Intro to

Reinforcement Learning

Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

In simpler words, it's a way to allow machines to learn through experience and provide useful inferences, like humans do, only much faster.

Types of ML

Supervised learning (Task-driven , like classification, truth values available)

Unsupervised Learning (Data-driven, clustering, truth values absent)

Reinforcement learning (What you're here for !)

WHAT IS REINFORCEMENT LEARNING ?

It is a form of Machine learning where the learning agent interacts with the environment in an effort to learn the best way to do a certain task.

It is perhaps, the most intuitive form of machine learning, as all humans and even animals learn to perform various actions through a trial-error reward-punishment mechanism , for example, babies learning to walk or birds learning to fly.

Elements of RL

Policy

This defined how our agent behaves for a given time.
It is simply a map from the states the agent may find itself in to the actions it plans to perform in those states.

Reward

Reward function is used to define a goal in a reinforcement learning problem.
A reward function is a function that provides a numerical score based on the state of the environment

Elements of RL

Value

The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. This ties with the "Q-Value" that Srishti will tell you about in upcoming slides.

Model of the environment

The agent has sensors to detect everything that happens in the environment, in a given state, and can perform an action to change the state.

Steps for RL

Let's take a simple look at how Reinforcement learning might work with the example of a dog at the OAT trying to get food.

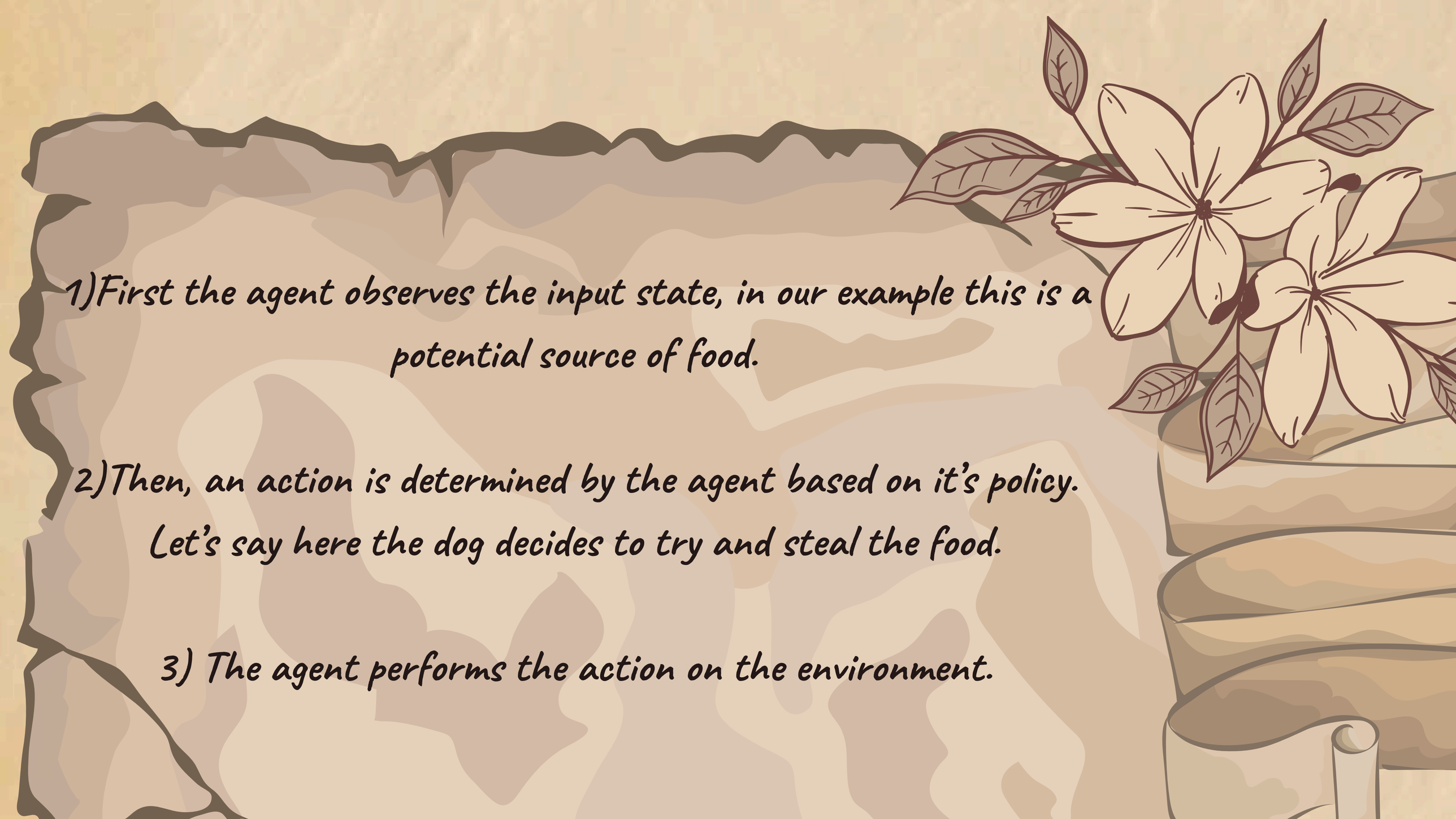
Here the ultimate goal of the agent (the dog) is to get as much food as possible.

