNING

- 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. 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 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.
- Definition: ***Reinforcement learning is a type of AI method where an intelligent agent (computer program) interacts with the environment and learns to act within that.***

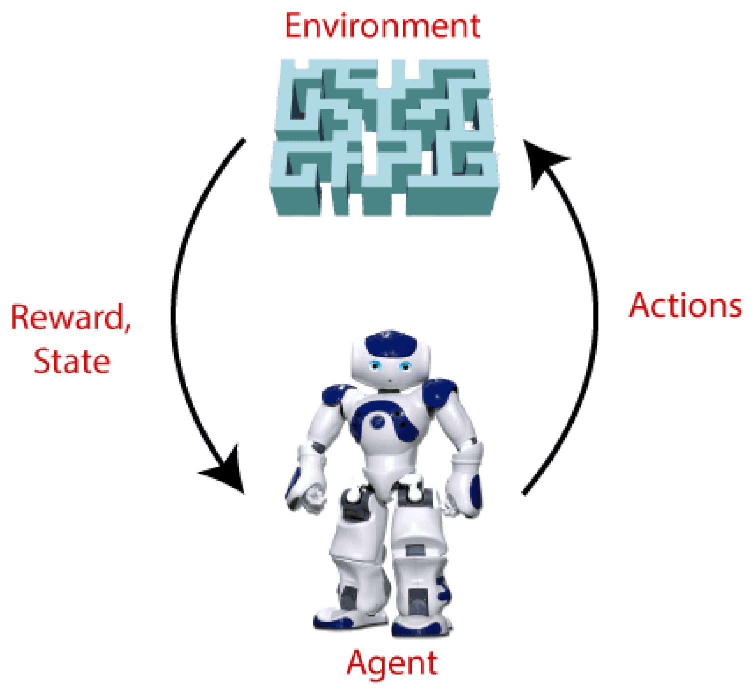
## EXAMPLE

- 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.



## REINFORCEMENT LEARNING

- 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

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:
   1. **Deterministic:** The same action is produced by the policy (π) at any state.
   2. **Stochastic:** In this policy, probability determines the produced action.

## CONT…

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

## ELEMENTS OF REINFORCEMENT LEARNING

- **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:

- **For deterministic policy: $a = \pi(s)$**
  **For stochastic policy: $\pi(a \mid s) = P[At = a \mid St = s]$**

## ELEMENTS OF REINFORCEMENT LEARNING

- **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.

## ELEMENTS OF REINFORCEMENT LEARNING

- **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.

## ELEMENTS OF REINFORCEMENT LEARNING

- **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.

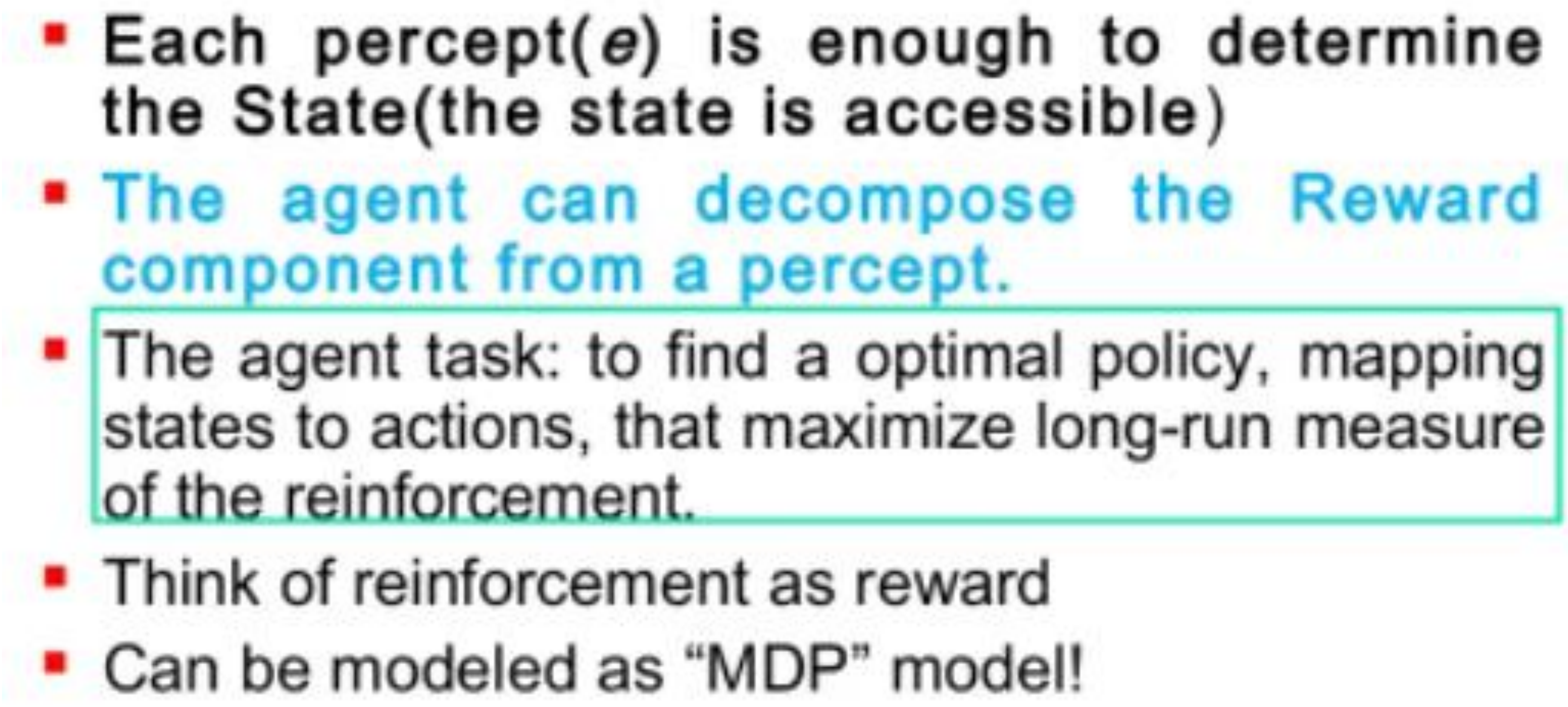
## REINFORCEMENT LEARNING

- Let's consider a problem where an agent can be in various states and can choose an action from a set of actions. Such type of problems are called ***Sequential Decision Problems***.

