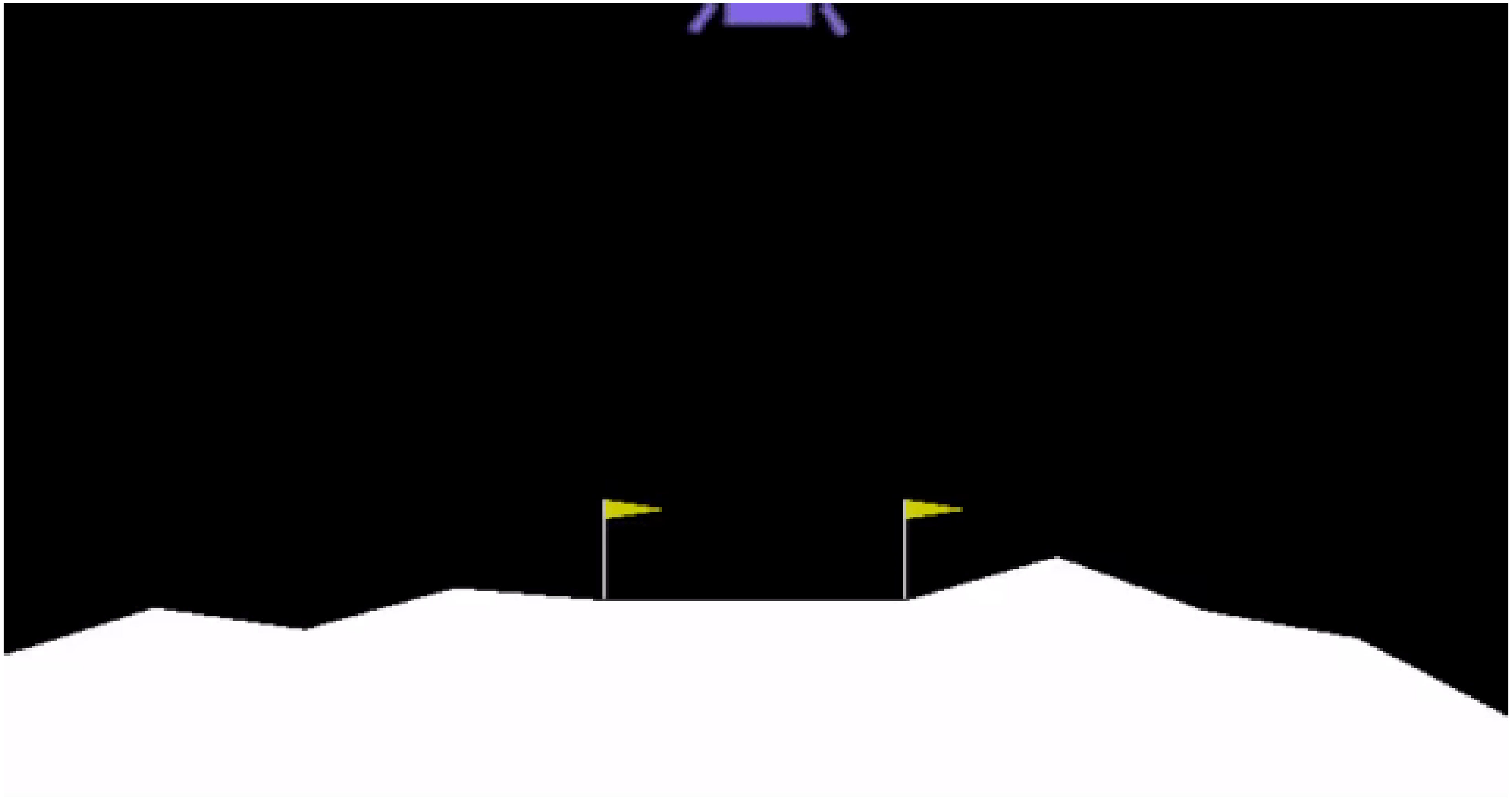
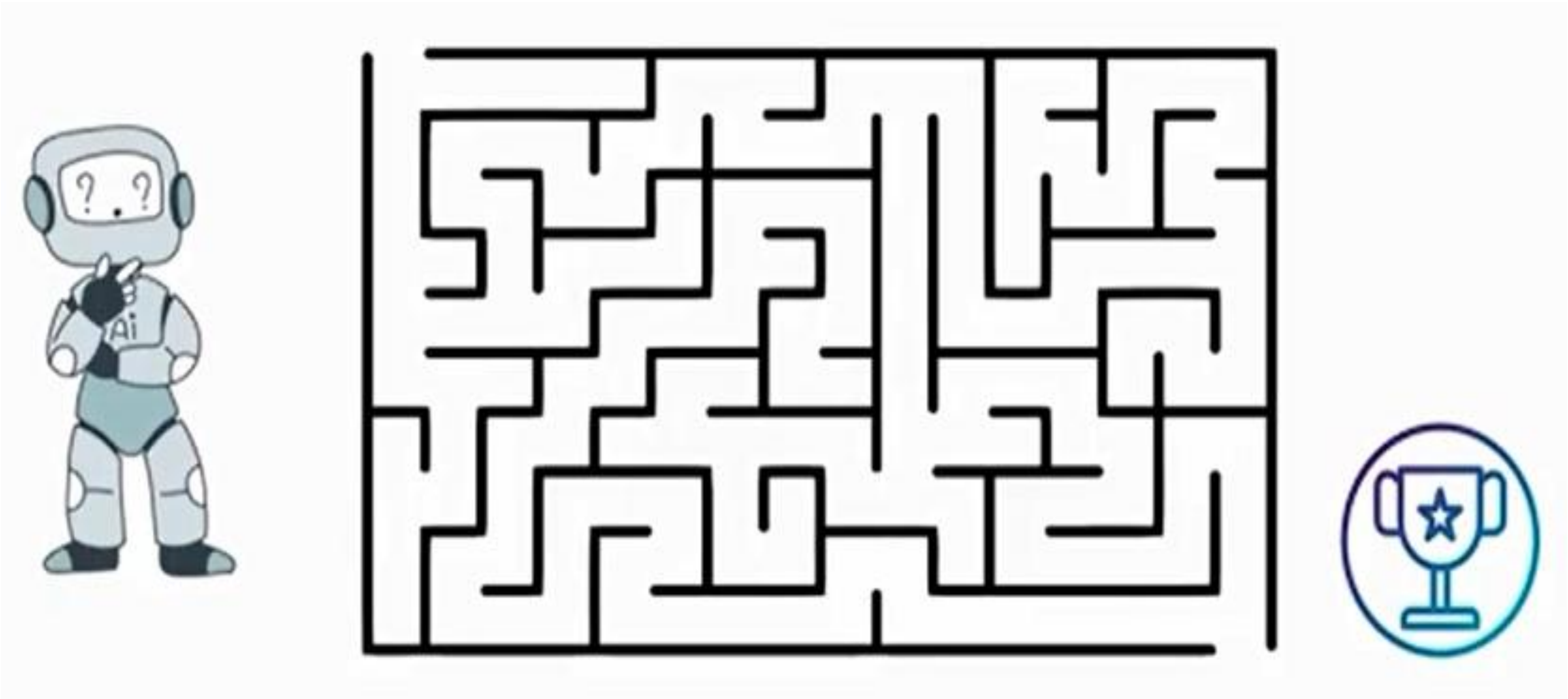
Reinforcement Learning