Let's say in a given state, the dog has 3 options to try and get food - beg, aggressively bark or silently steal.

1)First the agent observes the input state, in our example this is a potential source of food.

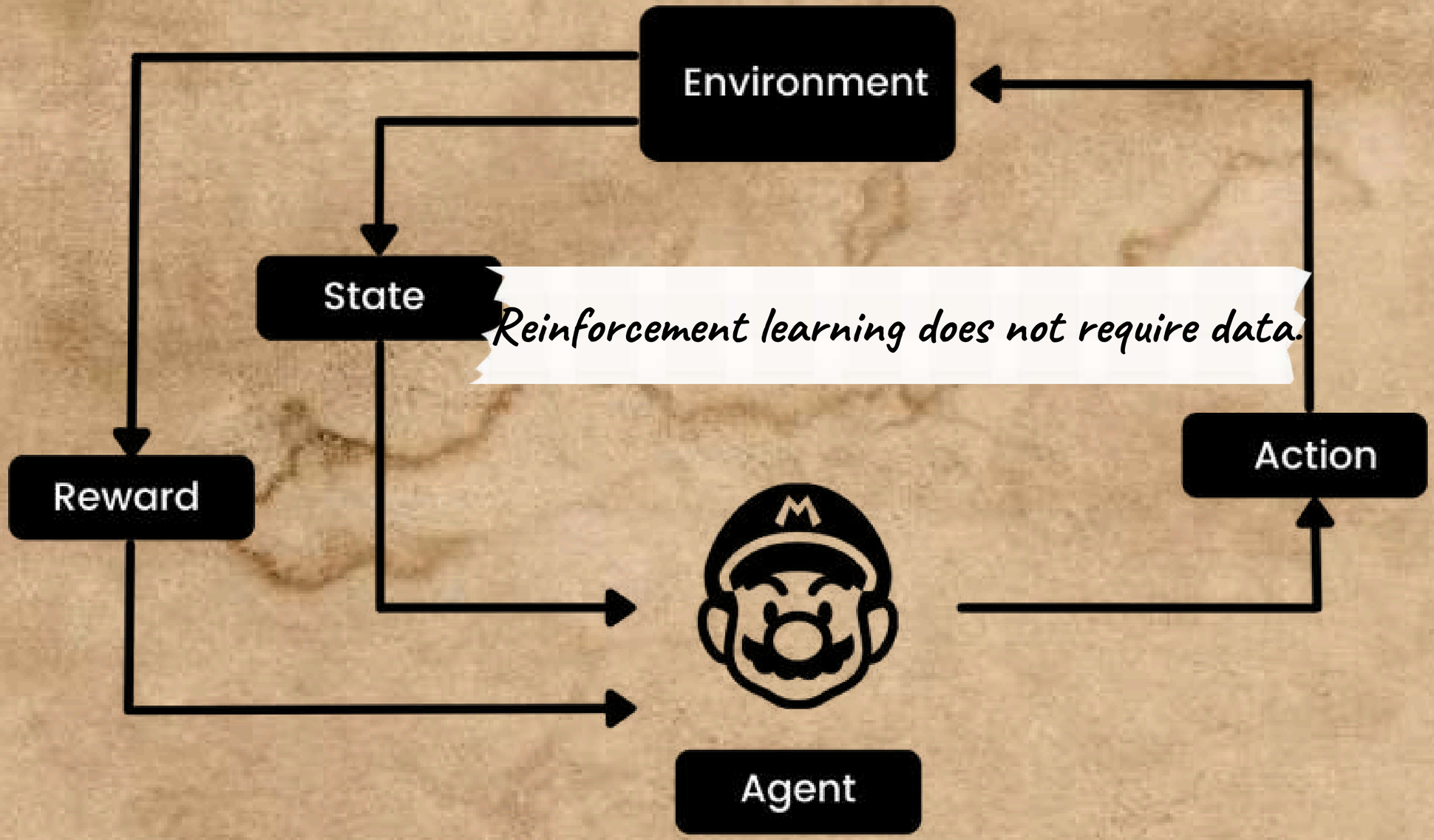
2)Then, an action is determined by the agent based on it's policy. Let's say here the dog decides to try and steal the food.