- An MDP is the mathematical framework which captures such a **fully observable, non-deterministic environment** with **Markovian Transition Model and additive rewards** in which the agent acts.

- The solution to an MDP is an ***optimal policy*** which refers to the choice of action for every state that **maximizes overall cumulative reward**. Thus, the ***transition model*** that represents an agent's environment(when the environment is known) and the ***optimal policy*** which decides what action the agent needs to perform in each state are required elements for training the agent learn a specific behavior.

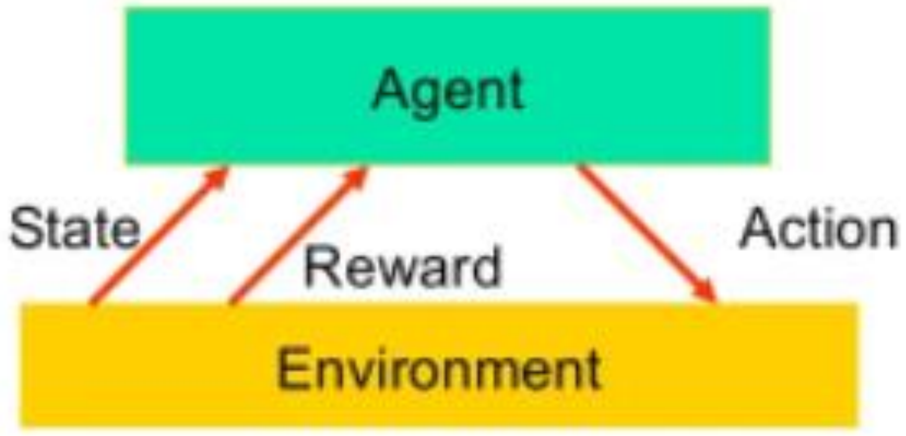
## REINFORCEMENT LEARNING MODEL



- Each percept(*e*) is enough to determine the State(the state is accessible)
- The agent can decompose the Reward component from a percept.
- The agent task: to find a optimal policy, mapping states to actions, that maximize long-run measure of the reinforcement.
- Think of reinforcement as reward
- Can be modeled as “MDP” model!

## Review of MDP Model

- MDP model <S,T,A,R>



- S– set of states
- A– set of actions
- T(s,a,s') = P(s'|s,a)– the probability of **transition** from *s* to *s'* given action *a*
- R(s,a)– the expected **reward** for taking action *a* in state *s*

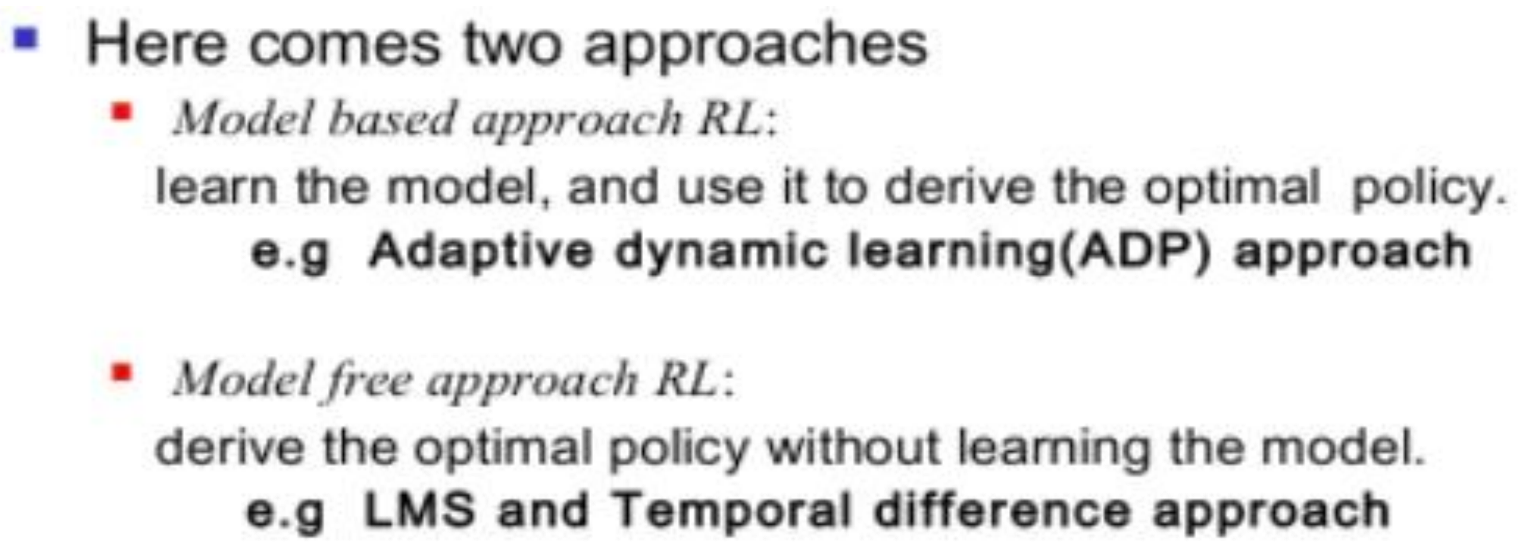
$$
s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r1]{a1} s2 \xrightarrow[r2]{a2} s3
$$

$$R(s,a) = \sum_{s'} P(s'|s,a) r(s,a,s')$$

$$R(s,a) = \sum_{s'} T(s,a,s') r(s,a,s')$$

# TYPES OF REINFORCEMENT LEARNING

- In almost all cased we do not know anything about the environment model- the transition function T(s,a,s').



