SYLLABUS

Module 1: Introduction to Reinforcement Learning and Multi Armed Bandits

- Introduction to Reinforcement Learning
- Basic Elements of RL and Value Functions
- Model of the Environment
- Exploration vs Exploitation
- Multi Armed Bandits
- Action Value Methods
- Epsilon Greedy Policy
- A simple Bandit Algorithm
- Second Action Selection Strategy

Module 2: Markov Decision Process

- Introduction to Markov Decision Process
- Recycling Robot
- Pole Balancing
- Policy and Value Functions
- Optimal Policies and Optimal Value Functions
- Bellman Equation for Recycling Robot

Module 3: Dynamic Programming (Value and Iteration)

- Introduction to Dynamic Programming
- Simple Analysis
- Policy Improvement Step
- Policy Iteration Procedure
- Value Iteration Procedure
- Geometric Interpretation of Policy Iteration an Iteration

Module 4: Monte-Carlo Methods (On-policy ar Algorithms)

- Introduction to Monte Carlo Techniques
- Monte Carlo Algorithm & Prediction
- Monte Carlo Control
- Alternative to Exploring starts
- Off Policy methods for prediction
- Incremental Implementation
- Incremental Update Algorithm

Module 5: Temporal Difference Learning

- Temporal Difference Introduction
- TD(0) Prediction
- SARSA
- Q Learning

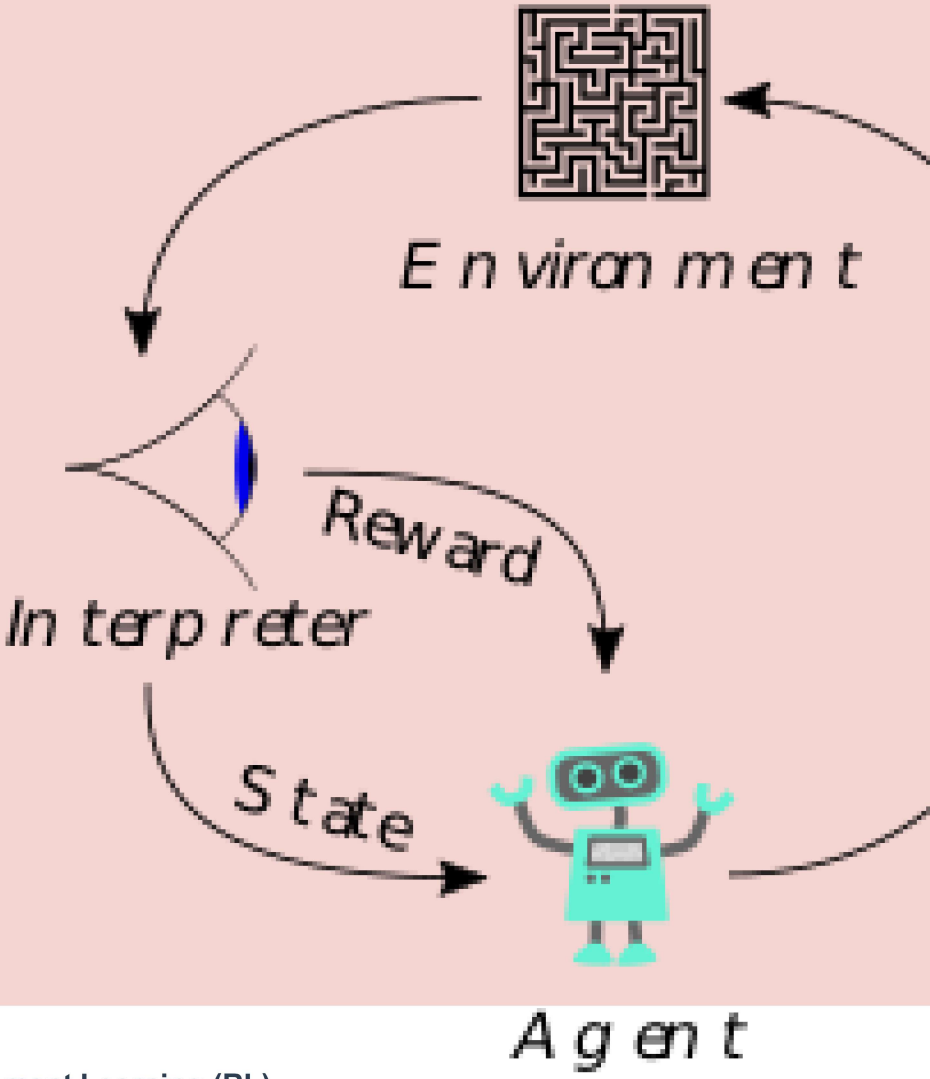
MODULE – 1

1.1 Introduction to Reinforcement Learning

1.1.1 Elements of RL
1.1.2 Fundamentals of RL Concepts - Examples
1.1.3 State, Action, Environment
1.1.4 Model based Environment
1.1.5 Model free Environment
1.1.6 Exploration
1.1.7 Exploitation
1.1.8 Contrast and Similarities

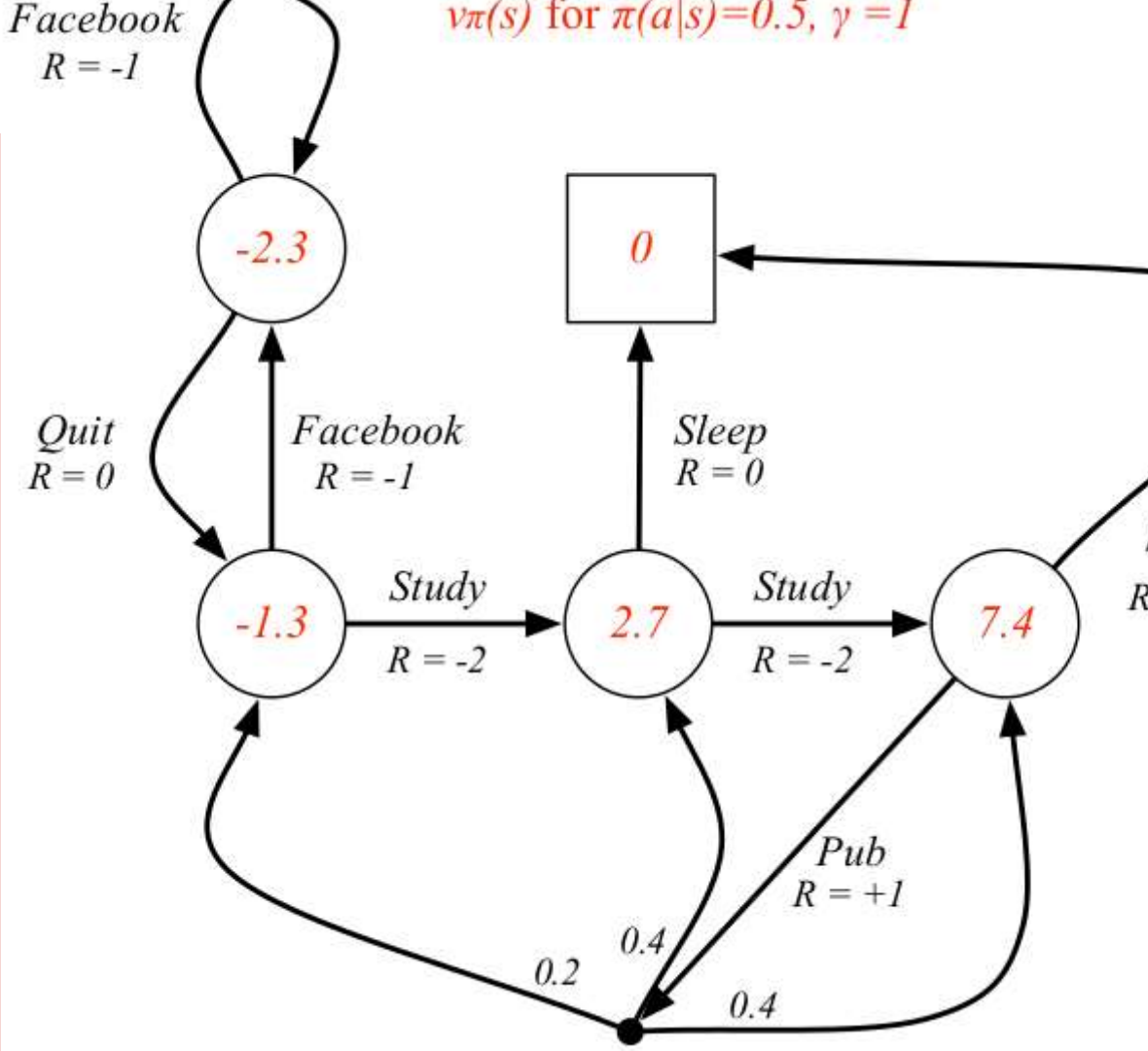
1.2 Multi-Armed Bandit (MAB)

1.2.1 An n-armed Bandit Problem
1.2.2 Action Value Methods
1.2.3 Epsilon Greedy Policy
1.2.4 Softmax Action Selection
1.2.5 Evaluation Versus Instruction
1.2.6 Incremental Implementation
1.2.7 Stationary Process
1.2.8 Non-Stationary Process
1.2.9 Comparison with RL
1.2.10 A simple Bandit Algorithm
1.2.11 Contextual Bandits : Action, states, rewards



MODULE – 2

2.1 Basic MDP
2.1.1 The Markov Property
2.1.2 Policy and Value Functions
2.1.3 Optimal Value functions
2.1.4 Optimality and Approximations
2.1.5 Finite MDP : Recycling Robot
2.1.6 Transition graph for recycling robot
2.1.7 Returns : Episodic Tasks
2.1.8 Returns : Continuing Tasks
2.1.9 MDP Policy Evaluations
2.1.10 Bellman Equation for Recycling Robot
2.2 Q Learning
2.2.1 Construction of Q-Table : How to?
2.2.2 Algorithm: Action, State, Learning rate, reward
2.2.3 Convergence: Q-values to optimal values
2.3 Model Free Algorithm
2.3.1 Descriptions
2.3.2 Vanilla Q-Learning
2.3.3 Deep Q-Learning
2.3.4 Update the Q-table using the Bellman Equation
2.4 The Deep-Q Learning Network
2.4.1 Initialize Target and Main neural networks
2.4.2 How to map States to (Action, Q-value) pairs
2.4.3 Choose an action using the Epsilon-Greedy Exploration Strategy



MODULE – 3

3 Dynamic Programming

3.1.1 Simple Analysis
3.1.2 Policy Improvement Step
3.1.3 DP: Value Iteration Algorithm
3.1.4 Algorithm : Policy Iteration Procedure
3.1.5 Geometric Interpretation of Policy Iteration and Value Iterati
3.1.6 DP : Policy Evaluation
3.1.7 DP : Policy Improvement
3.1.8 DP: Policy Iteration
3.1.9 DP: Value Iteration Algorithm
3.1.10 DP in Action : Is DP applicable to all environments?