3) The agent performs the action on the environment.



4) The agent receives some scalar reward or reinforcement from the environment. Here let's say the dog is caught and shooed away.

5) Information about the given reward given for that state/action is recorded, so now the dog's policy is updated to value that action lesser for this state in future rounds of training.



Reinforcement learning does not require data.

what is Q-learning?

The “Q” stands for quality. Quality represents how valuable the action is in maximizing future rewards.
It will find the best series of actions based on the agent’s current state.

it is a type of reinforcement learning that is

model-free

- learn the consequences of their actions through the experience without transition and reward function.

value-based

- The value-based method trains the value function to learn which state is more valuable and take action.

off-policy algorithm

- In the off-policy, the agent learns from its own actions and doesn’t depend on the current policy.

Terms to know without which you'll be just as clueless as your agent if not more

States(s): the current position of the agent in the environment.

Action(a): a step the agent takes in a particular state.

Rewards: for every action, the agent receives a reward and penalty.

Terms to know without which you'll be just as clueless as your agent if not more

Q-value: it determines how good an action (A) is, taken at a particular state (S)

Episodes: the end of the stage, where agents can't take new action. It happens when the agent has achieved the goal or failed.

New Q(St, a): expected optimal Q-value of doing the action in a particular state.

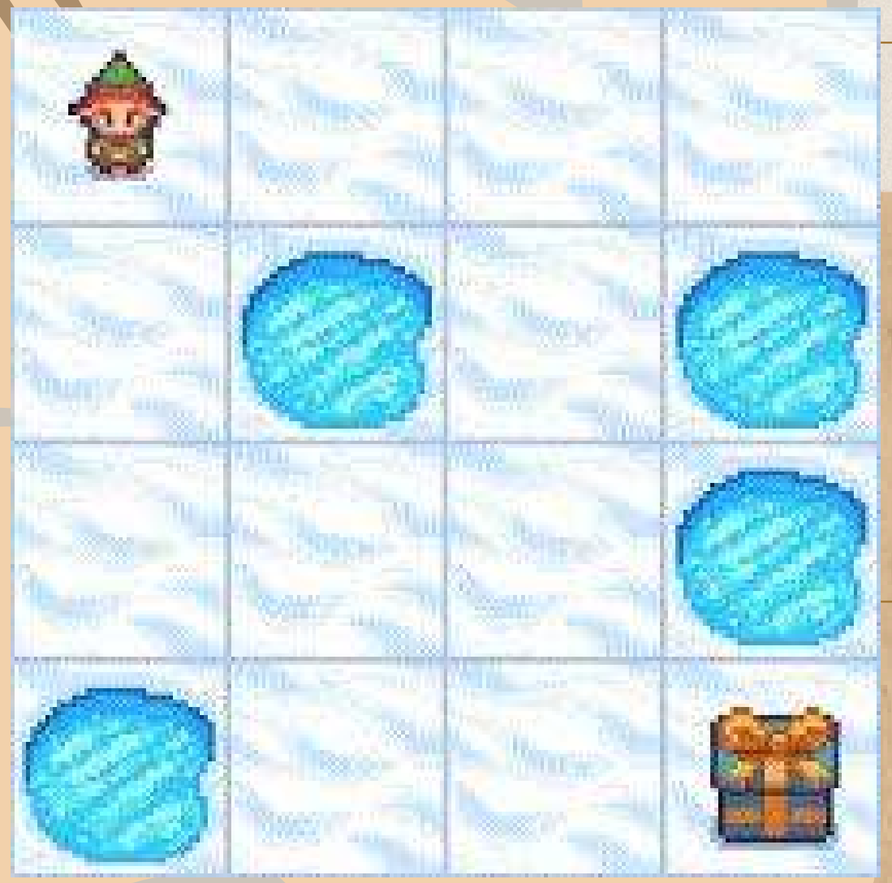
Terms to know without which you'll be just as clueless as your agent if not more

Q(St, At): it is the current estimation of New Q(St, 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 New Q(St, a) by using the current state and action and previous state and action.