By
Prof(Dr) Premanand P Ghadekar



Reinforcement Learning

Reinforcement Learning is a type of Machine learning where an Agent learns to behave in a environment by performing actions and seeing the results.



Reinforcement Learning includes training of the algorithms using a system of **reward and punishment.**

Reinforcement learning

Reinforcement learning focuses on teaching agents through trial and error

Reinforcement learning

Related to learning which is best action to perform situation by situation in order **to maximize aggregate reward**

RL agent has to learn policy without domain expert.

Agent has to choose between

- Exploiting its current knowledge of the environment (perform an action already tried previously in that situation) or
- Exploring actions never tried before in that situation.

Reinforcement Learning Process

Reinforcement Learning system is comprised of two main components:

- Agent
- Environment

Reinforcement Learning Process

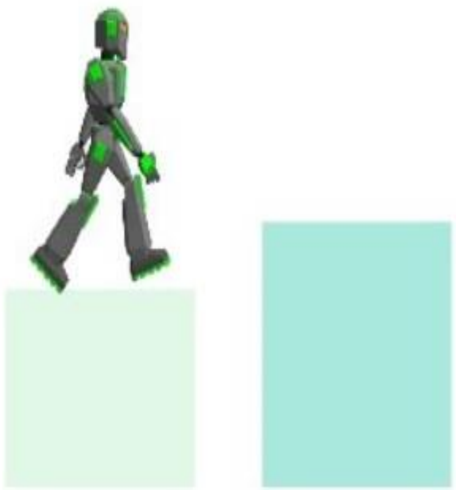
Reinforcement Learning system is comprised of two main components:

- Agent
- Environment

Reinforcement Learning Process

Reinforcement Learning system is comprised of two main components:

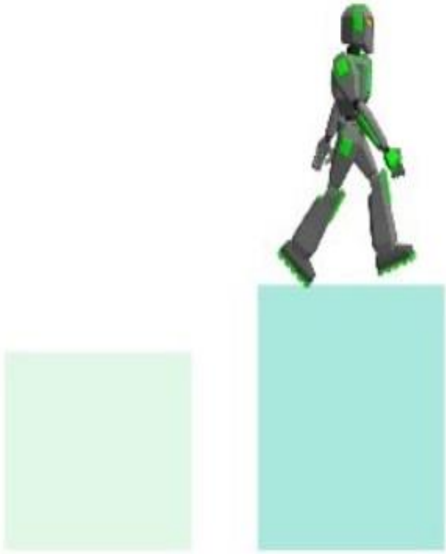
- Agent
- Environment



Reinforcement Learning Process

Reinforcement Learning system is comprised of two main components:

- Agent
- Environment



Reinforcement Learning Process

Reinforcement Learning system is comprised of two main components:

- Agent
- Environment

Reinforcement Learning Process

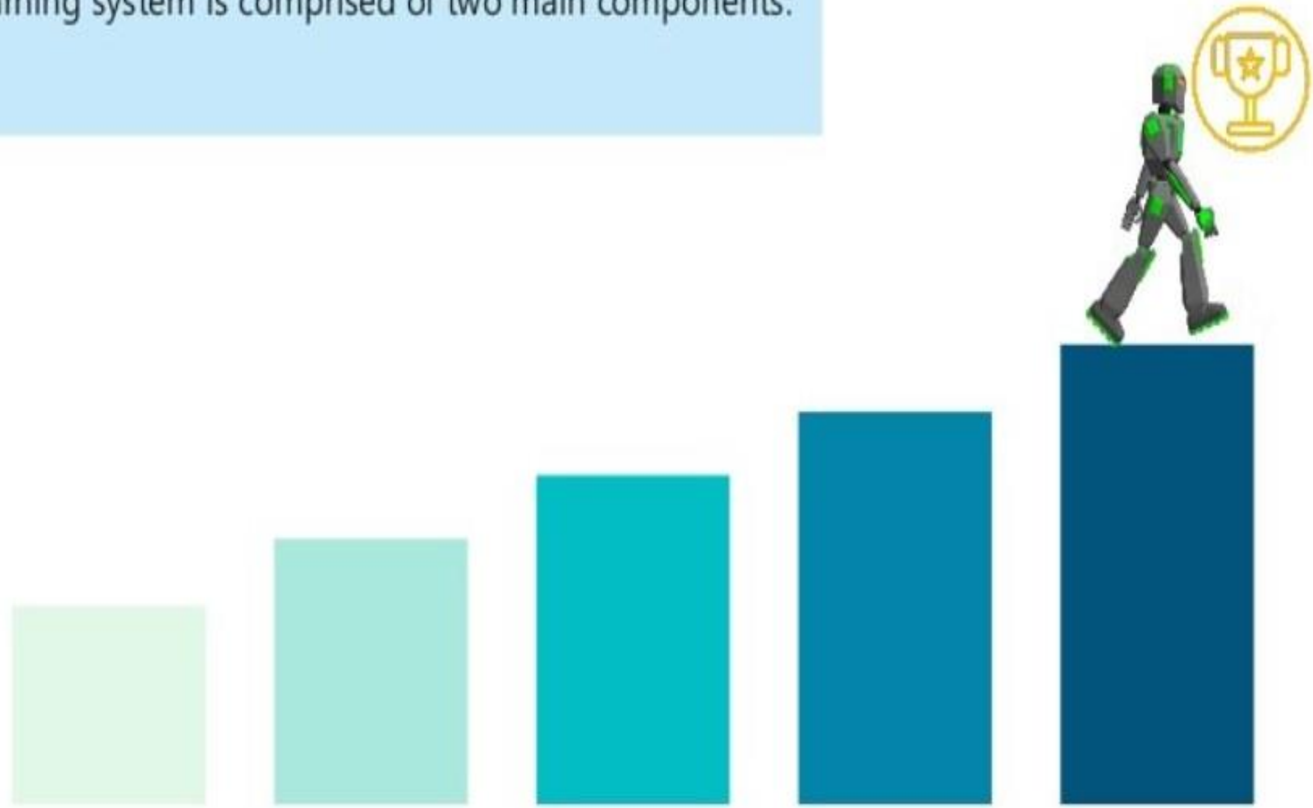
Reinforcement Learning system is comprised of two main components:

- Agent
- Environment

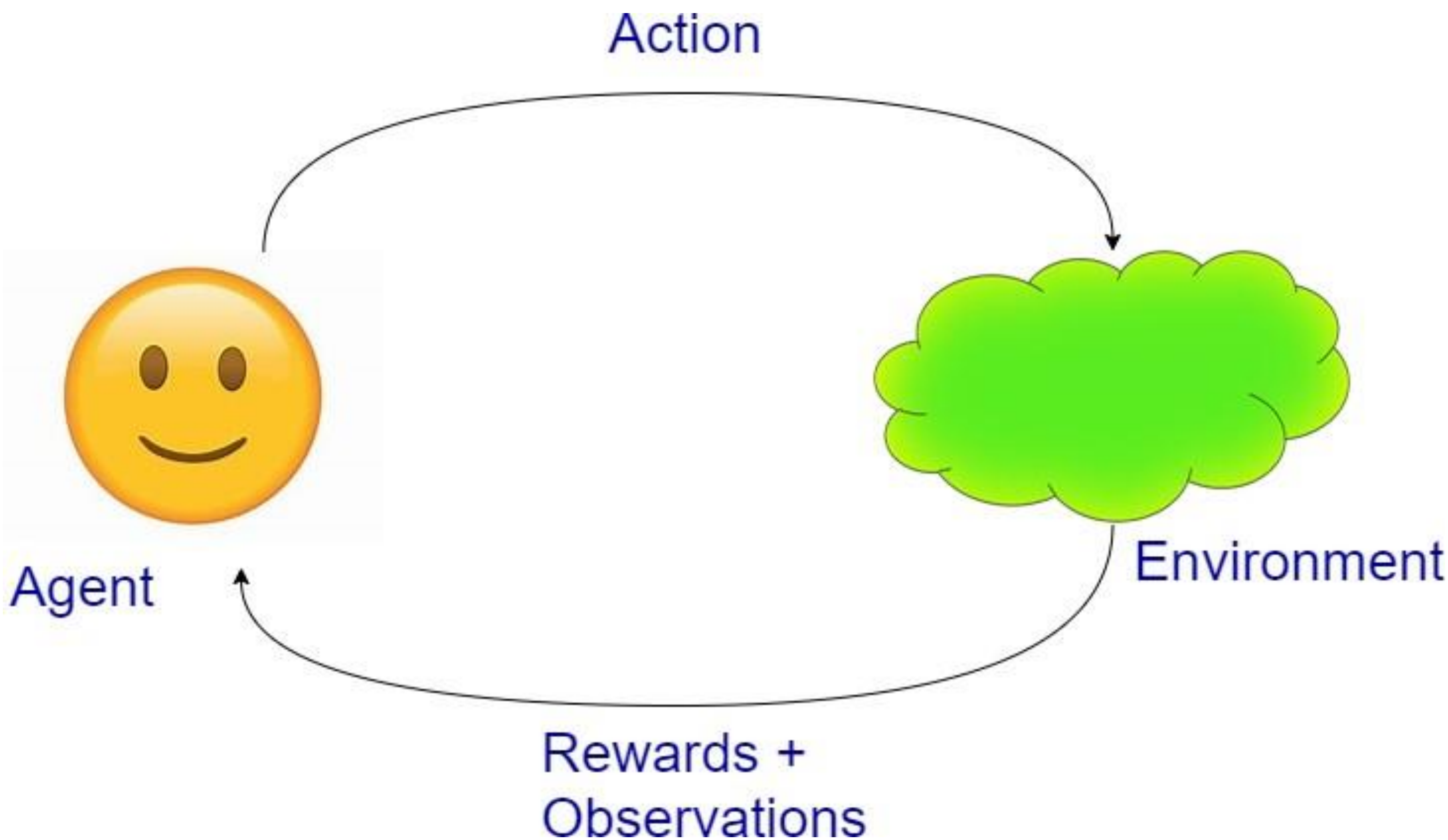
Reinforcement Learning Process

Reinforcement Learning system is comprised of two main components:

- Agent
- Environment

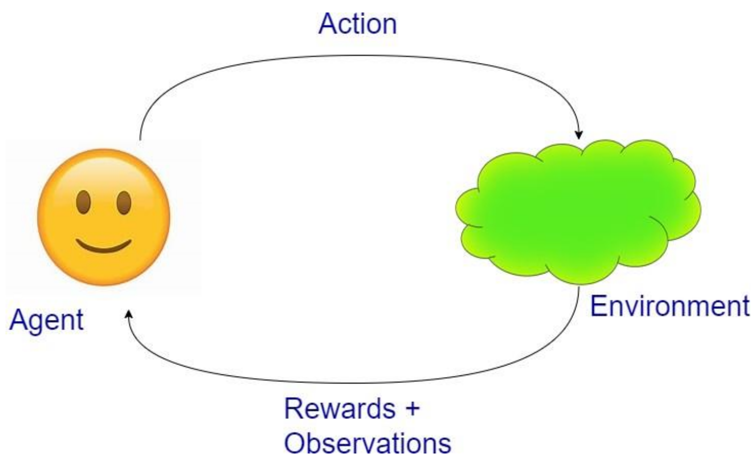


Reinforcement learning Set up



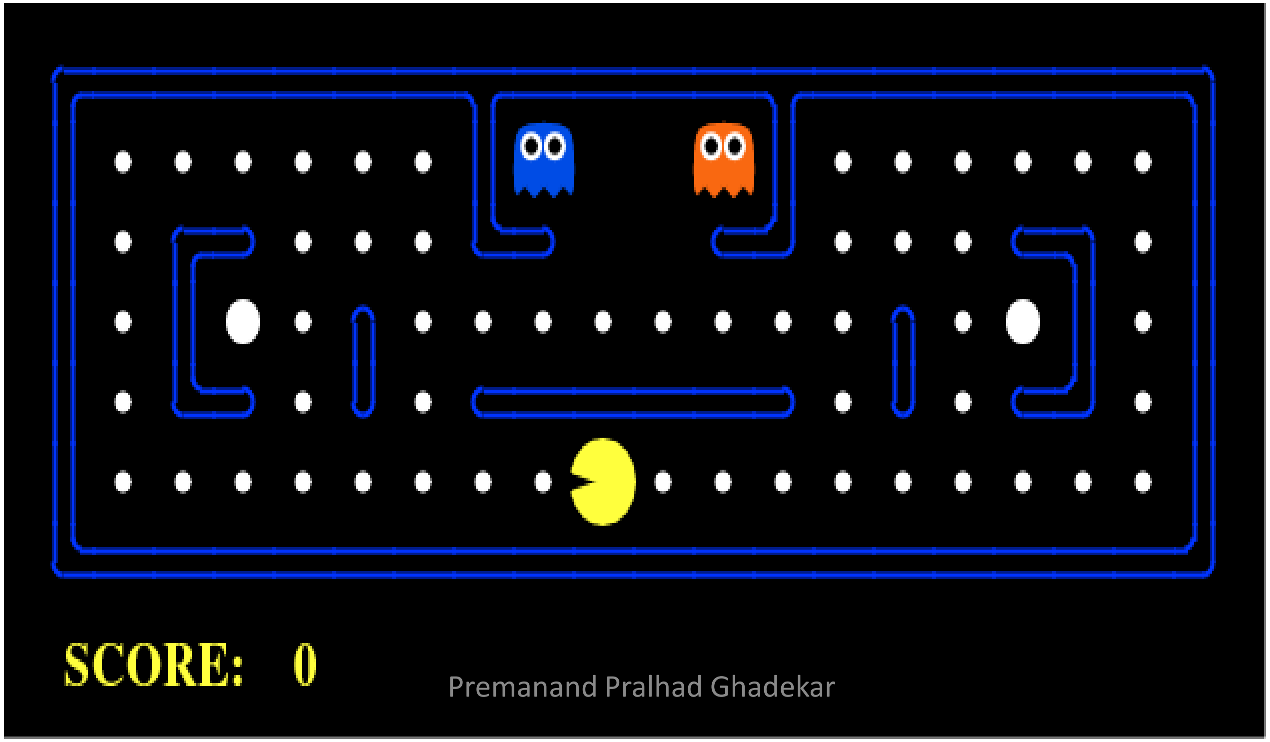
Concepts

- **Agent :**
  - The actor operating within the environment
  - It is usually governed by a policy (a rule that decides what actions to take)
- **Environment:**
  - The world in which the agent can operate in
- **Action:**
  - The agent can do something within the environment known as action
- **Rewards and Observations**
  - In return to the action agent receives reward and a view of what the environment looks like after acting on it



Reinforcement Learning Process

- Let's take the game of **PacMan** where the goal of the **agent(PacMan)** is to eat the food in the grid while avoiding the ghosts on its way.
- In this case, **the grid world is the interactive environment for the agent** where it acts. Agent receives a reward for eating food and punishment if it gets killed by the ghost (loses the game).
- The **states are the location of the agent in the grid world** and **the total cumulative reward is the agent winning the game.**