- Here comes two approaches
  - *Model based approach RL:*
    learn the model, and use it to derive the optimal policy.
    **e.g Adaptive dynamic learning(ADP) approach**
  - *Model free approach RL:*
    derive the optimal policy without learning the model.
    **e.g LMS and Temporal difference approach**

## TYPES OF REINFORCEMENT LEARNING

- There are mainly two types of reinforcement learning, which are:
- model-based learning
- model-free learning

## MODEL-BASED LEARNING

- In model-based learning an agent generates an approximation of the transition function, Tˆ(s,a,s’ ), by keeping counts of the number of times it arrives in each state s’ after entering each q-state (s,a). The agent can then generate the the approximate transition function Tˆ upon request by normalizing the counts it has collected - dividing the count for each observed tuple (s,a,s’ ) by the sum over the counts for all instances where the agent was in q-state (s,a). Normalization of counts scales them such that they sum to one, allowing them to be interpreted as probabilities.

## MODEL-FREE LEARNING

- There are several model-free learning algorithms, and we'll cover three of them: direct evaluation, temporal difference learning, and Q-learning. Direct evaluation and temporal difference learning fall under a class of algorithms known as passive reinforcement learning.

## MODEL-FREE LEARNING

- In passive reinforcement learning, an agent is given a policy to follow and learns the value of states under that policy as it experiences episodes, which is exactly what is done by policy evaluation for MDPs when T and R are known.

- Q-learning falls under a second class of model-free learning algorithms known as active reinforcement learning, during which the learning agent can use the feedback it receives to iteratively update its policy while learning until eventually determining the optimal policy after sufficient exploration.

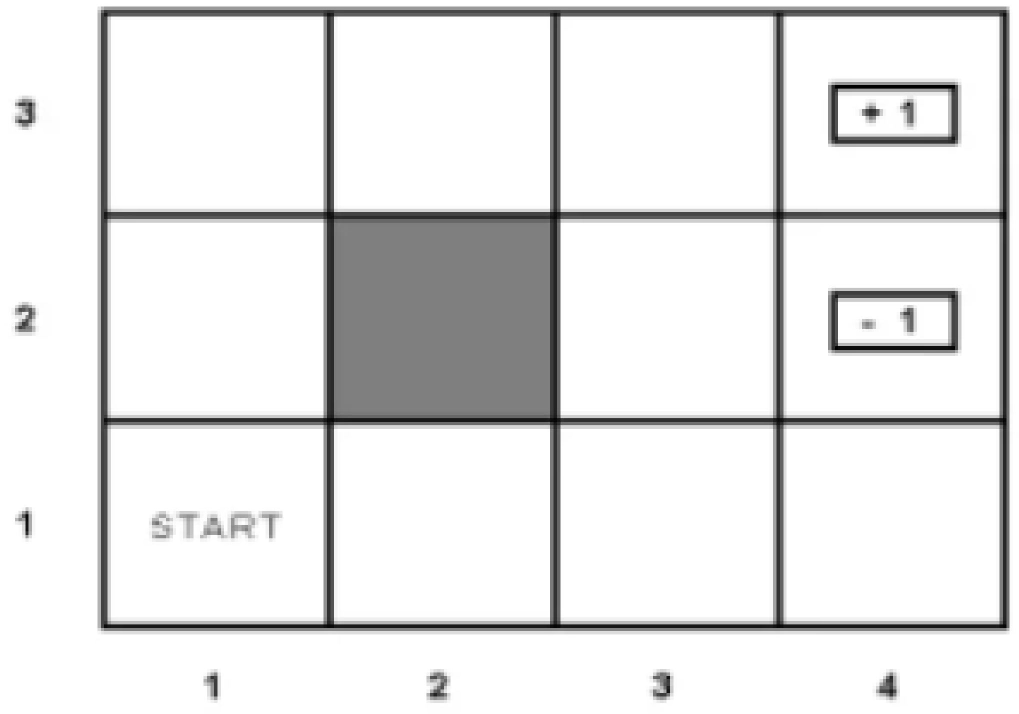
## PASSIVE LEARNING VS. ACTIVE LEARNING

1) Passive RL

2) Active RL

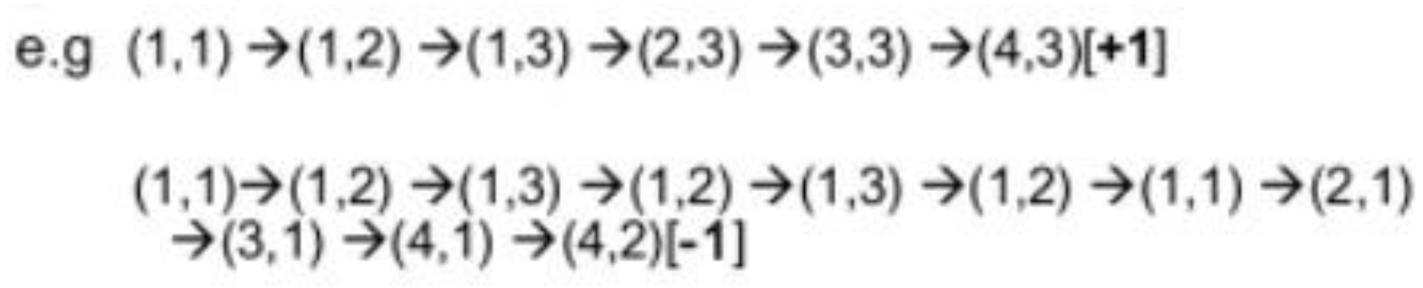
- In case of passive RL, the agent's policy is fixed which means that it is ***told what to do***. OR the agent imply watches the world going by and tries to learn the utilities of being in various states .
- In active RL, an agent ***needs to decide what to do*** as there's no fixed policy that it can act on.
- Therefore, the goal of a passive RL agent is to execute a fixed policy (sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.

## EXAMPLE



## PASSIVE LEARNING SCENARIO

- The agent see the sequence of state transitions and associate rewards.
- The environment generates state transitions and the agent perceive them



e.g (1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3)[**+1**]

(1,1) → (1,2) → (1,3) → (1,2) → (1,3) → (1,2) → (1,1) → (2,1) → (3,1) → (4,1) → (4,2)[**-1**]

- Need to update the utility value using the given training sequences.