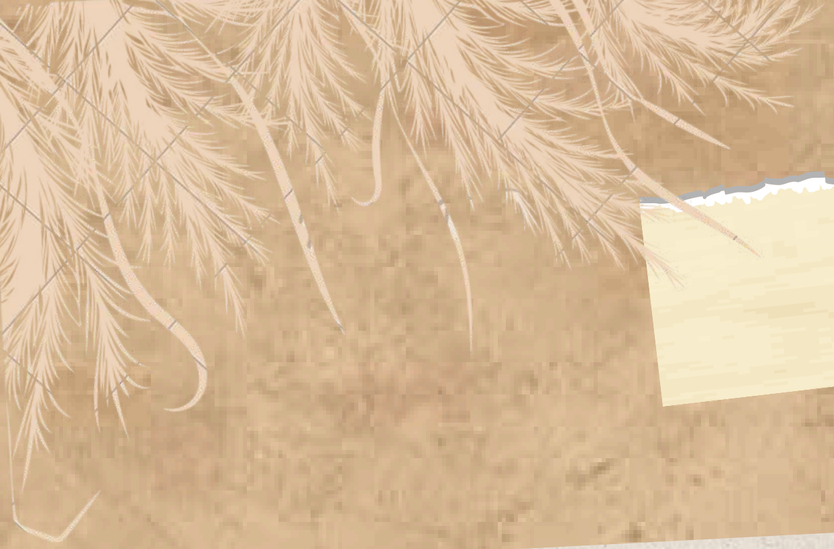
You wouldn't wanna swim in a frozen lake in Kanpur's winter so, better save yourself



How did you decide which path to follow to get yourself the present and not become the past ?

because

you took the best possible decision at every step to reach the outcome.





WHAT IS A Q-VALUE?

- A Q-value indicates the quality of a particular action **a** in a given state **s**: Q(s,a)
- Q-values are our current estimate of the possible future rewards that can be accumulated through all the remaining steps in the current episode if the AI agent takes action **a** in the current state **s**.
- Q- values become larger as we approach the desired goal.

THE NUMBER OF STATES IS FINITE AND THE NUMBER OF ACTIONS ARE FINITE.

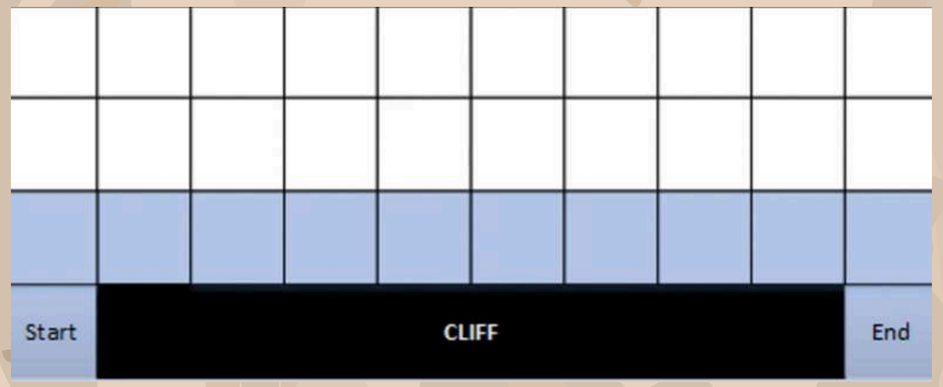
FOR EXAMPLE: CONSIDER THE FAMOUS CLIFF-WALKING EXAMPLE.

All the states can be written as an ordered pair (x,y) example (0,0) (2,0) (1,2) etc.

And for every state, we have 4 actions: Up, Down, Right, and Left.