Bellman Equation:

$$Q^*(s,\ a) = \sum_{s'} P(s'|s,\ a)(R(s,\ a,\ s') + \gamma \max_{a'} Q^*(s',\ a'))$$

Q-Value iteration:

$$Q^*(s,\ a) \leftarrow \sum_{s'} P(s'|s,\ a)(R(s,\ a,\ s') + \gamma \max_{a'} Q^*(s',\ a'))$$

MODULE – 4

4.1 Introduction to Monte Carlo Techniques

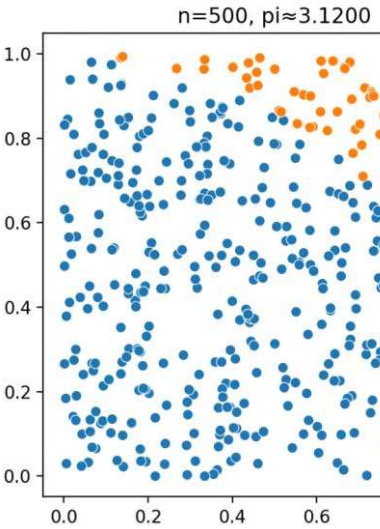
4.1.1 Understanding Monte Carlo Method
4.1.2 Prediction and Control tasks

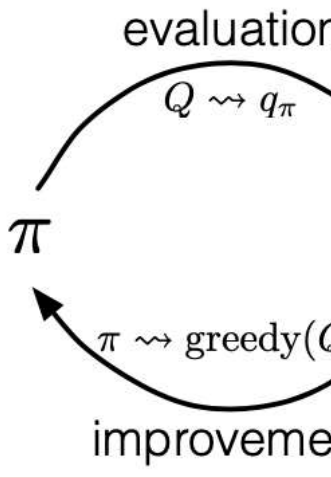
4.2 Monte Carlo Prediction

4.2.1 MC Prediction (Q Function)

4.3 MC Control Algorithms

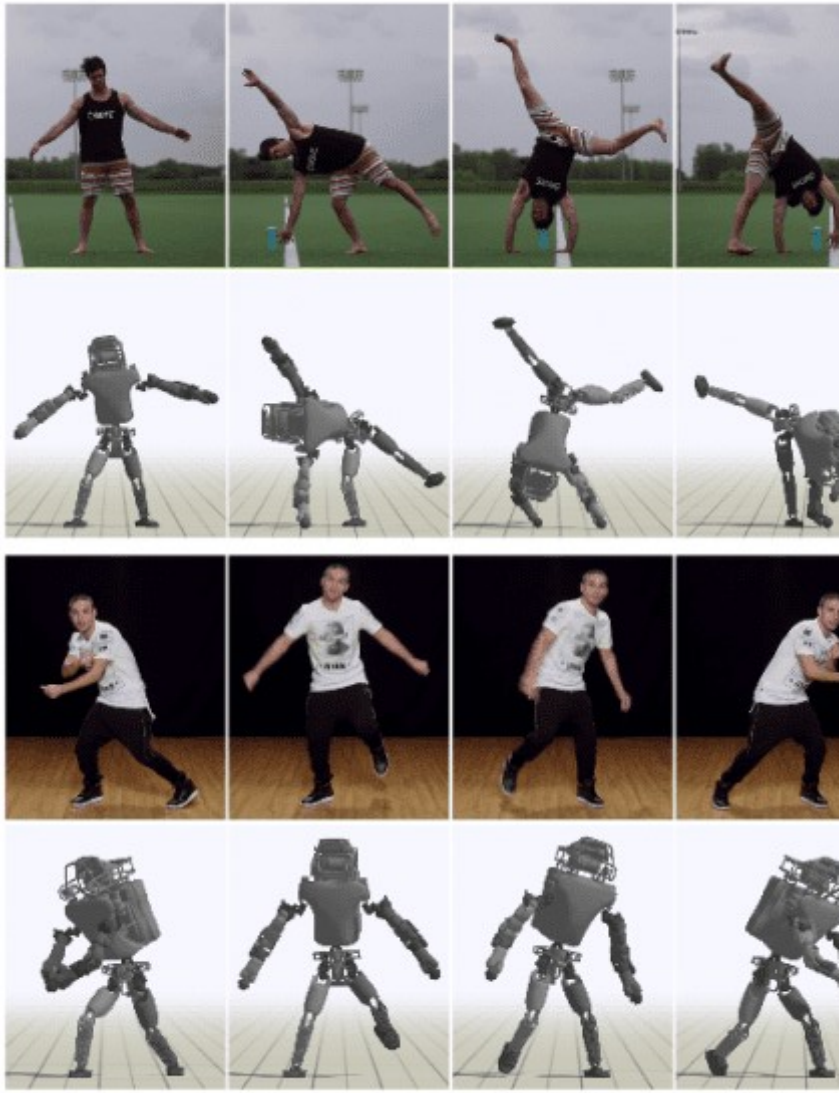
4.3.1 On - Policy MC Control
4.3.2 Off - Policy MC Control
4.3.3 Incremental Implementation
4.3.4 Evaluation Policy and Interpretation