## DIRECT UTILITY ESTIMATION/ LMS(LEAST MEANS SQUARE) UPDATING IN PASSIVE LEARNING

- ***Reward to go*** of a state

  the sum of the rewards from that state until a terminal state is reached
- Key: use observed ***reward to go*** of the state as the direct evidence of the actual expected utility of that state
- Learning utility function directly from sequence example

## DIRECT UTILITY ESTIMATION/ LMS(LEAST MEANS SQUARE) UPDATING IN PASSIVE LEARNING

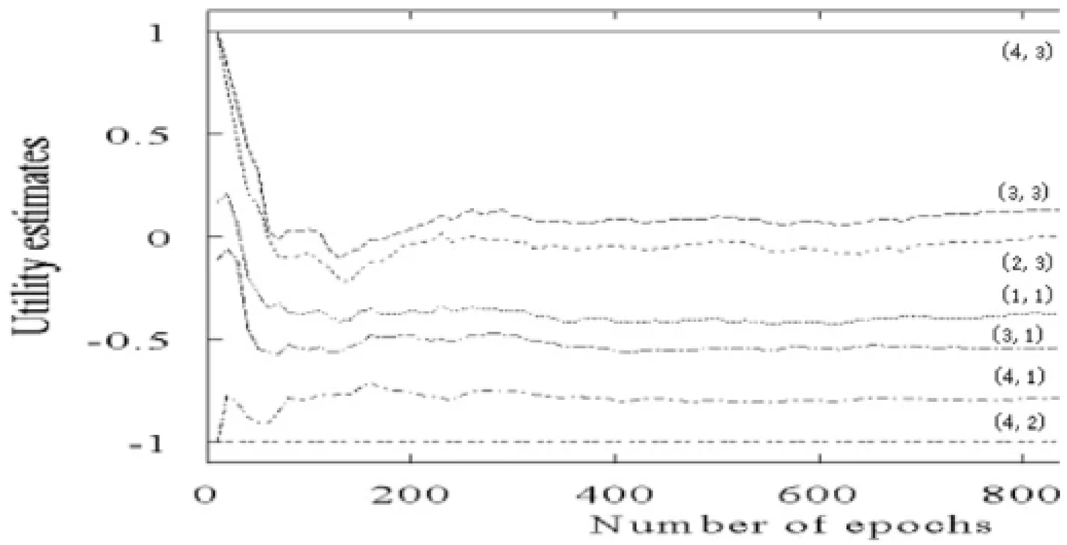
All direct evaluation does is fix some policy π and have the agent that's learning experience several episodes while following π.

As the agent collects samples through these episodes it maintains counts of the total utility obtained from each state and the number of times it visited each state.

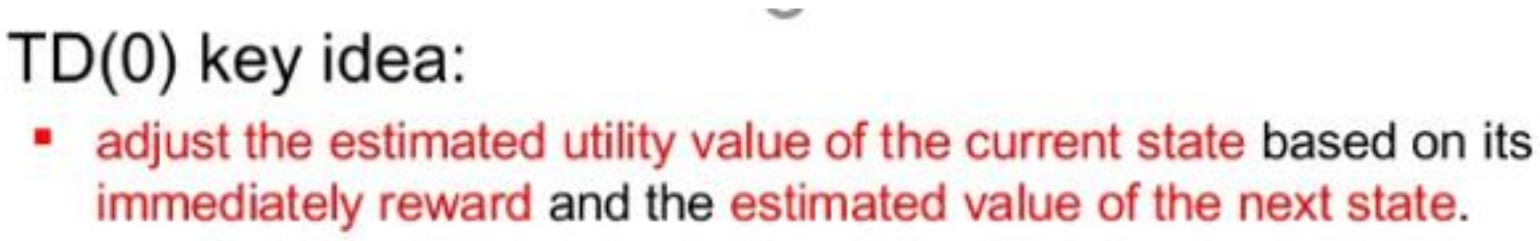
At any point, we can compute the estimated value of any state s by dividing the total utility obtained from s by the number of times s was visited.

## DRAWBACK OF LMS

- The actual utility state is constrained to be probability-weighted average of its successor's utilities.
- Meet very slowly to correct utilities values (requires a lot of sequences)
- Example: No. of epochs>1000!



## TEMPORAL DIFFERENCE LEARNING IN PASSIVE LEARNING



TD(0) key idea:

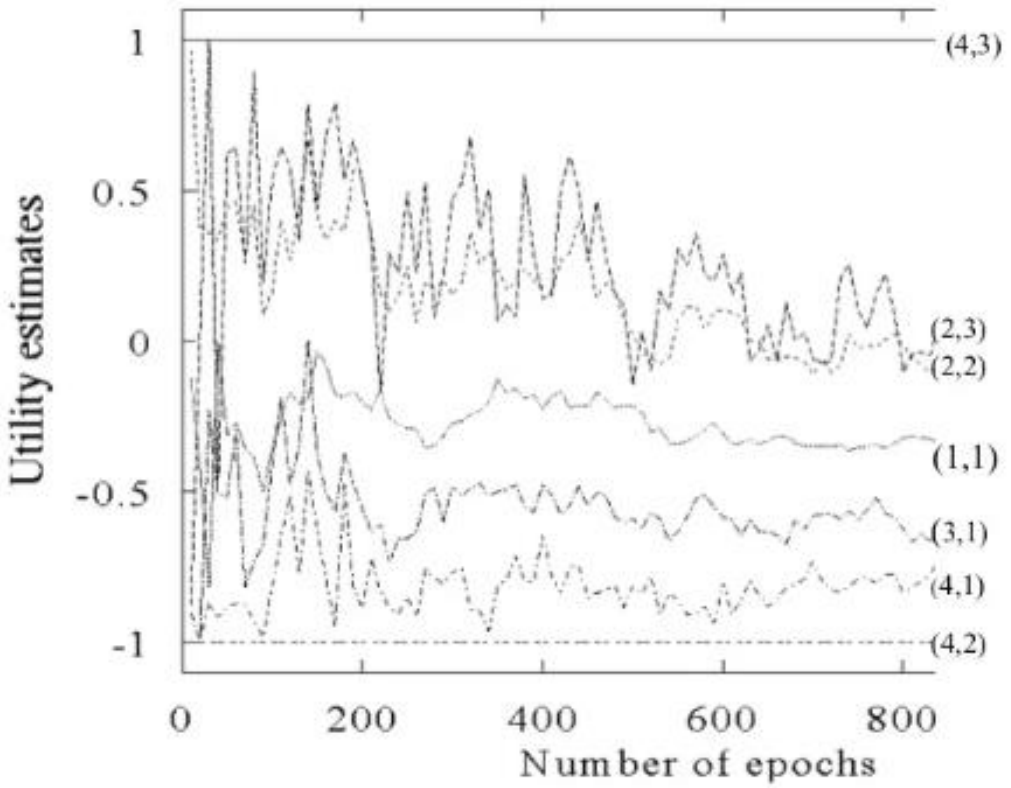
- adjust the estimated utility value of the current state based on its immediately reward and the estimated value of the next state.