	0	1	2	3
0	Up: -4.10 Right: -3.44 Down: -3.44 Left: -4.10	Up: -3.44 Right: -2.71 Down: -2.71 Left: -4.10	Up: -2.71 Right: -1.90 Down: -1.90 Left: -3.44	Up: -1.90 Right: -1.90 Down: -1.00 Left: -2.71
1	Up: -4.10 Right: -2.71 Down: -4.10 Left: -3.44	Up: -3.44 Right: -1.90 Down: -100.00 Left: -3.44	Up: -2.71 Right: -1.00 Down: -100.00 Left: -2.71	Up: -1.90 Right: -1.00 Down: 0.00 Left: -1.90
2	Up: -3.44 Right: -100.00 Down: -4.10 Left: -4.10	CLIFF		Up: 0.00 Right: 0.00 Down: 0.00 Left: 0.00

START END



Q-learning works similarly, it tries to find the best possible action at a state, this is the Q-value for that action-state pair.

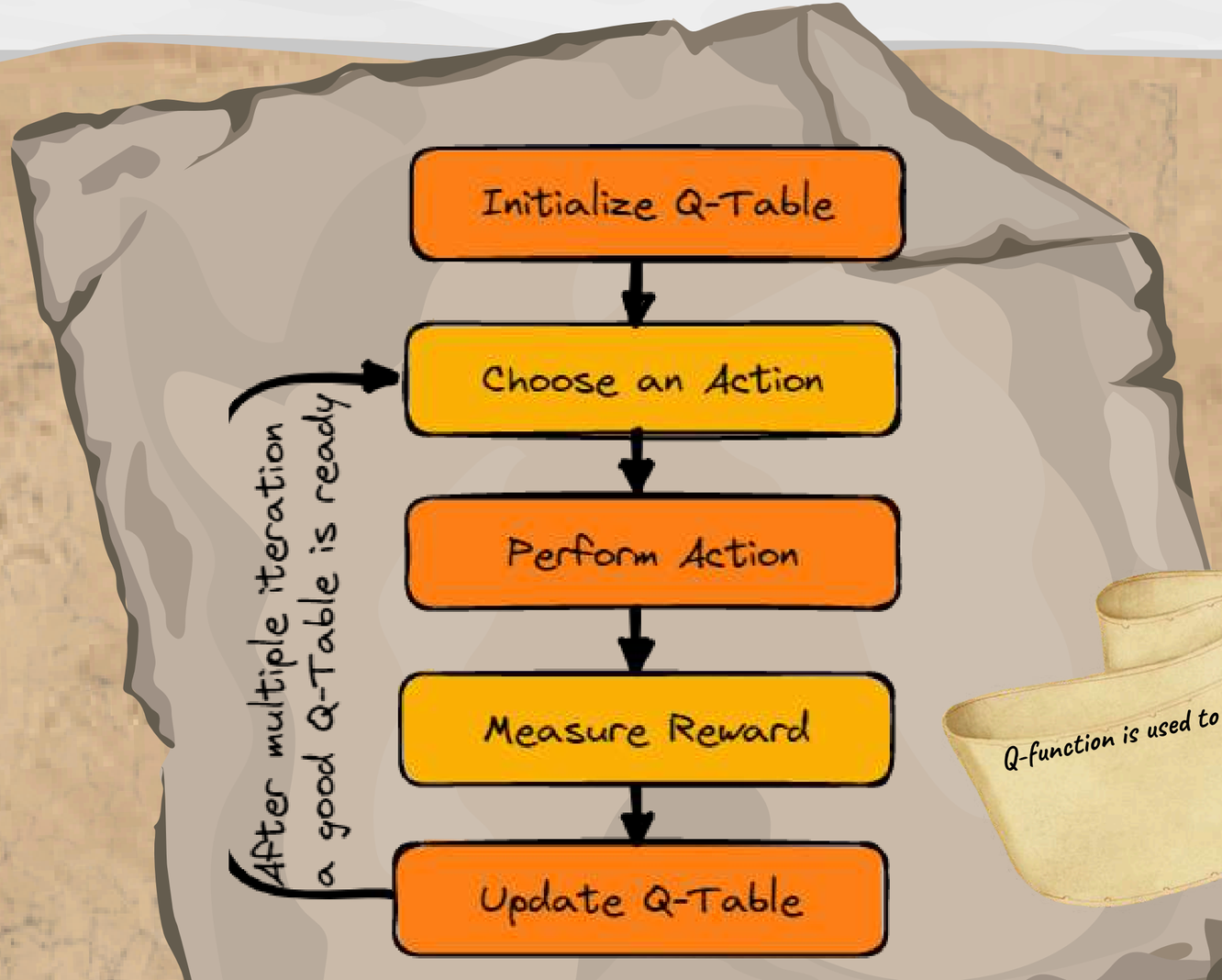
The Q-value for every action-state pair is stored in a data structure called the Q-table.