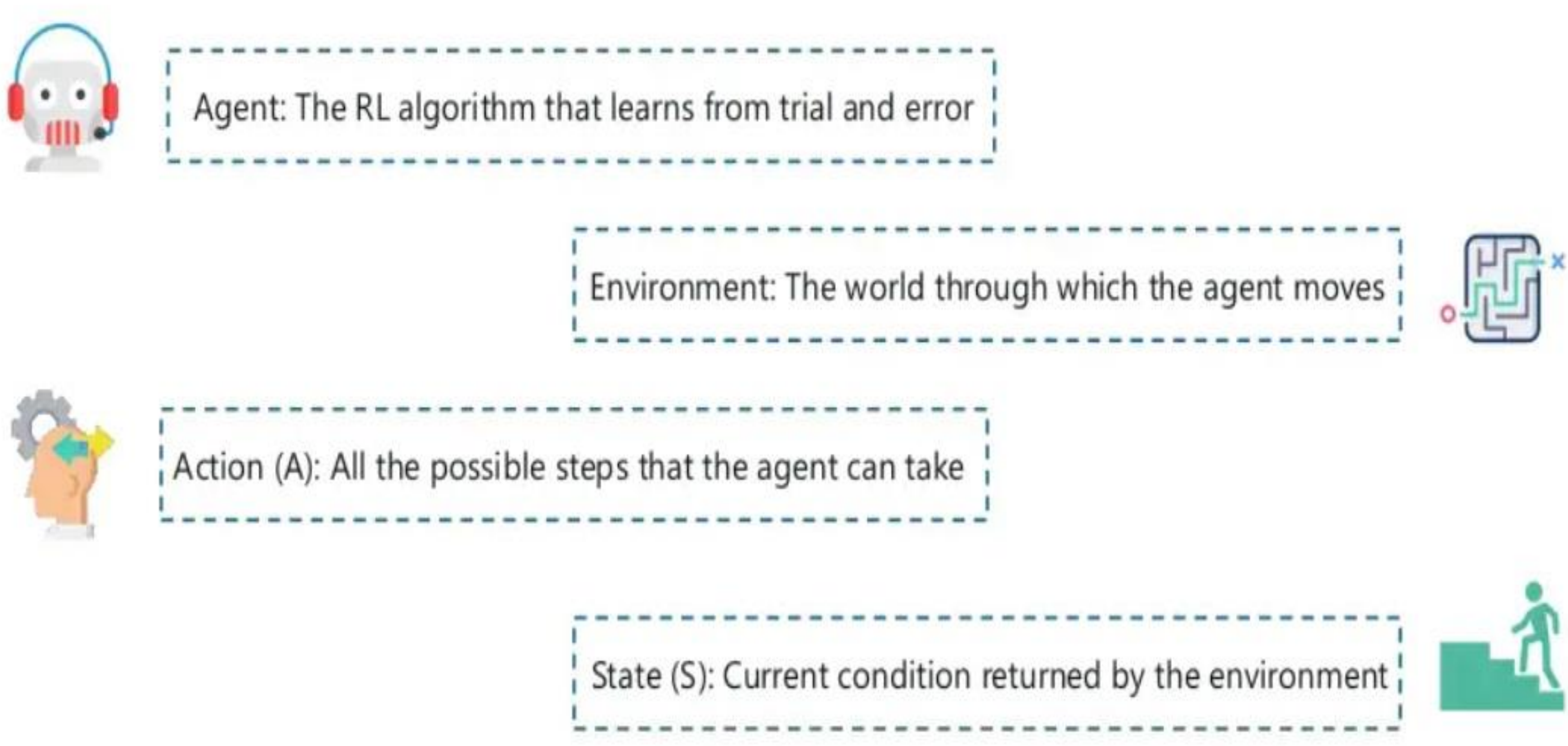


Reinforcement Learning

Counter Strike Example

1. The RL Agent (Player1) collects state S^0 from the environment
2. Based on the state S^0, the RL agent takes an action A^0, initially the action is random
3. The environment is now in a new state S^1
4. RL agent now gets a reward R^1 from the environment
5. The RL loop goes on until the RL agent is dead or reaches the destination

Reinforcement Learning Definition



Reinforcement Learning Definition

Reward (R): An instant return from the environment to appraise the last action

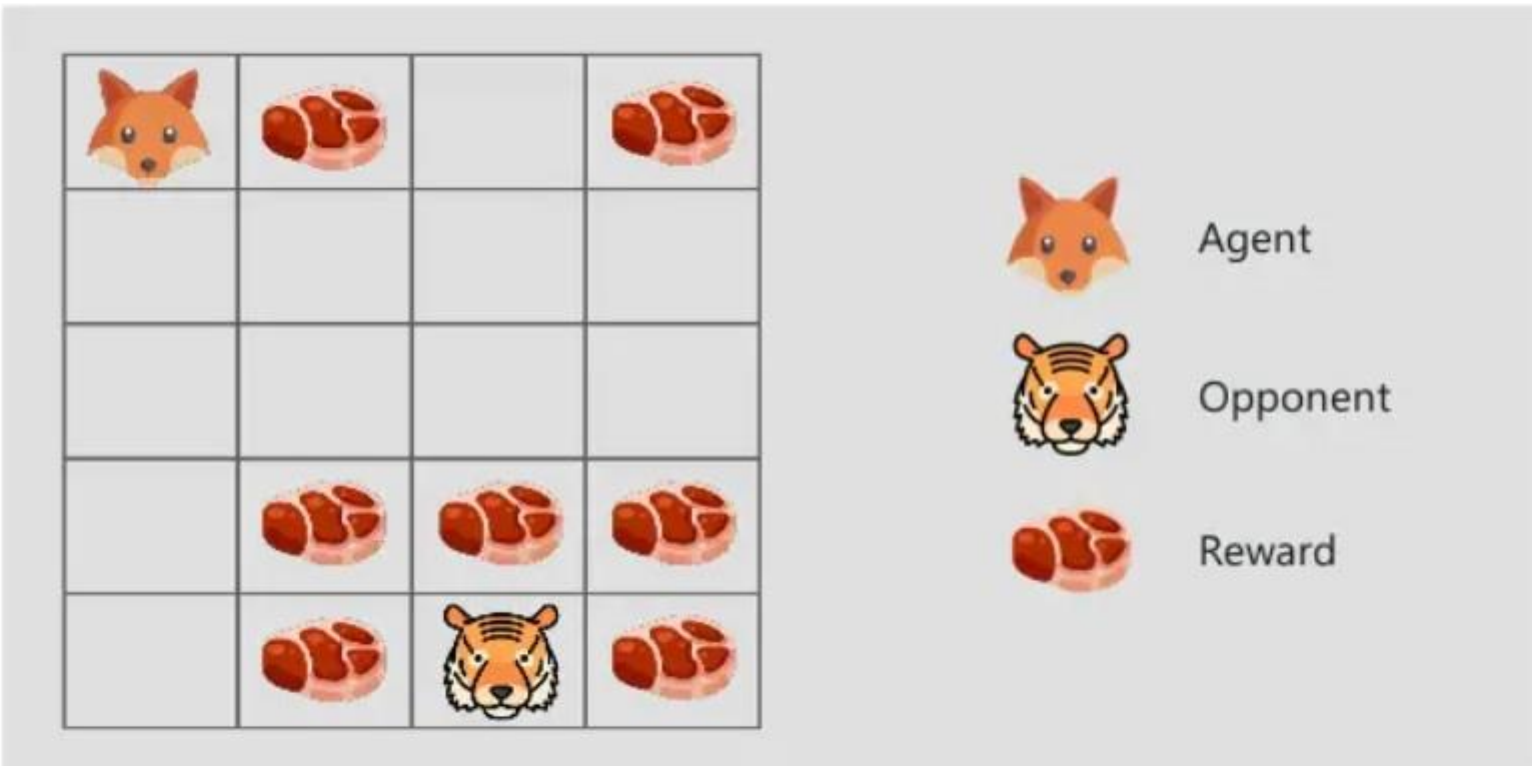
Policy (π): The approach that the agent uses to determine the next action based on the current state