TD learning does not require the agent to learn the transition model. The update occurs between successive states and agent only updates states that are directly affected.

## Temporal Difference Learning

- Temporal difference learning (TD learning) uses the idea of learning from every experience, rather than simply keeping track of total rewards and number of times states are visited and learning at the end as direct evaluation does.
- In policy evaluation, we used the system of equations generated by our fixed policy and the Bellman equation to determine the values of states under that policy (or used iterative updates like with value iteration).
- $V \pi (s) = \sum s\ 0\ T(s,\pi(s),s\ 0\ )[R(s,\pi(s),s\ 0\ ) +\gamma V \pi\ (s\ 0\ )]$
- Each of these equations equates the value of one state to the weighted average over the discounted values of that state’s successors plus the rewards reaped in transitioning to them. TD learning tries to answer the question of how to compute this weighted average without the weights, cleverly doing so with an exponential moving average.

## EXAMPLE



Utility estimates
1
0.5
0
-0.5
-1
0
200
400
600
800
Number of epochs
(4,3)
(2,3)
(2,2)
(1,1)
(3,1)
(4,1)
(4,2)

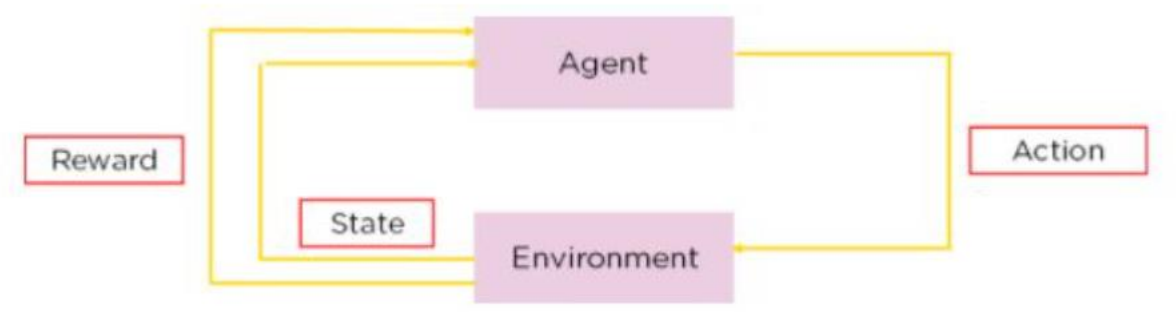
## ADAPTIVE DYNAMIC PROGRAMMING IN PASSIVE LEARNING

- ADP is a smarter method than Direct Utility Estimation as it runs trials to learn the model of the environment by estimating the utility of a state as a sum of reward for being in that state and the expected discounted reward of being in the next state.

- It is a fast but can become quite costly to compute for large state spaces. ADP is a model based approach and requires the transition model of the environment.

## Q-LEARNING IN REINFORCEMENT SYSTEM

- 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 IN REINFORCEMENT SYSTEM

- 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.

## Q-LEARNING IN REINFORCEMENT SYSTEM

- Both direct evaluation and TD learning will eventually learn the true value of all states under the policy they follow.
- However, they both have a major inherent issue - we want to find an optimal policy for our agent, which requires knowledge of the q-values of states.
- To compute q-values from the values we have, we require a transition function and reward function as dictated by the Bellman equation.

## EXAMPLE-1

- 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.

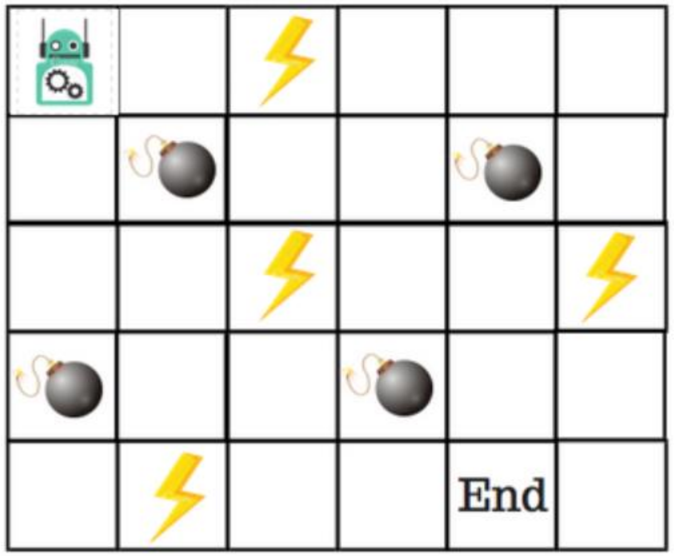
Television
Advertisement

- 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.

Television
Advertisement
DVD

## EXAMPLE-2

- Let's say that a robot has to cross a maze and reach the end point. There are mines, and the robot can only move one tile at a time. If the robot steps onto a mine, the robot is dead. The robot has to reach the end point in the shortest time possible.