The Q-table guides the agent to choose the action with the highest Q-value for a given state, thus leading best long-term rewards.

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.

At the start, the agent will choose to take the random action, and on the second run, it will use an updated Q-Table to select the action.

Q-learning algorithm for updating the Q-table



Q-function is used to update the Q-table after the outcome and the reward

Q-TABLE

Possible Actions

Possible States (Row, Column)	Up	Right	Down	Left
0, 0	-4.10	-3.44	-3.44	-4.10
0, 1	-3.44	-2.71	-2.71	-4.10
0, 2	-2.71	-1.90	-1.90	-3.44
0, 3	-1.90	-1.90	-1.00	-2.71
1, 0	-4.10	-2.71	-4.10	-3.44
1, 1	-3.44	-1.90	-100.00	-3.44
1, 2	-2.71	-1.00	-100.00	-2.71
1, 3	-1.90	-1.00	0.00	-1.90
2, 0	-3.44	-100.00	-4.10	-4.10
2, 1	0.00	0.00	0.00	0.00
2, 2	0.00	0.00	0.00	0.00
2, 3	0.00	0.00	0.00	0.00

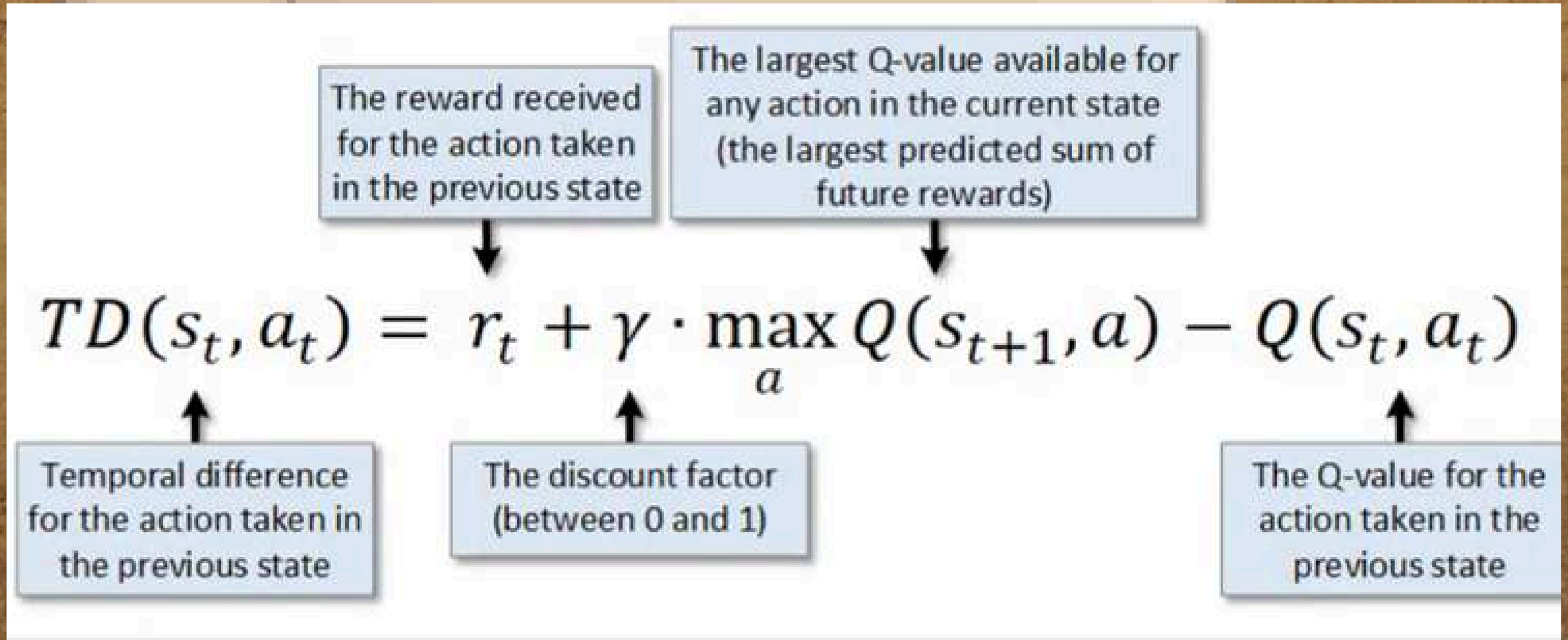
How do we choose the action?