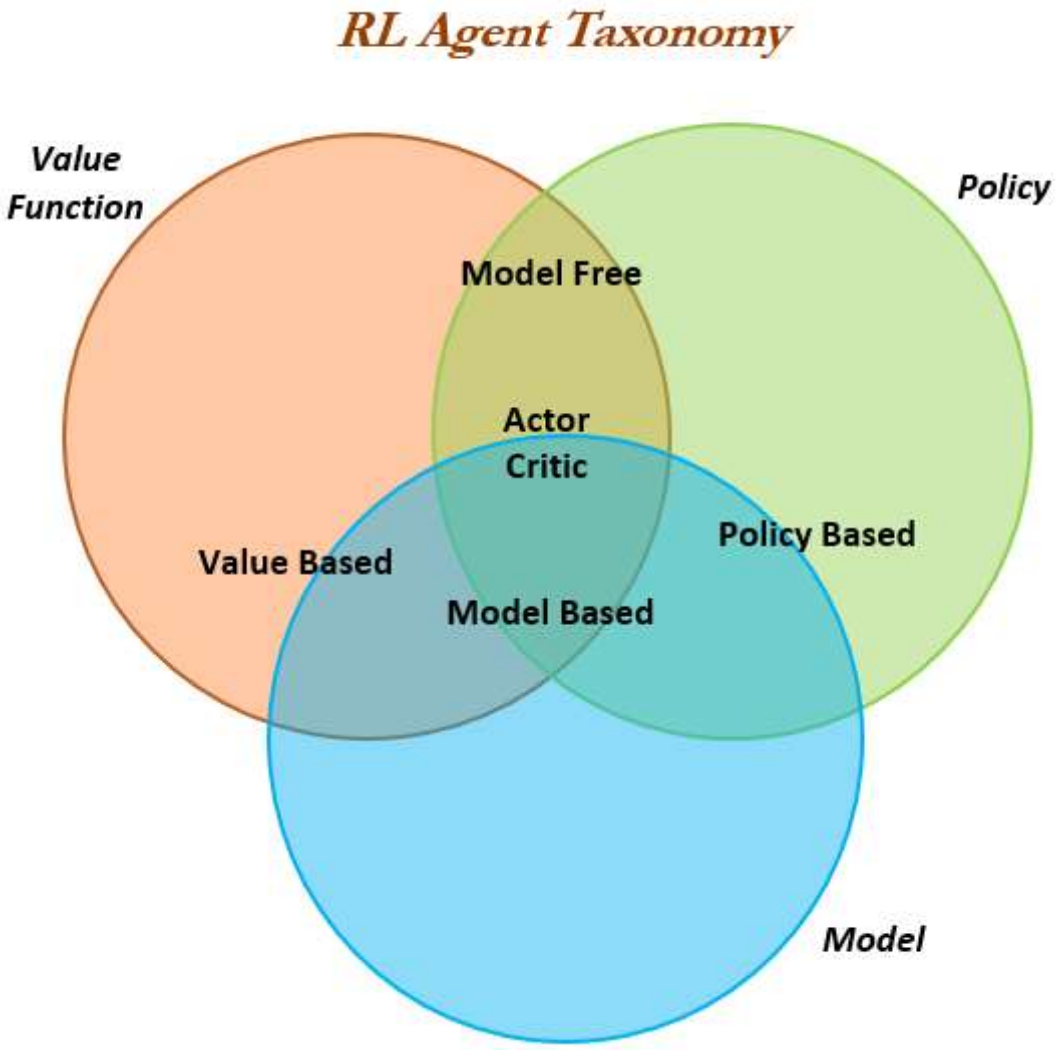


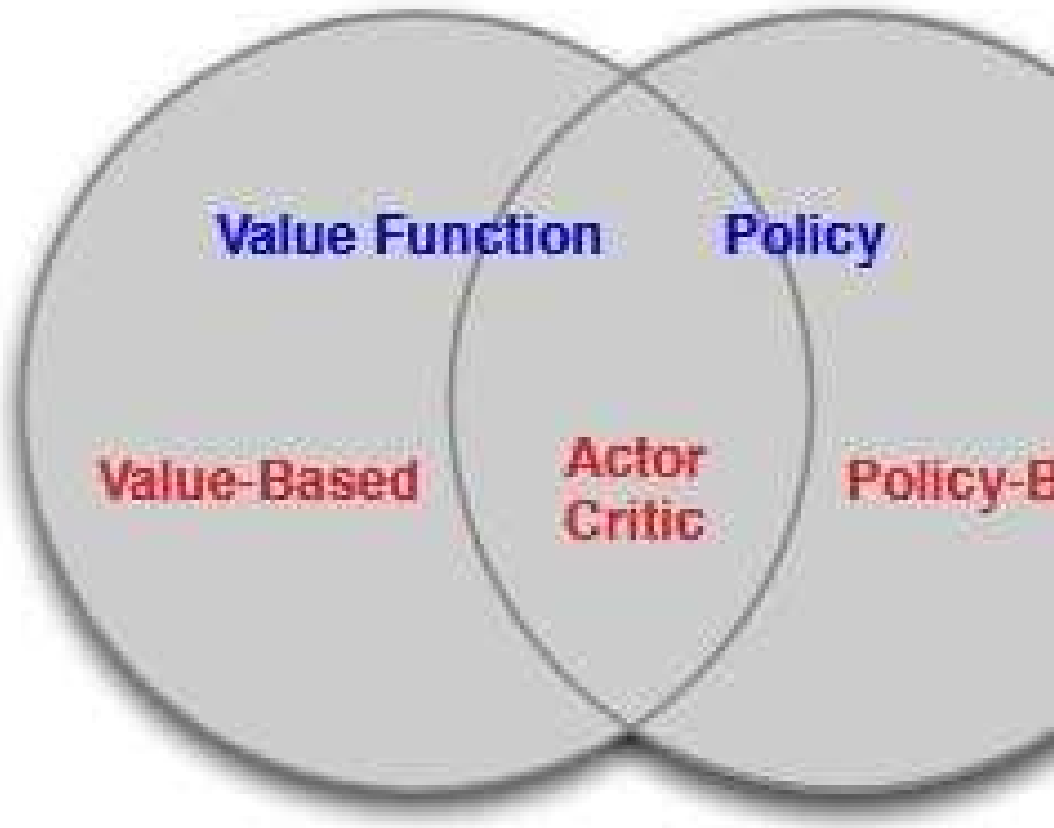
MODULE – 5

5.1 Introduction to Temporal Differencing
5.1.1 TD Learning - Concepts
5.1.2 TD Prediction Algorithm
5.2.1 On-Policy TD Control - SARSA
5.2 TD Control
5.2.2 Computing optimal policy : TD(0)
5.3 Off Policy TD Control
5.3.1 Computing optimal policy
5.3.2 Difference between Q learning and SARSA
Comparison of TD
5.4 Actor Critic Methods
5.5 Comparing DP ,MC, TD methods
5.6 Review of RL Techniques and Problem Statements

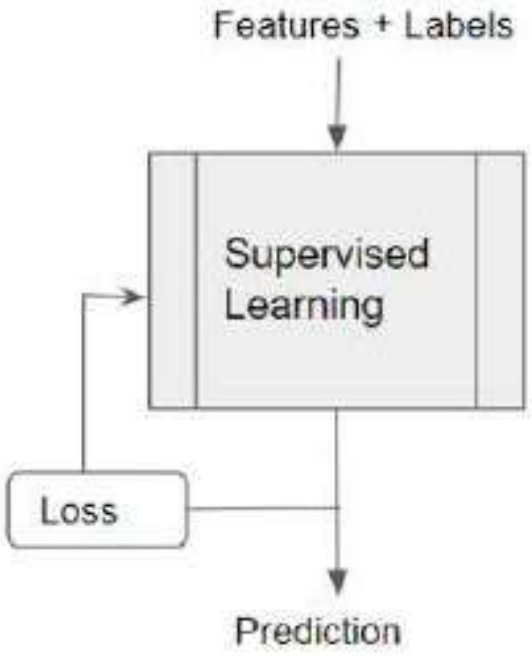


REINFORCEMENT LEARNING – BIRD'S EYE V

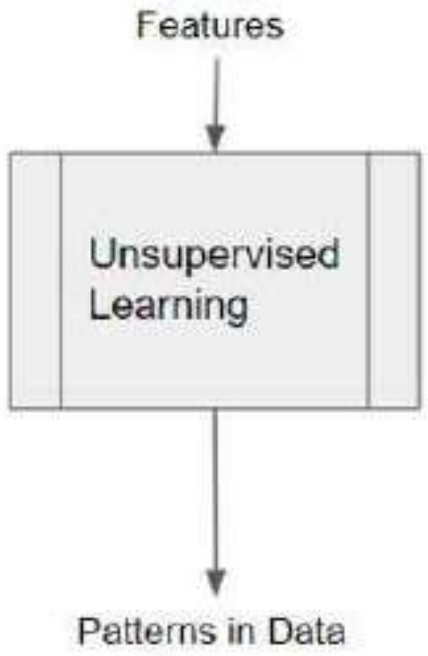




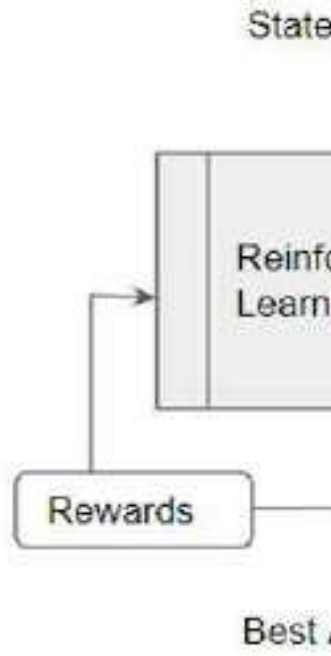
MACHINE LEARNING : A COMPARISON



Supervised learning uses labeled data as input, and predicts outcomes. It receives feedback from a loss acting as a 'supervisor'.



Unsupervised Learning uses data as input and detects hidden patterns in the data such as clusters or anomalies. It receives no feedback from a supervisor.



Reinforcement L
gathers inputs anc
feedback by intera
the external world.
the best action that
take while interactir
world.

SIMPLE EXAMPLE OF AGENT IN AN ENVIRONMI

State:

Map Locations

$\{< 0,0 >, < 1,0 > \cdots < 3,3 >\}$

Actions:

Move within map

Reaching chest ends episode

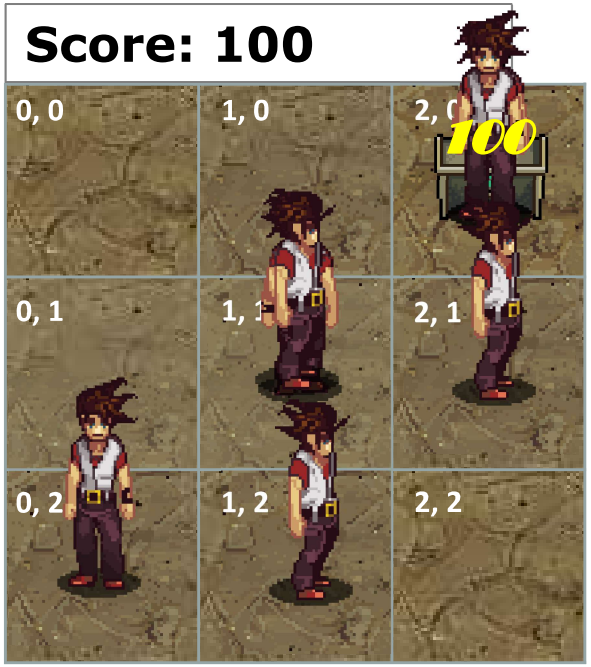
$A_{0,0} = \{ east, south \}$
$A_{1,0} = \{ east, south, west \}$
$A_{2,0} = \{ \phi \}$
...
$A_{2,2} = \{ north, west \}$

Reward:

100 at chest

0 for others

$R_{east}(< 1,0 >, < 2,0 >) = 100$
$R_{north}(< 2,1 >, < 2,0 >) = 100$
$R_*(*,*) = 0$