Value (V): The expected long-term return with discount, as opposed to the short-term reward R

Action-value (Q): This similar to Value, except, it takes an extra parameter, the current action (A)

Reinforcement Learning Concept- Reward Maximization

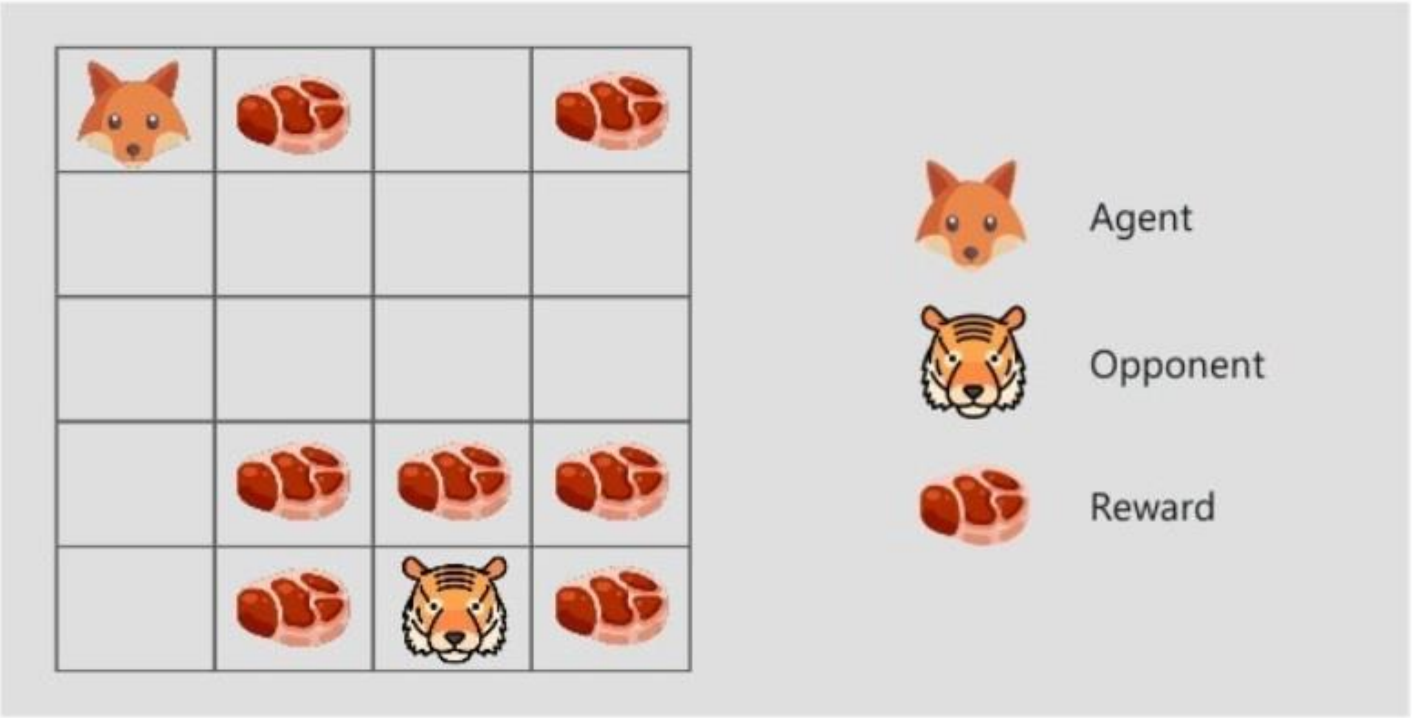
Reward maximization theory states that, *a RL agent must be trained in such a way that, he takes the best action so that the reward is maximum.*



Exploitation and Exploration

Exploitation is about using the already known exploited information to heighten the rewards

Exploration is about exploring and capturing more information about an environment

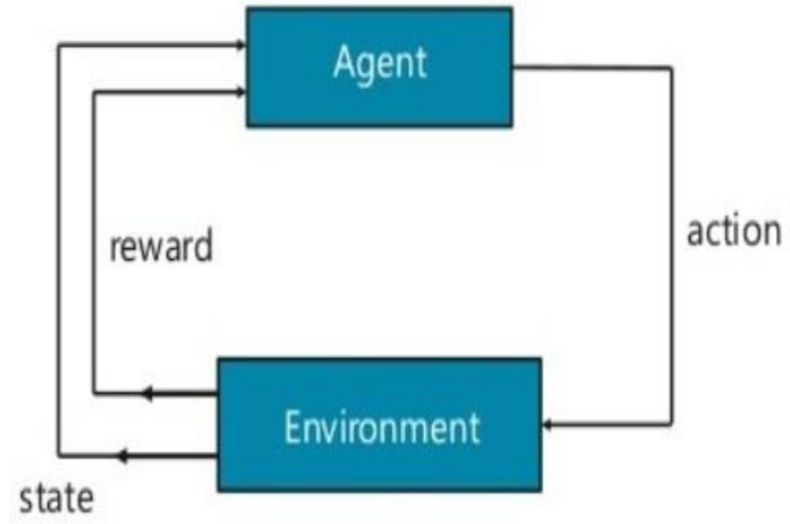


RL Representation-Markov Decision Process

The mathematical approach for mapping a solution in reinforcement learning is called *Markov Decision Process* (MDP)