## CONT…

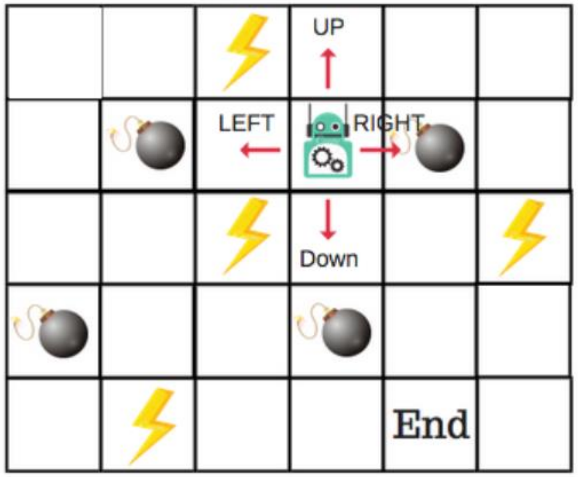
- The scoring/reward system is as below:

1. The robot loses 1 point at each step. This is done so that the robot takes the shortest path and reaches the goal as fast as possible.
2. If the robot steps on a mine, the point loss is 100 and the game ends.
3. If the robot gets power ⚡, it gains 1 point.
4. If the robot reaches the end goal, the robot gets 100 points.

**How do we train a robot to reach the end goal with the shortest path without stepping on a mine?**

## SOLUTION: INTRODUCING THE Q-TABLE

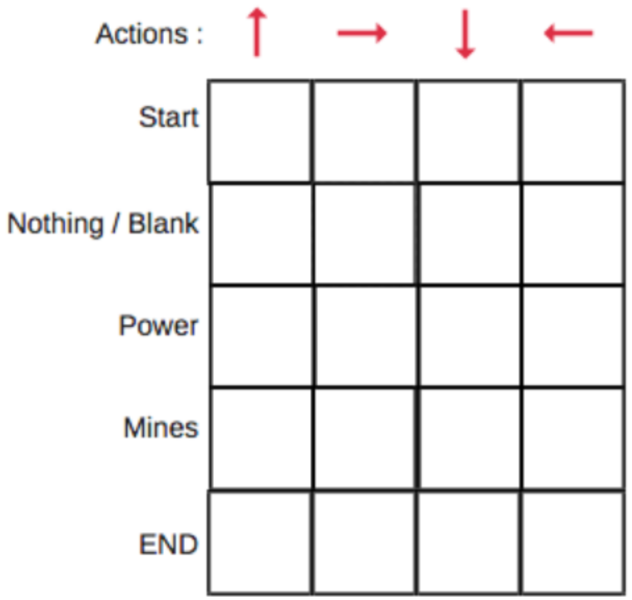
- Q-Table is just a fancy name for a simple lookup table where we calculate the maximum expected future rewards for action at each state. Basically, this table will guide us to the best action at each state.



- There will be four numbers of actions at each non-edge tile. When a robot is at a state it can either move up or down or right or left.

## Q-TABLE

- **In the Q-Table, the columns are the actions and the rows are the states.**



| Actions : | ↑ | → | ↓ | ← |
|---|---|---|---|---|
| Start | | | | |
| Nothing / Blank | | | | |
| Power | | | | |
| Mines | | | | |
| END | | | | |

- Each Q-table score will be the maximum expected future reward that the robot will get if it takes that action at that state. This is an iterative process, as we need to improve the Q-Table at each iteration.

## CONT...

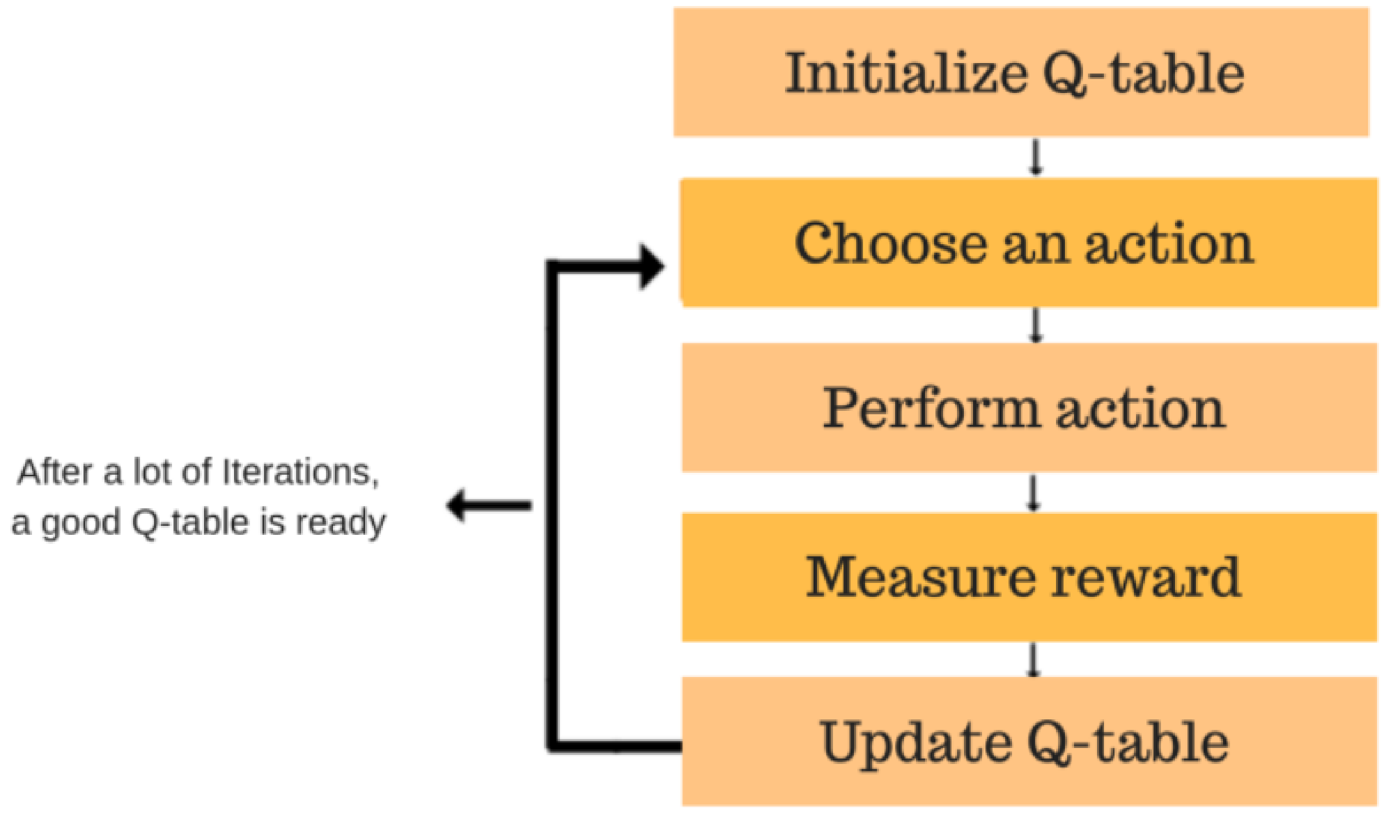
- questions are:
- How do we calculate the values of the Q-table?
- Are the values available or predefined?
- Answer:
- **Q-function**
- The **Q-function** uses the Bellman equation and takes two inputs: state (**s**) and action (**a**).