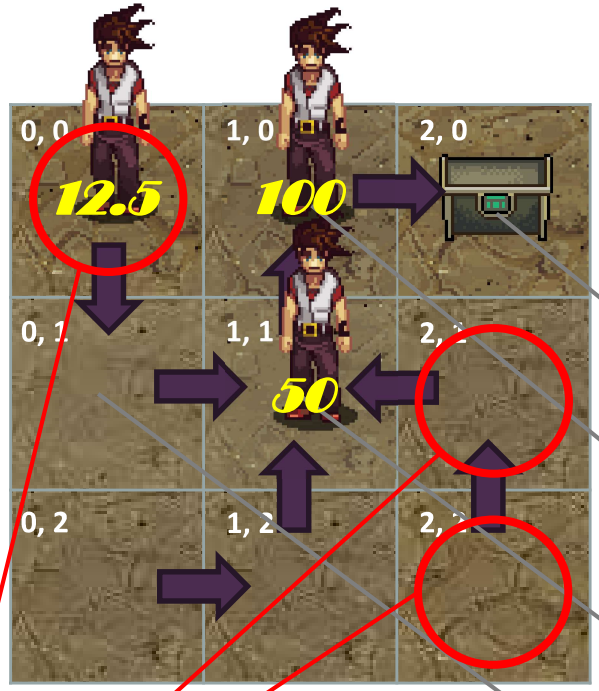
POLICIES

R_{east} $(< 1,0 >, < 2,0 >$
$R_{north}(< 2,1 >, < 2,0 >$
R_{*} $(*,*)$
$\gamma = 0.5$

Policy

$$\pi(s) = a$$

$\pi(< 0,0 >) = \{ south \}$
$\pi(< 0,1 >) = \{ east \}$
$\pi(< 0,2 >) = \{ east \}$
$\pi(< 1,0 >) = \{ east \}$
$\pi(< 1,1 >) = \{ north \}$
$\pi(< 1,2 >) = \{ north \}$
$\pi(< 2,0 >) = \{ \phi \}$
$\pi(< 2,1 >) = \{ west \}$
$\pi(< 2,2 >) = \{ north \}$



Policy could be better

Evaluating Poli

$$V^{\pi}(s) = \sum_{i=0}^{\infty} \gamma^{i} r$$

$V^{\pi}(< 1,0 >) = \gamma^{0} * 1$

$V^{\pi}(< 1,1 >) = \gamma^{0} * 0 + \gamma$

Move to <0,1> Move to <1,1> Move to <1,0>

$V^{\pi}(< 0,0 >) = \boxed{\gamma^{0} * 0} + \boxed{\gamma^{1} * 0} + \boxed{\gamma^{2} * 0} +$

Why Reinforcement Learning?

Learning from a small subset of actions will not help expand the vast realm of solutions that may work for a particular problem

Learn: To Walk

Why Reinforcement Learning?

This is going slow the growth that technology is capable of. Machines need to learn to perform actions by themselves and not just learn off humans

Objective: Climb Mountain

Important Terms in Reinforcement Learning

Agent

Agent is the model that is being trained via reinforcement learning

Important Terms in Reinforcement Learning

Environment

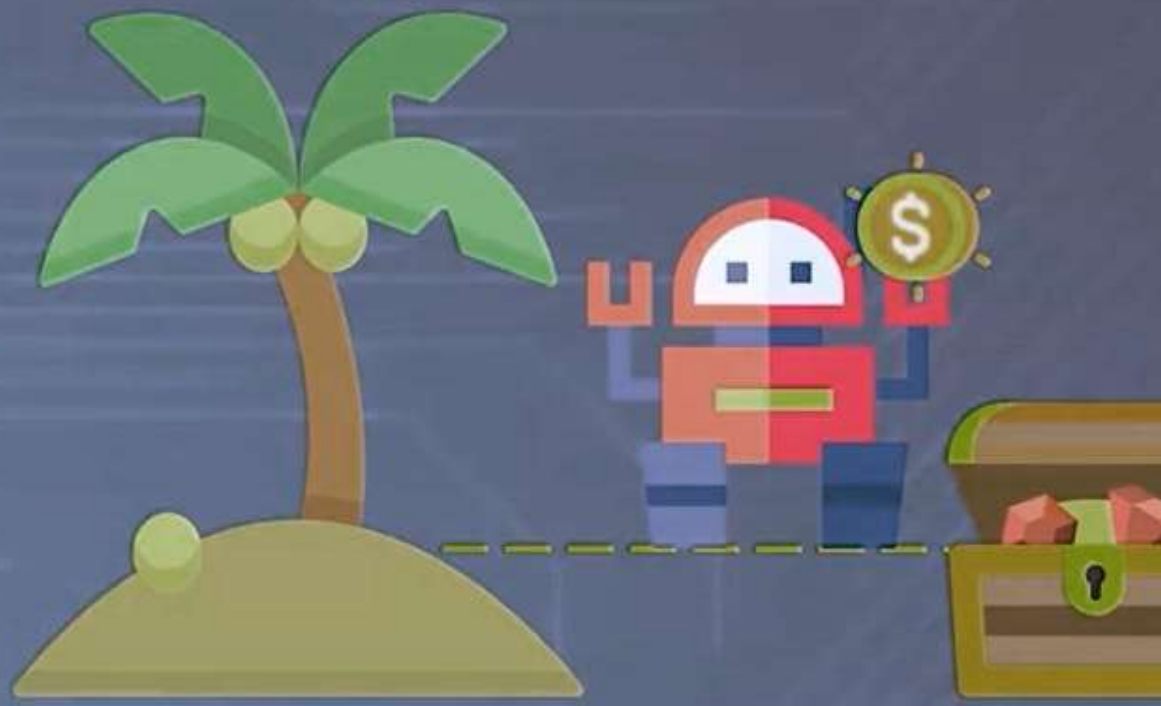
The training situation that the model must optimize to is called its environment

Outp

Important Terms in Reinforcement Learning

Reward

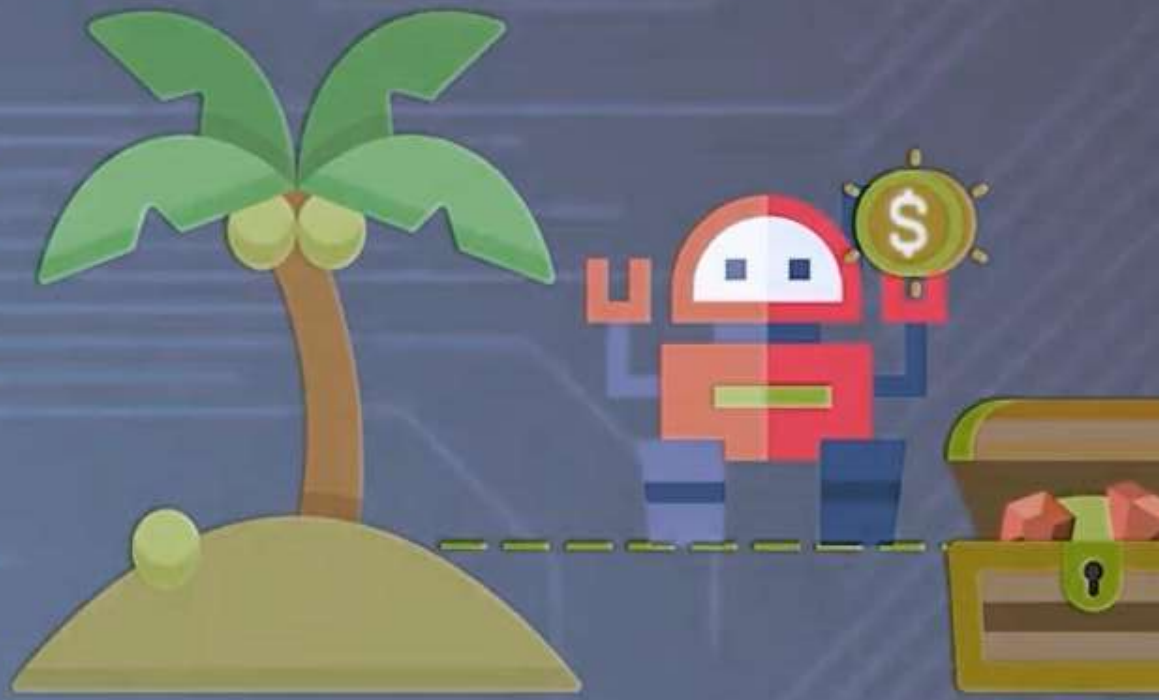
To help the model move in the right direction, it is rewarded/ points are given to it to appraise some action



Important Terms in Reinforcement Learning

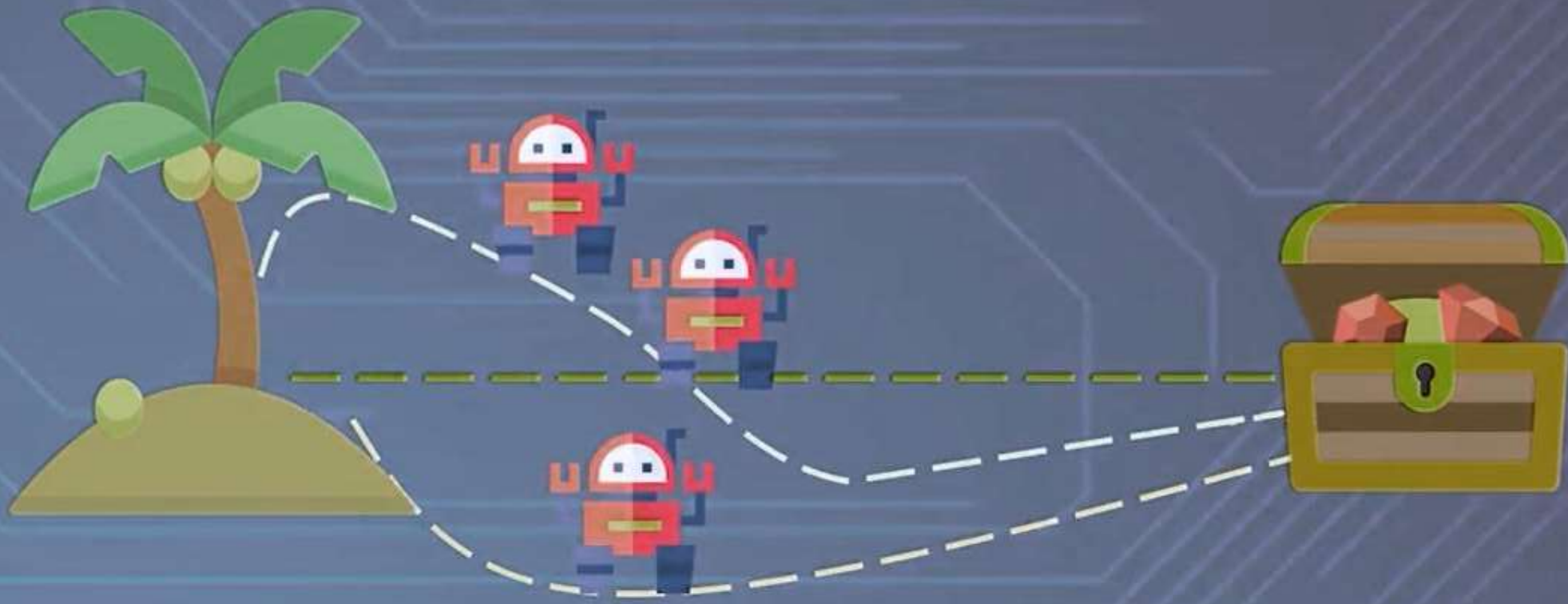
Policy

Policy determines how an agent will behave at anytime. It acts as a mapping between Action and present State



What is Reinforcement Learning?