The following parameters are used to attain a solution:

- Set of actions, A
- Set of states, S
- Reward, R
- Policy, π
- Value, V

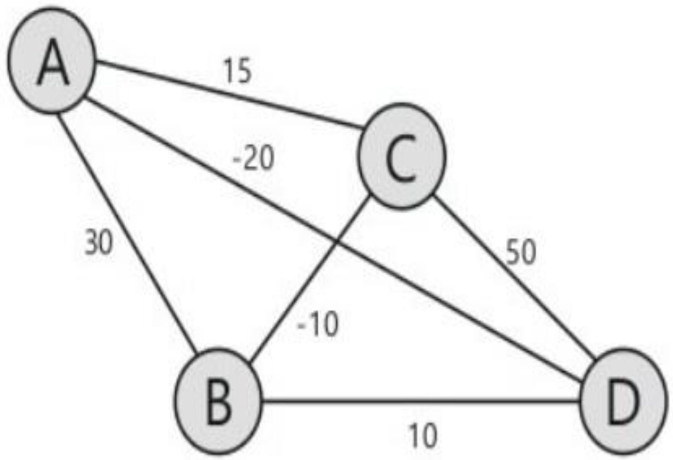


Markov Decision Process-Shortest Path Problem

Goal: Find the shortest path between A and D with minimum possible cost

In this problem,

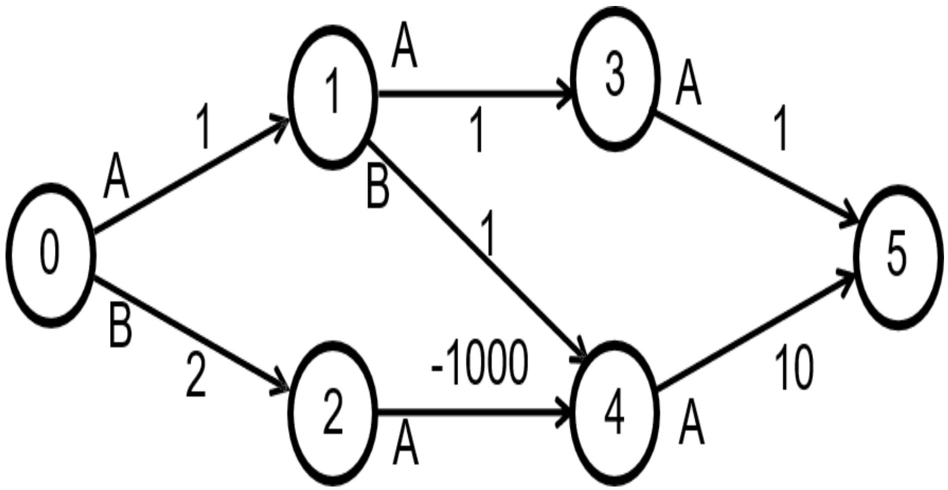
- Set of states are denoted by nodes i.e. {A, B, C, D}
- Action is to traverse from one node to another {A -> B, C -> D}
- Reward is the cost represented by each edge
- Policy is the path taken to reach the destination {A -> C -> D}



Markov Decision Process (MDP)

The mathematical appaoarach for mapping a solution in RL is called MDP

Markov Assumption: s_{t+1} and r_{t+1} depend only on st and at but not on anything that happened before time t



RL represenation as MDP

Parameters used

- states : $S = \{s_1, s_2, ...s_{|S|}\}$
- Actions: $A = \{a_1, ..., a_{|A|}\}$
- Transition probabilities: $T(s, a, s_j) = Pr(s_j|s, a)$
- Rewards:$R : SXAXS- > R$
- Policy: $\pi : S- > A$, π is the set of all policies

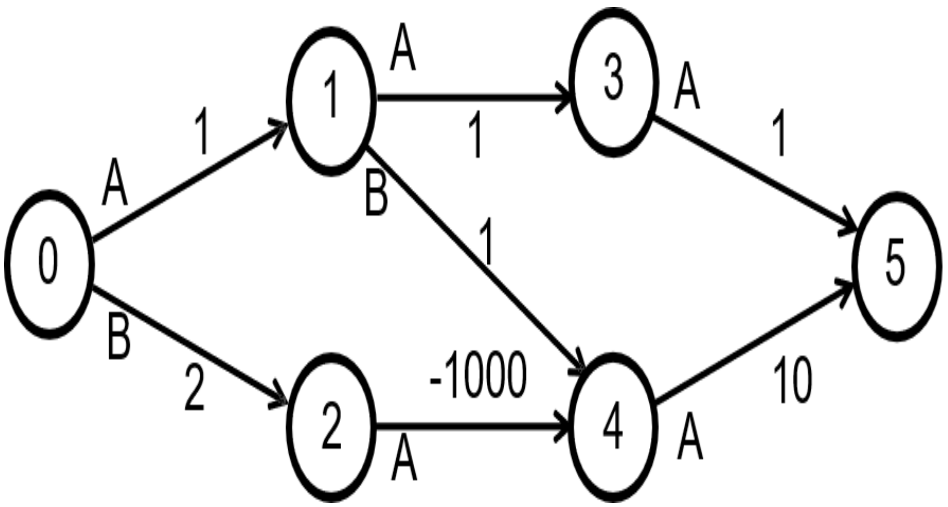
  Value function:
- $V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ...|s0 = s, \pi]$