$$\underbrace{\boxed{Q^{\pi}(s_t, a_t)}}_{\text{Q-Values for the state given a particular state}} = \underbrace{\boxed{\underline{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots}}_{\text{Expected discounted cumulative reward}} \underbrace{\boxed{s_t, a_t}}_{\text{Given the state and action}}]$$

## CONT…

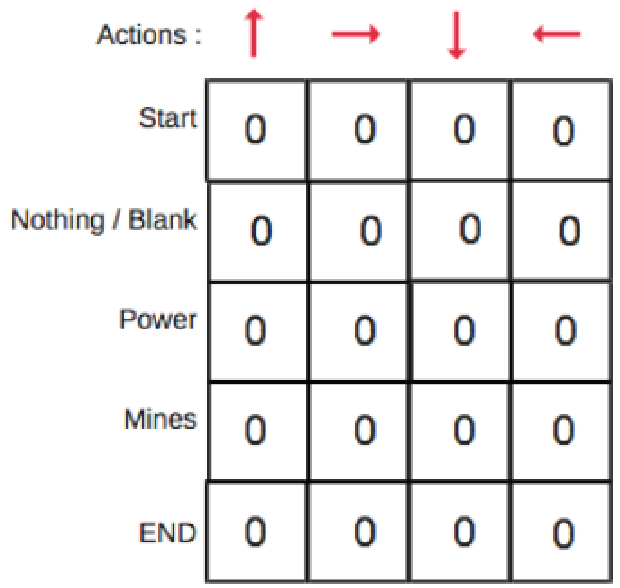
- Using the above function, we get the values of **Q** for the cells in the table.
- When we start, all the values in the Q-table are zeros.
- There is an iterative process of updating the values. As we start to explore the environment**,** the Q-function gives us better and better approximations by continuously updating the Q-values in the table.
- Now, let’s understand how the updating takes place.

## Q-LEARNING ALGORITHM PROCESS



## 1. INITIALIZE THE Q-TABLE

- There are n columns, where n= number of actions. There are m rows, where m= number of states. We will initialize the values at 0.



| Actions : | ↑ | → | ↓ | ← |
|---|---|---|---|---|
| Start | 0 | 0 | 0 | 0 |
| Nothing / Blank | 0 | 0 | 0 | 0 |
| Power | 0 | 0 | 0 | 0 |
| Mines | 0 | 0 | 0 | 0 |
| END | 0 | 0 | 0 | 0 |

four actions (a=4) and five states (s=5).

## 2&3. CHOOSE AND PERFORM AN ACTION

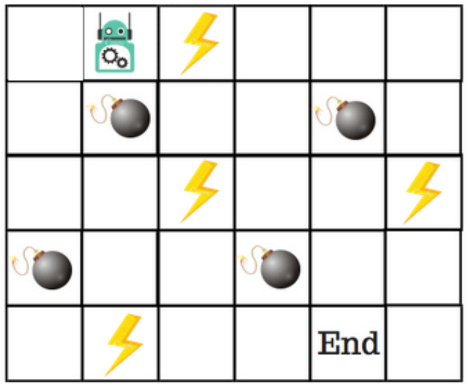
- Choose an action (a) in the state (s) based on the Q-Table. But, as mentioned earlier, when the episode initially starts, every Q-value is 0.
- So now the concept of exploration and exploitation trade-off comes into play.

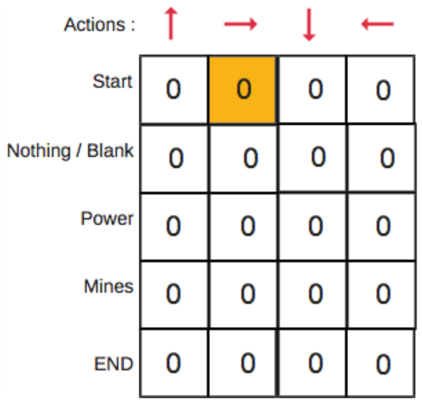
## CONT...

- In the beginning, the epsilon rates will be higher. The robot will explore the environment and randomly choose actions. The logic behind this is that the robot does not know anything about the environment.
- As the robot explores the environment, the epsilon rate decreases and the robot starts to exploit the environment.
- During the process of exploration, the robot progressively becomes more confident in estimating the Q-values.

## CONT…

- **For the robot example, there are four actions to choose from**: up, down, left, and right. We are starting the training now — our robot knows nothing about the environment. So the robot chooses a random action, say right.