It will know the best path by the time taken on each path. It might even come up a unique solution all by itself

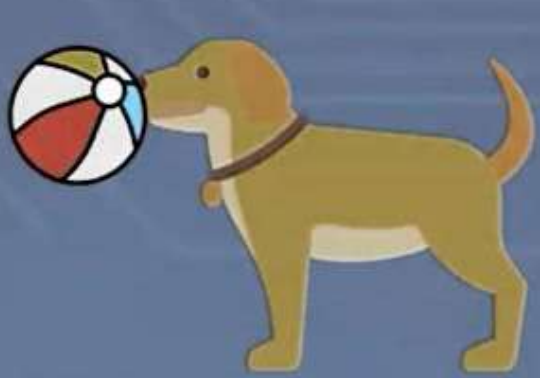


Supervised vs Unsupervised vs Reinforcement Learni

Supervised Learning	Unsupervised Learning	Reinforcement Learnin
Data provided is labeled data, with output values specified	Data provided is unlabeled data, the outputs are not specified, machine makes its own prediction	The machine learns from its environment using rewards ar errors
Used to solve Regression and classification problems	Used to solve Association and clustering problems	Used to solve Reward based problems
Labeled data is used	Unlabeled data is used	No predefined data is used
External Supervision	No supervision	No supervision
Solves problems by mapping labeled input to known output	Solves problems by understanding patterns and discovering output	Follows Trail and Error proble solving approach

Reinforcement Learning Example

The dog will follow a policy to maximize its reward and hence, will follow every command and might even learn a new action, like begging, by itself



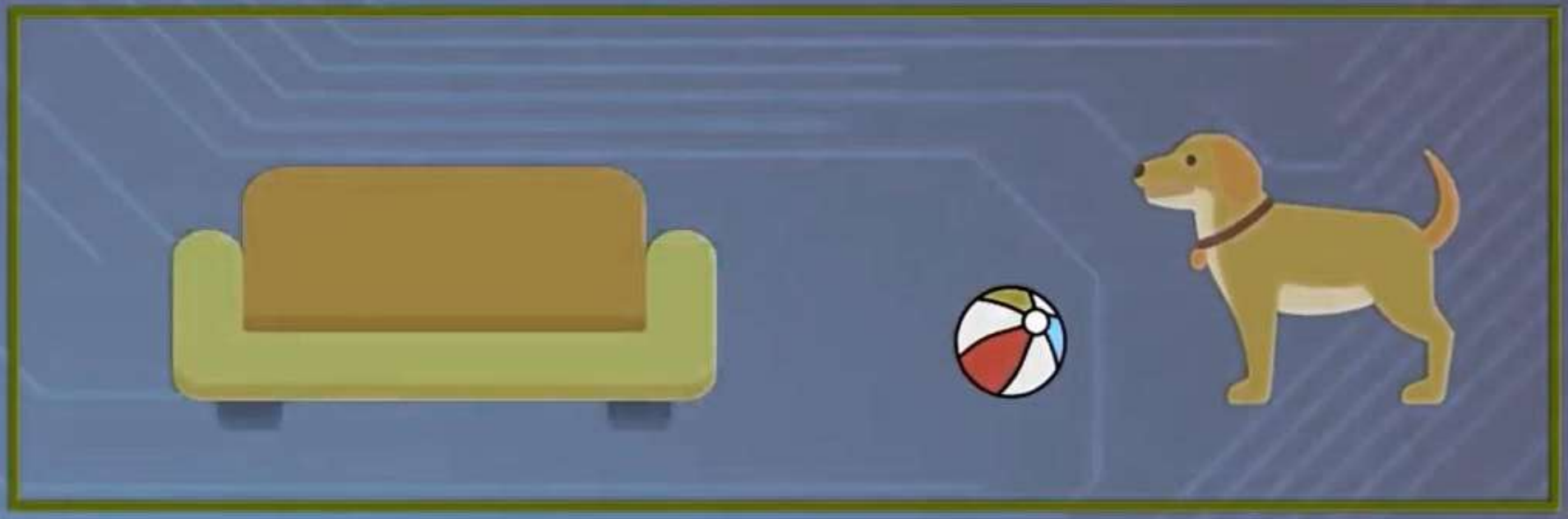
Fetching

Handshake

Begging

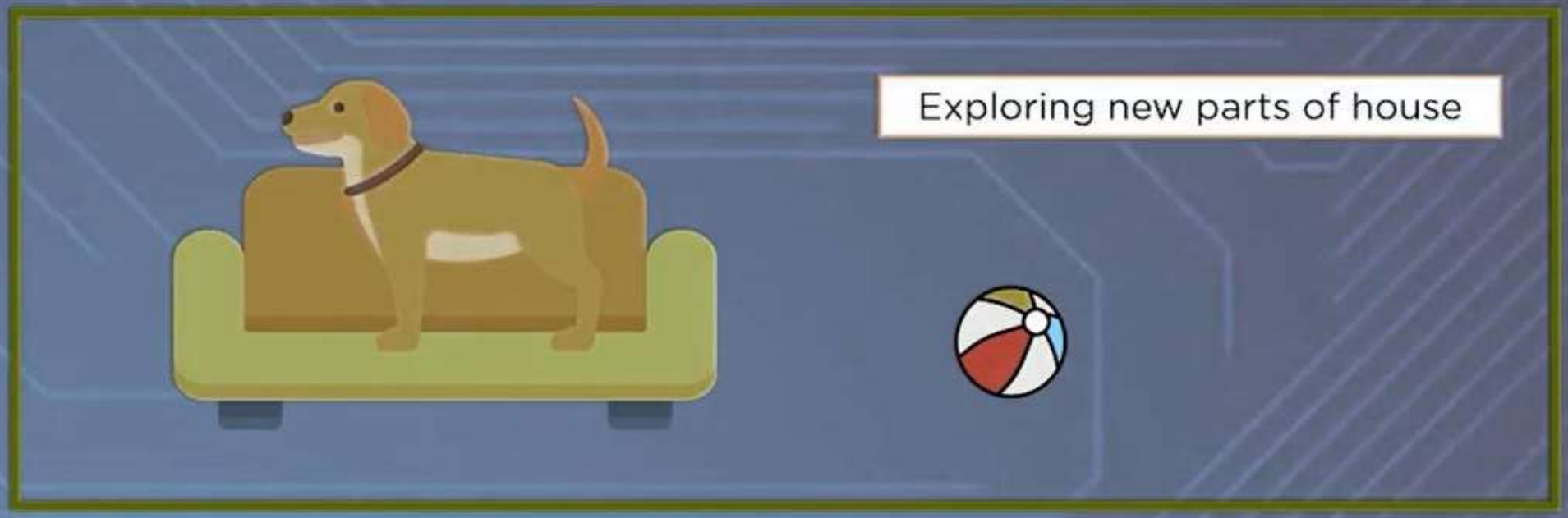
Reinforcement Learning Example

The dog will also want to run around and play and explore its environment. This quality of a model is called Exploration



Reinforcement Learning Example

The dog will also want to run around and play and explore its environment. This quality of a model is called Exploration