The actions are chosen using the epsilon greedy algorithm.

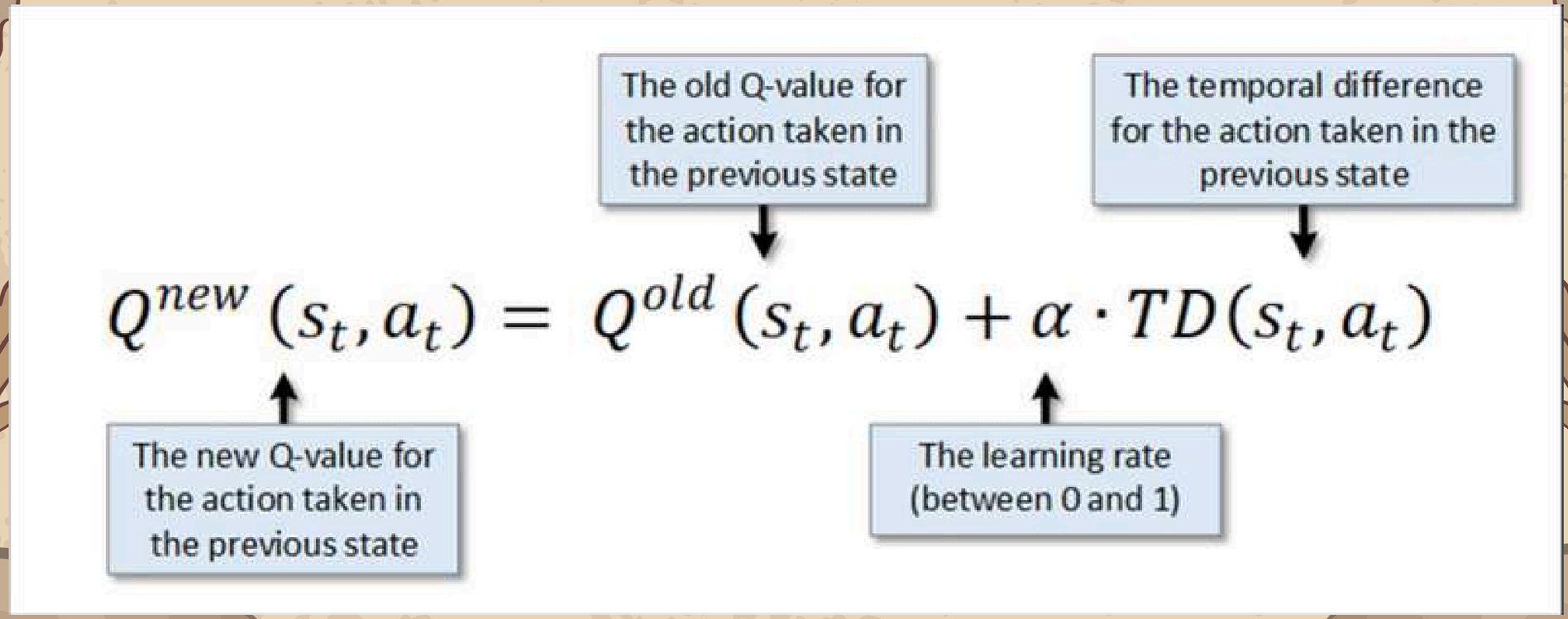
This algorithm will usually will usually choose the most promising action for the AI agent, but it will occasionally choose a less promising option in order to encourage the agent to explore the environment

HOW TO UPDATE THE Q TABLE?

TEMPORAL DIFFERENCES



BELLMAN EQUATION



We train the AI agent across a very large number of episodes. In each episode, the AI agent takes multiple actions and the Q-table is updated after each episode. The episode terminates after the AI agent reaches the target or stumbles upon a forbidden area.
(In our case: falls off the cliff)

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 to determine the best course of action.

examples .....

- Using Q-learning , we can make an Ad recommendation system which will suggest related products to our previous purchases or websites we may have visited. The reward will be if user clicks on the suggested product.
- It can be used to train agents to play a wide variety of games, from simple grid worlds to complex video games, as long as the problem can be described as a sequential decision-making process with states, actions, and rewards, where the future depends mainly on the current state and chosen actions.

We didn't tell you to bring your laptop for no reason

Let's get our hands dirty, let's code

Amazon Warehouse.ipynb - Colaboratory (google.com)

Thank You