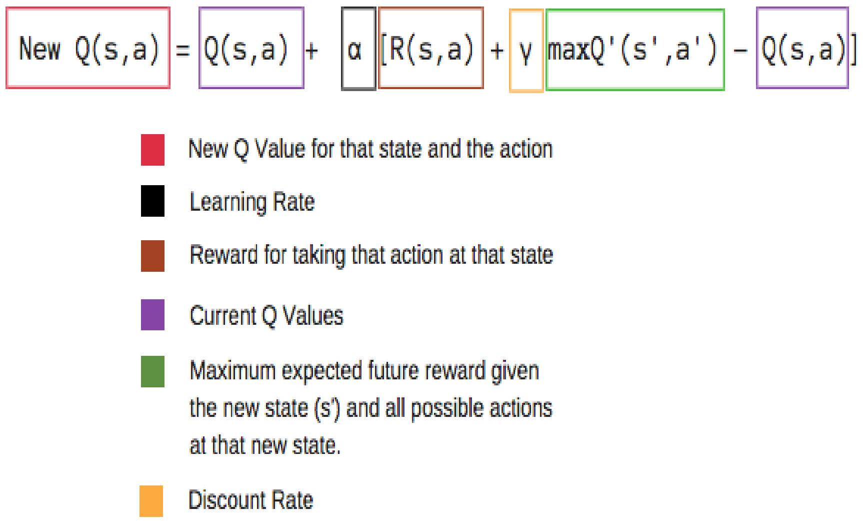


| Actions : | ↑ | → | ↓ | ← |
|---|---|---|---|---|
| Start | 0 | 0 | 0 | 0 |
| Nothing / Blank | 0 | 0 | 0 | 0 |
| Power | 0 | 0 | 0 | 0 |
| Mines | 0 | 0 | 0 | 0 |
| END | 0 | 0 | 0 | 0 |

We can now update the Q-values for being at the start and moving right using the Bellman equation.

## 4&5: EVALUATE

- Now we have taken an action and observed an outcome and reward. We need to update the function Q(s,a).



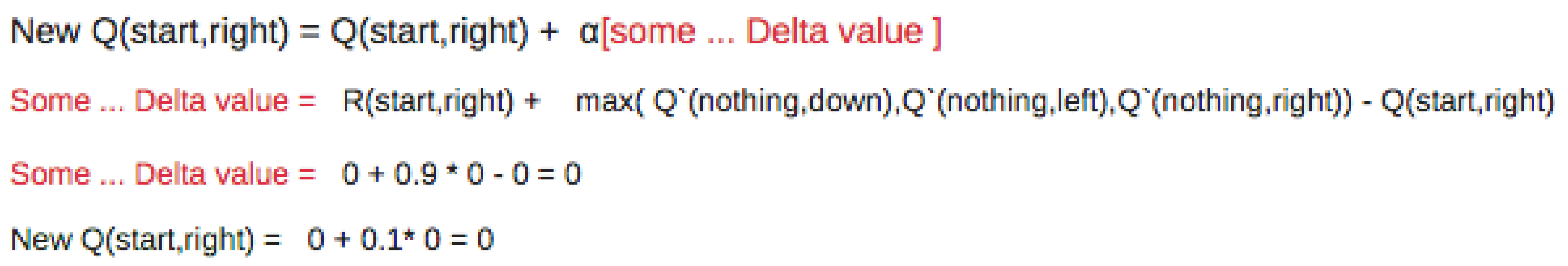
$$\text{New } Q(s,a) = Q(s,a) + \alpha\,[R(s,a) + \gamma\, \max Q'(s',a') - Q(s,a)]$$

- 🟥 New Q Value for that state and the action
- ⬛ Learning Rate
- 🟫 Reward for taking that action at that state
- 🟪 Current Q Values
- 🟩 Maximum expected future reward given the new state (s') and all possible actions at that new state.
- 🟧 Discount Rate

➢ In the case of the robot game, to reiterate the scoring/reward structure is:

- **power** = +1
- **mine** = -100
- **end** = +100

## CONT…


New Q(start,right) = Q(start,right) + α[some ... Delta value ]

Some ... Delta value = R(start,right) + max( Q`(nothing,down),Q`(nothing,left),Q`(nothing,right)) - Q(start,right)

Some ... Delta value = 0 + 0.9 * 0 - 0 = 0

New Q(start,right) = 0 + 0.1* 0 = 0

Repeat this again and again until the learning is stopped. In this way the Q-Table will be updated.

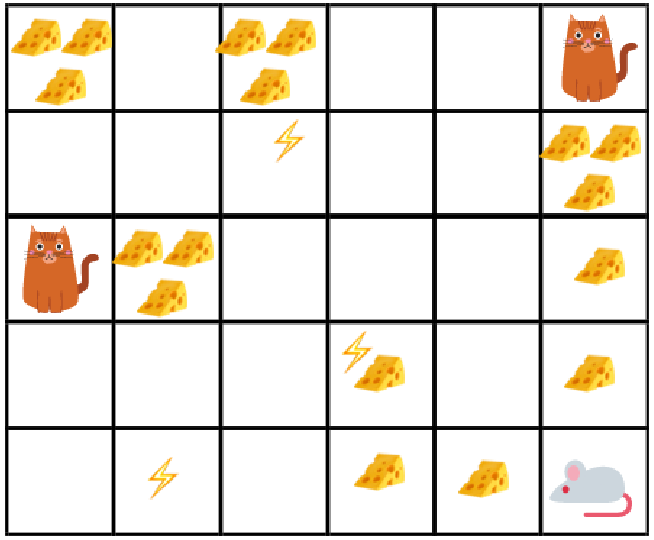
## EXPLORATION & EXPLOIATION

- Exploration is all about finding more information about an environment.
- The more we explore, the better we understand the world(eg. T and R)
- Exploration helps the agent for long-term beneficial.
- Exploitation is exploiting already known information to maximize the rewards.
- Based on what we know about the world, we can take actions with the aim of get highest reward.
- Exploitation helps the agent maximize its short and medium-term reward.

## EXAMPLE-1

- Say you go to the same restaurant every day. You are basically **exploiting.** But on the other hand, if you search for new restaurant every time before going to any one of them, then it’s **exploration**. Exploration is very important for the search of future rewards which might be higher than the near rewards.

## EXAMPLE-2



- In given picture, robotic mouse can have a good amount of small cheese (+0.5 each). But at the top of the maze there is a big sum of cheese (+100). So, if we only focus on the nearest reward, our robotic mouse will never reach the big sum of cheese – it will just exploit.
- But if the robotic mouse does a little bit of exploration, it can find the big reward i.e. the big cheese.