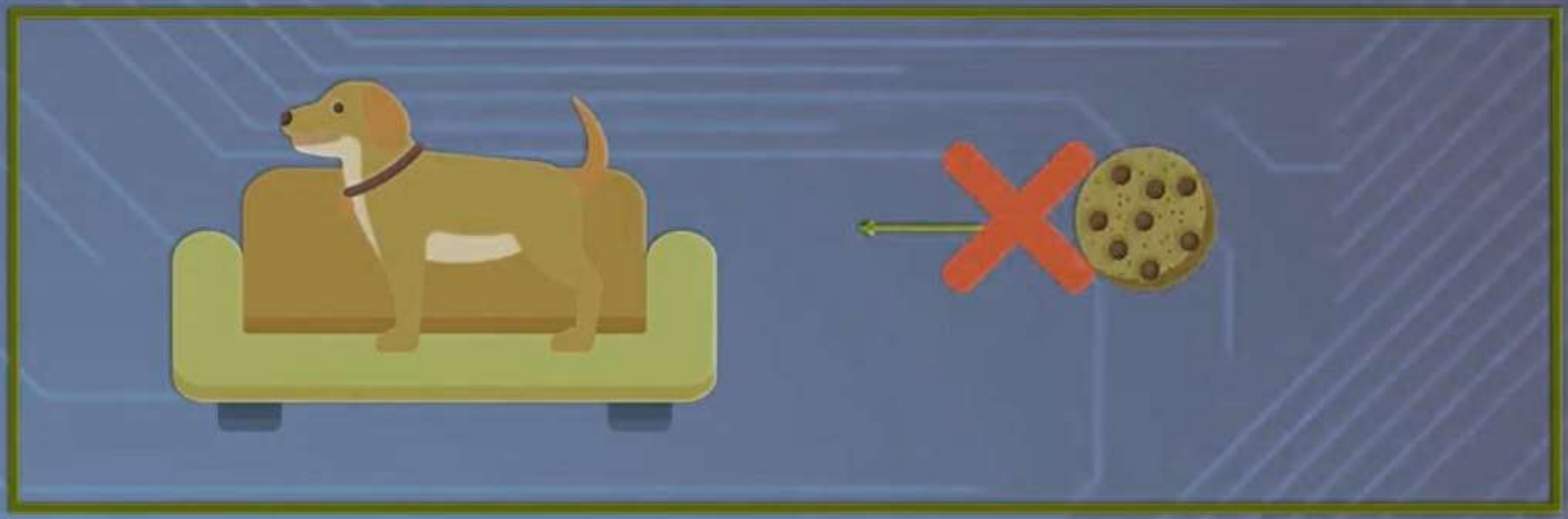


Reinforcement Learning Example

The tendency of the dog to maximize rewards is called Exploitation. There is always a tradeoff between exploration and exploitation as exploration actions may lead to lesser rewards

Reinforcement Learning Example

The tendency of the dog to maximize rewards is called Exploitation. There is always a tradeoff between exploration and exploitation as exploration actions may lead to lesser rewards



Reinforcement Learning Example

The tendency of the dog to maximize rewards is called Exploitation. There is always a tradeoff between exploration and exploitation as exploration actions may lead to lesser rewards

Action: Climbing on the sofa
Reward: None

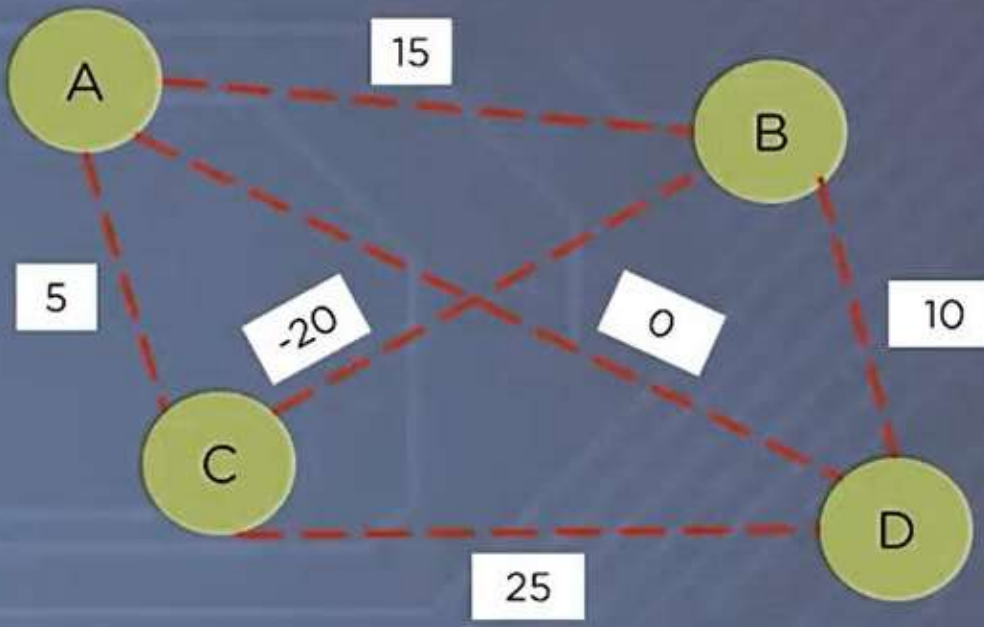
Markov's Decision Process

Markov's Decision Process is a Reinforcement Learning policy used to map a current state to an action where the agent continuously interacts with the environment to produce new solutions and receive rewards

Markov's Decision Process

In the diagram shown, we need to find the shortest path between node A and D. Each path has a reward associated with it and the path with maximum reward is what we want to choose

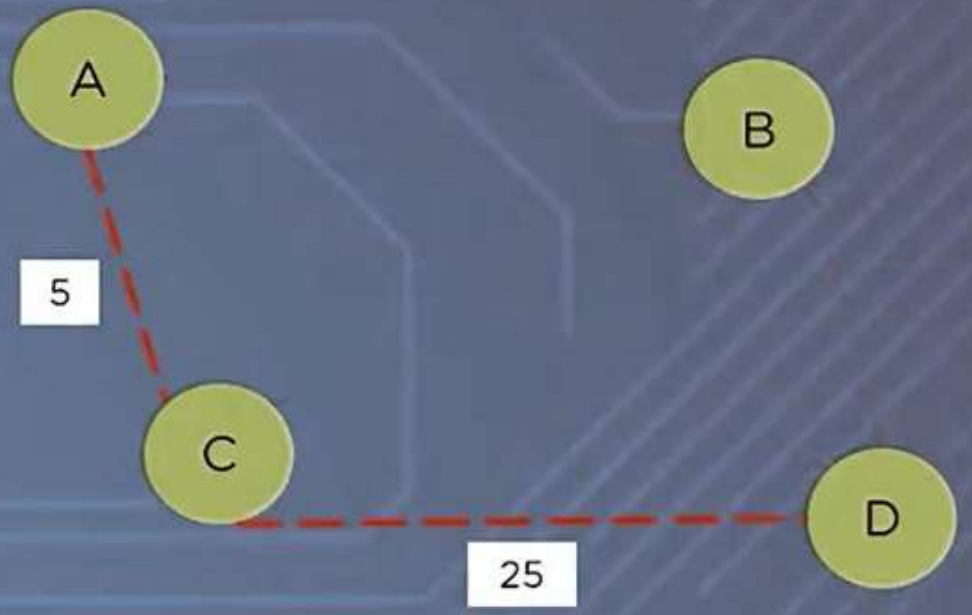
The nodes; A, B, C, D; denote the nodes.
To travel from node to node(A to B) is an action.
Reward is the cost at each path and policy is each path taken



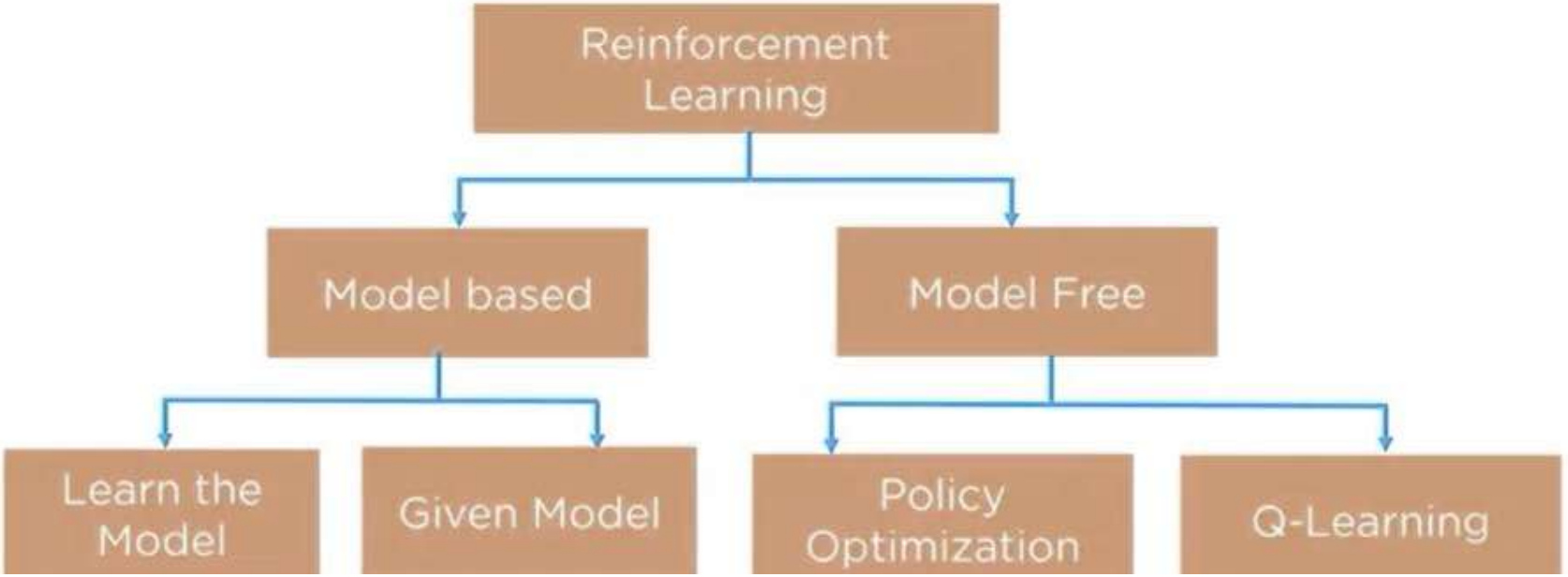
Markov's Decision Process

In the diagram shown, we need to find the shortest path between node A and D. Each path has a reward associated with it and the path with maximum reward is what we want to choose

The process will maximize the output based on the reward at each step and will traverse the path with the highest reward. This process does not explore, but maximizes reward

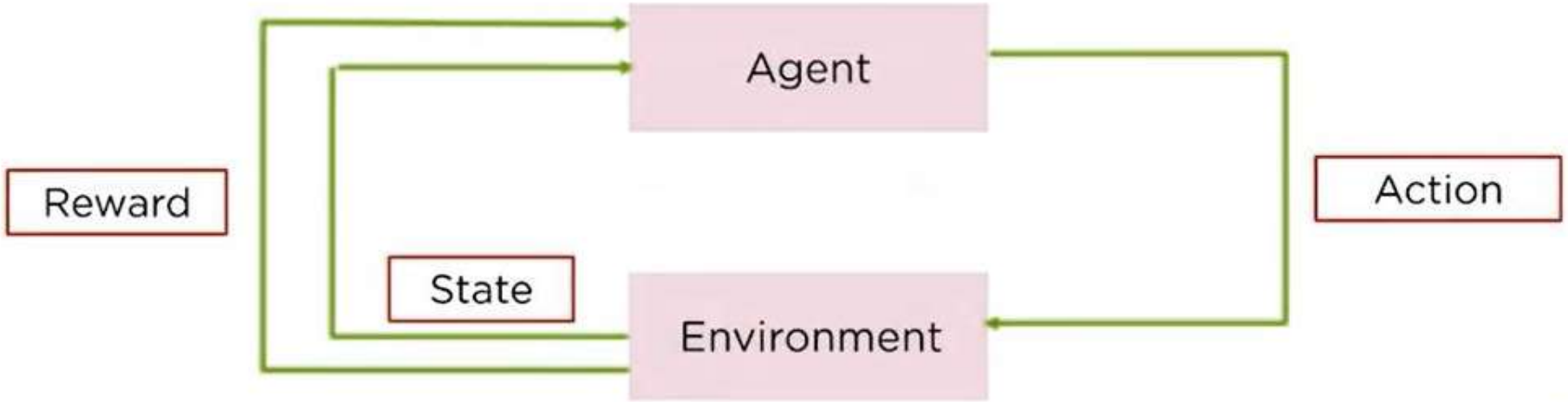


Reinforcement learning can be divided based on whether the machine uses a model to learn or learns by itself



What is Q-Learning?

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



What is Q-Learning? : Ad suggestion

Using Q-learning, we can make an Ad recommendation system which will suggest related products to our previous purchase. The reward will be if user clicks on the suggested product

Some Important Terms

States

The State, S, represents the current position of an agent in an environment

Action

The Action, A, is the step taken by the agent when it is in a particular state

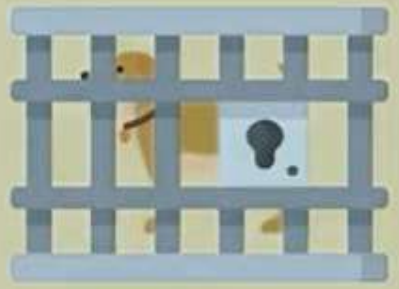
Rewards

For every action agent will get positive or nega reward

Some Important Terms

Episodes

When an agent ends up in a terminating state and can't take a new action



Q-Values

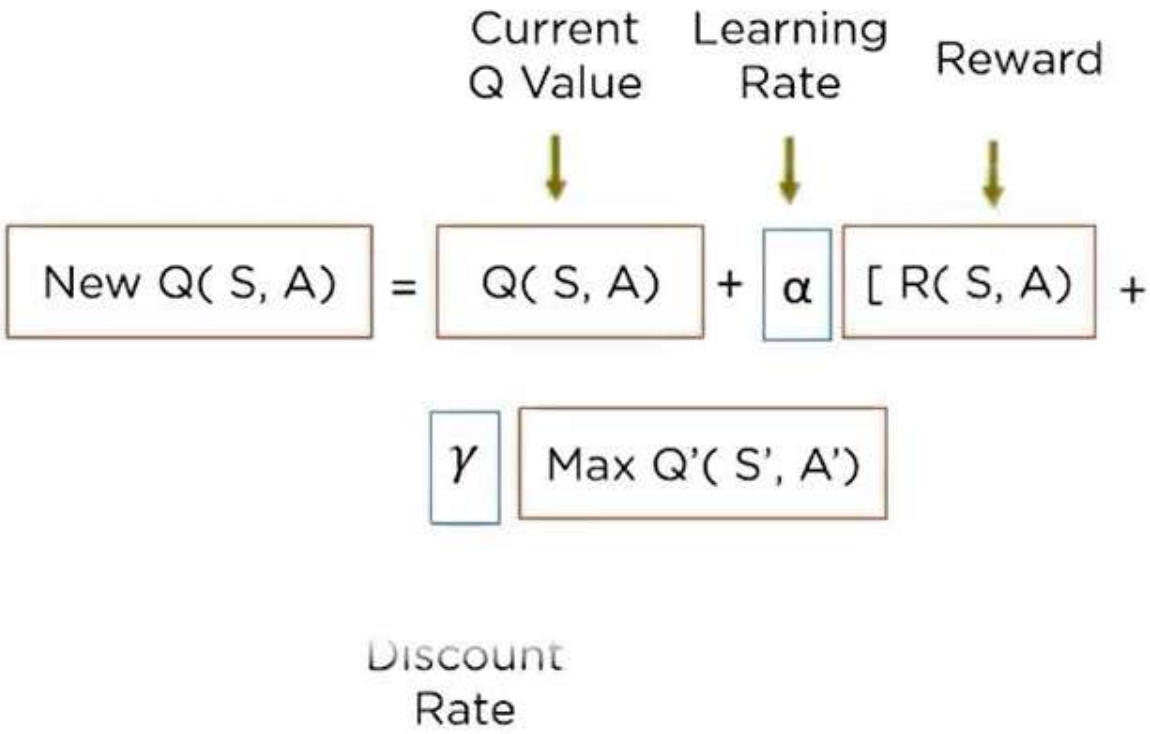
Used to determine how good an Action, A, taken at a particular state, S, is **Q(A,S)**

Temporal Differe

A formula used to
the Q-Value by u
the value of curr
state and action
previous state a
action

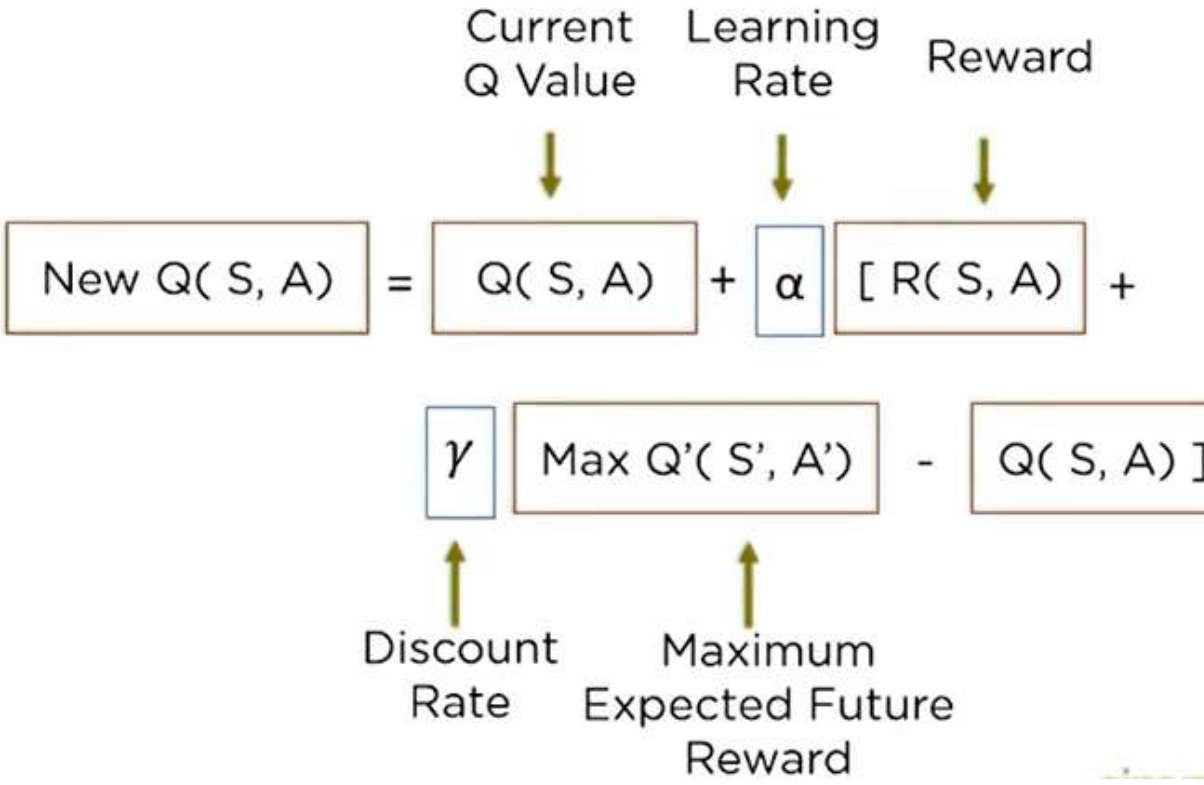
Bellman's Equation

The Bellman Equation is used to determine the value of a particular state and deduce how good it is to be in/ take that state.
The optimal state will give us the highest optimal value



Bellman's Equation

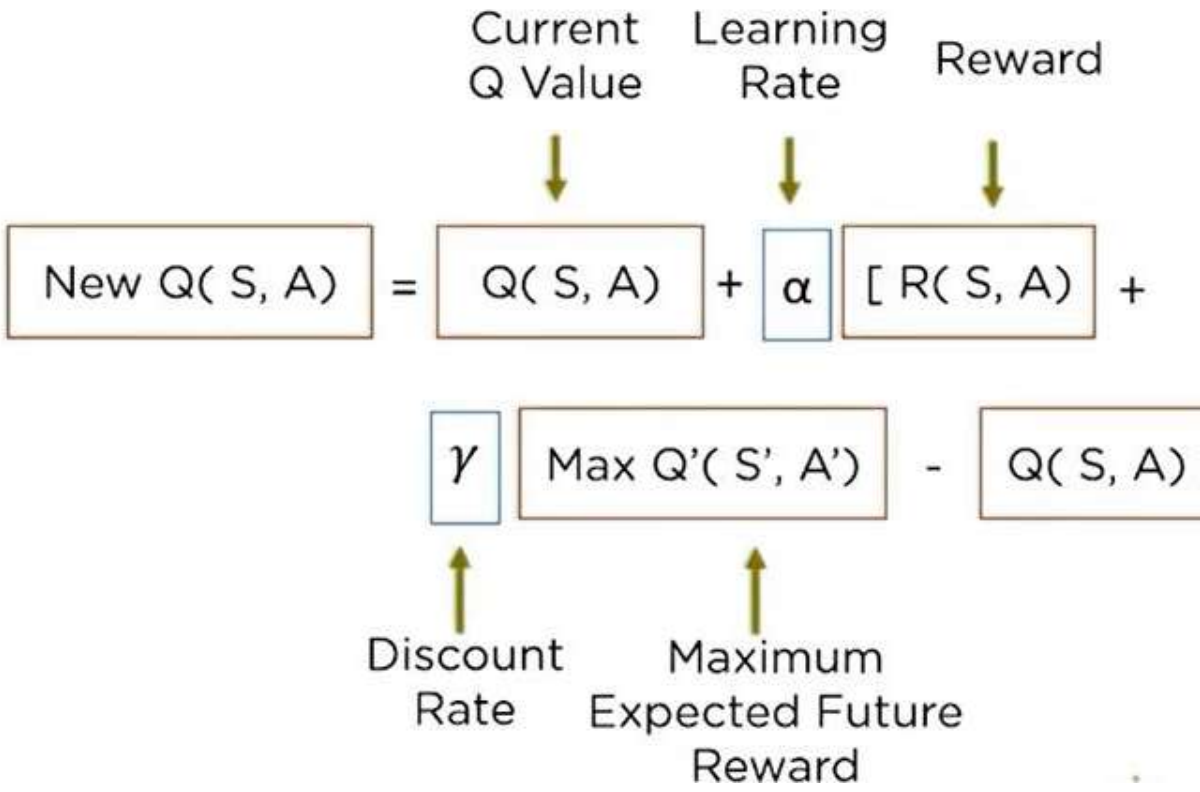
The Bellman Equation is used to determine the value of a particular state and deduce how good it is to be in/ take that state.
The optimal state will give us the highest optimal value



Bellman's Equation

Factors Influencing Q-values :

- Current State and Action (S , A)
- Previous State and Action (S', A')
- Reward for Action, R
- Maximum expected future reward



Steps in Q-Learning

Step 1

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

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

Step 2

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

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

Step 3

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

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

Step 4

Continue the same until the table is filled or an episode ends

Action	Fetching	
Start	5	
Idle	2	
Wrong Action	2	
Correct Action	54	
End	3	

The below table gives us an idea of how many times an action has been taken and how positively(correct action) or negatively(wrong action) it is going to affect the next state

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

WHAT IS REINFORCEMENT LEARNING?

Reinforcement Learning is a sub-branch of ML that trains a mod
return an optimum solution for a problem by taking a sequenc
decisions by itself. Consider a robot learning to go from one plac
another.

REINFORCEMENT LEARNING

For example,

Step -1 : imagine that you want to solve the simple supervised learning problem image classification with two target classes—dog and cat.

Step – 2: Gather the training dataset and implement the classifier using your favor learning (DL) toolkit. After a while, the model that has converged demonstrates e performance. Great! You deploy it and leave it running for a while. However, vacation at some seaside resort, you return to discover that dog grooming fashio changed and a significant portion of your queries are now misclassified, so you update your training images and repeat the process again.

Not so great!

REINFORCEMENT LEARNING

- Supervised (inductive) learning is the simplest and most studied of learning
- How can an agent learn behaviors when it doesn't have a teacher tell it how to perform?
  - The agent has a task to perform
  - It takes some actions in the world
  - At some later point, it gets feedback telling it how well it did on performing task
  - The agent performs the same task over and over again
- This problem is called **reinforcement learning:**
  - The agent gets *positive reinforcement* for tasks done well
  - The agent gets *negative reinforcement* for tasks done poorly

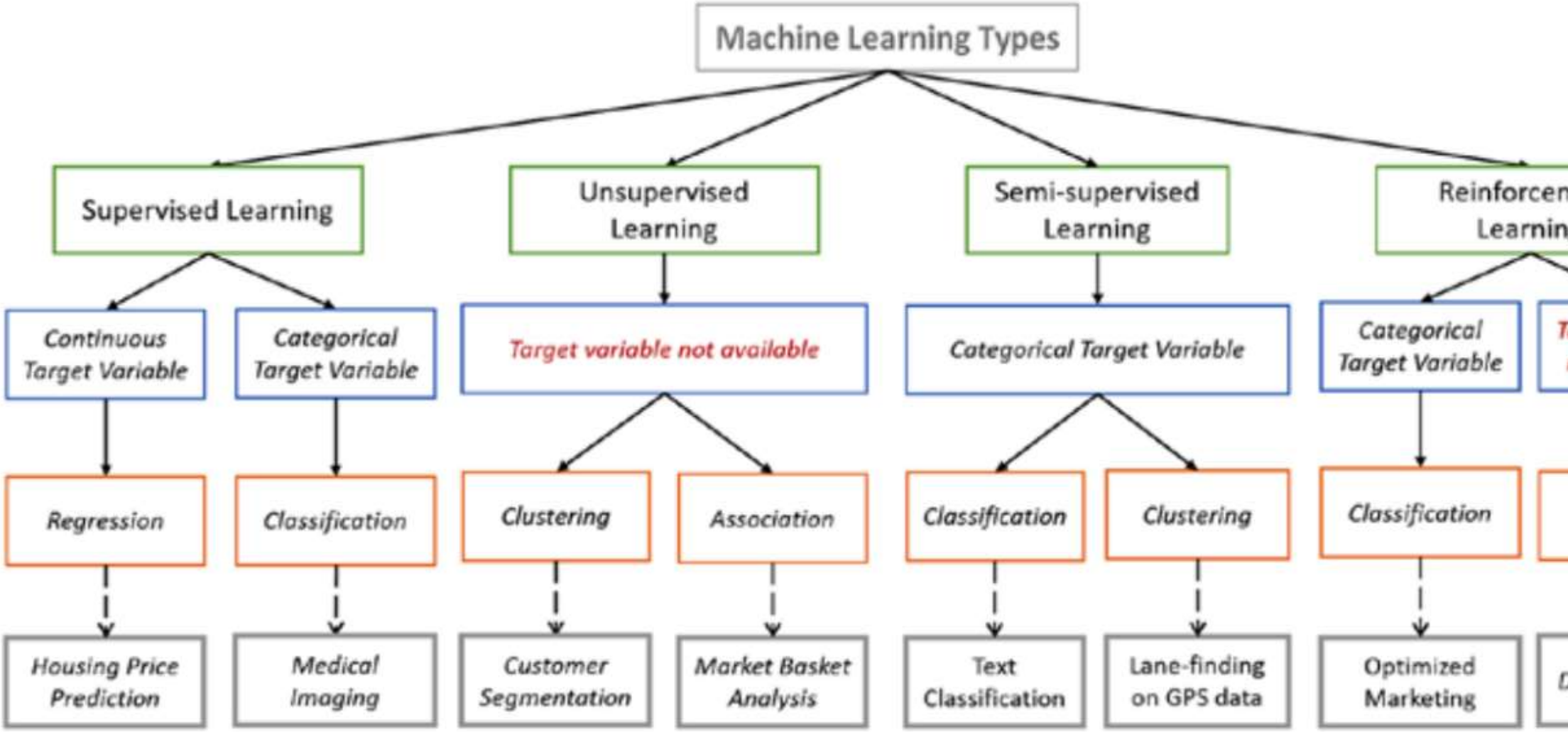
REINFORCEMENT LEARNING

- The goal is to get the agent to act in the world so as t maximize its rewards
- The agent has to figure out what it did that made it ge the reward/punishment
  - This is known as the **credit assignment** problem

Reinforcement learning approaches can be used to train computers to do many tasks

- backgammon and chess playing
- job shop scheduling
- controlling robot limbs

MACHINE LEARNING TYPES & APPLICATIO



RL APPLICATIONS – 1. AUTONOMOUS VEHIC

What is DeepRacer AWS?

Image result for AWS Deepracer

AWS DeepRacer is an autonomous 1/18th scale race car designed to test RL models by ra
a physical track. Using cameras to view the track and a reinforcement model to control
and steering, the car shows how a model trained in a simulated environment can be tra
to the real-world.

RL APPLICATIONS – 2. AD OPTIMISATIO

consider a very simple scenario:

You're the owner of a news website. To pay for the costs of hosting and staff, yo entered a contract with a company to run their ads on your website. The compa provided you with five different ads and will pay you one dollar every time a clicks on one of the ads.

Solution: Multi-Armed Bandit Problem, a RL Formulation
Policy : Epsilon greedy algorithm

RL APPLICATIONS – 3. CONTEXTUAL BAND

Contextual advertisements with personalization and recommendations

Solution: Multi-Armed Bandit Problem, a RL Formulation
Policy : Epsilon greedy algorithm