  where γ is a discount factor

MDP

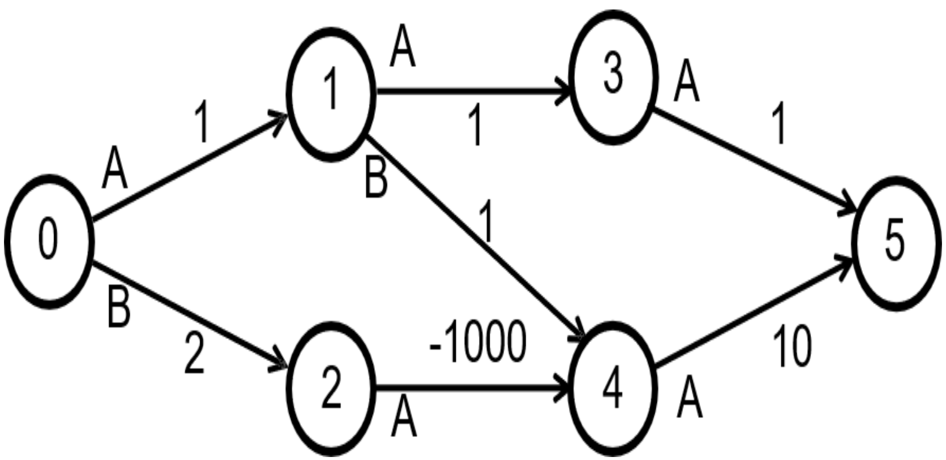
Policy: Complete mapping of state action, from initial to final state

How many possibilities?



Three policies are possible

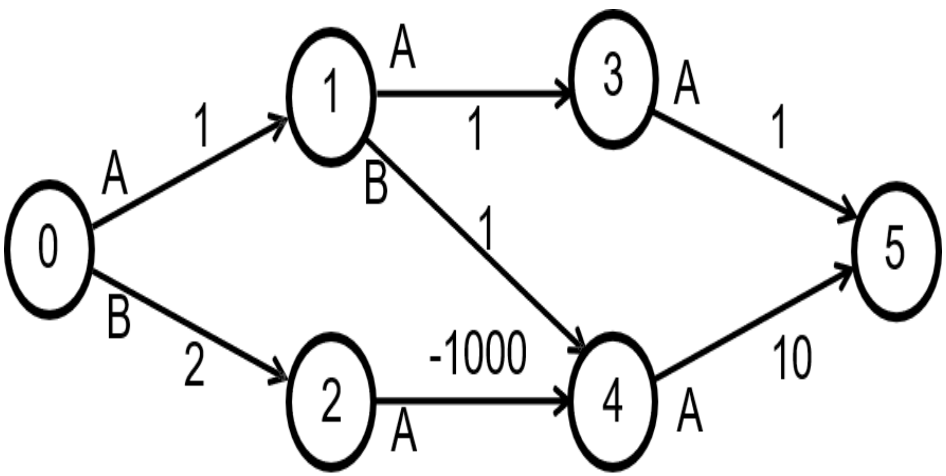
- $0->1->3->5$
- $0->1->4->5$
- $0->2->4->5$



MDP

Value: Expected long term reward returned with discount factor
Values of three policies

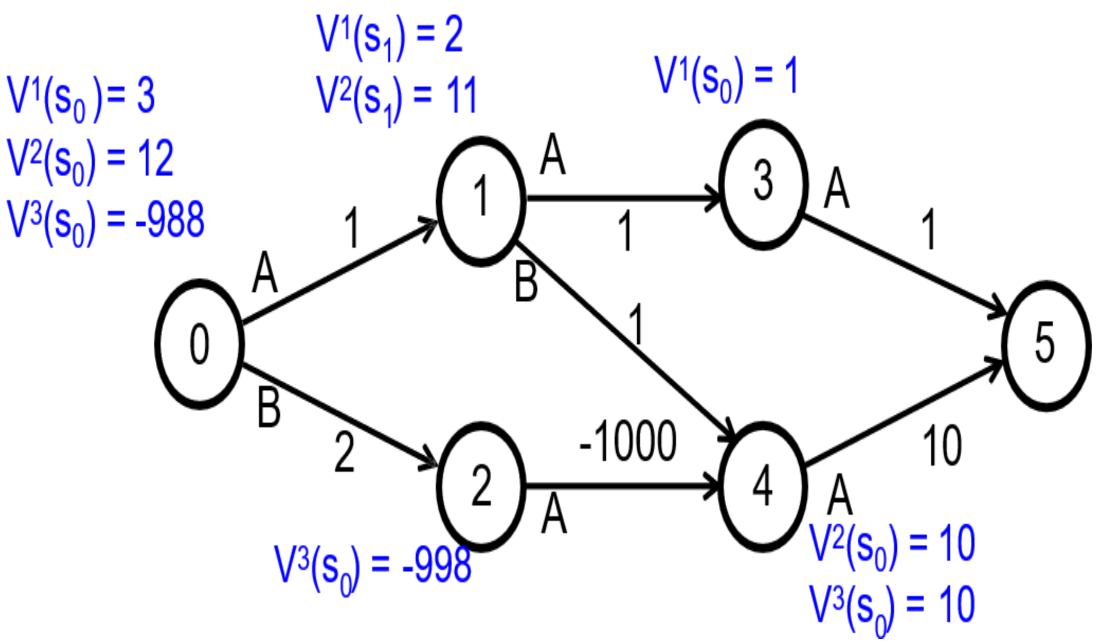
- $0 -> 1 -> 3 -> 5 = 1 + 1 + 1 = 3$
- $0 -> 1 -> 4 -> 5 = 1 + 1 + 10 = 12$
- $0 -> 2 -> 4 -> 5 = 2 - 1000 + 10 = -988$



Value Functions with Policies

Associating a value with each state For a fixed policy

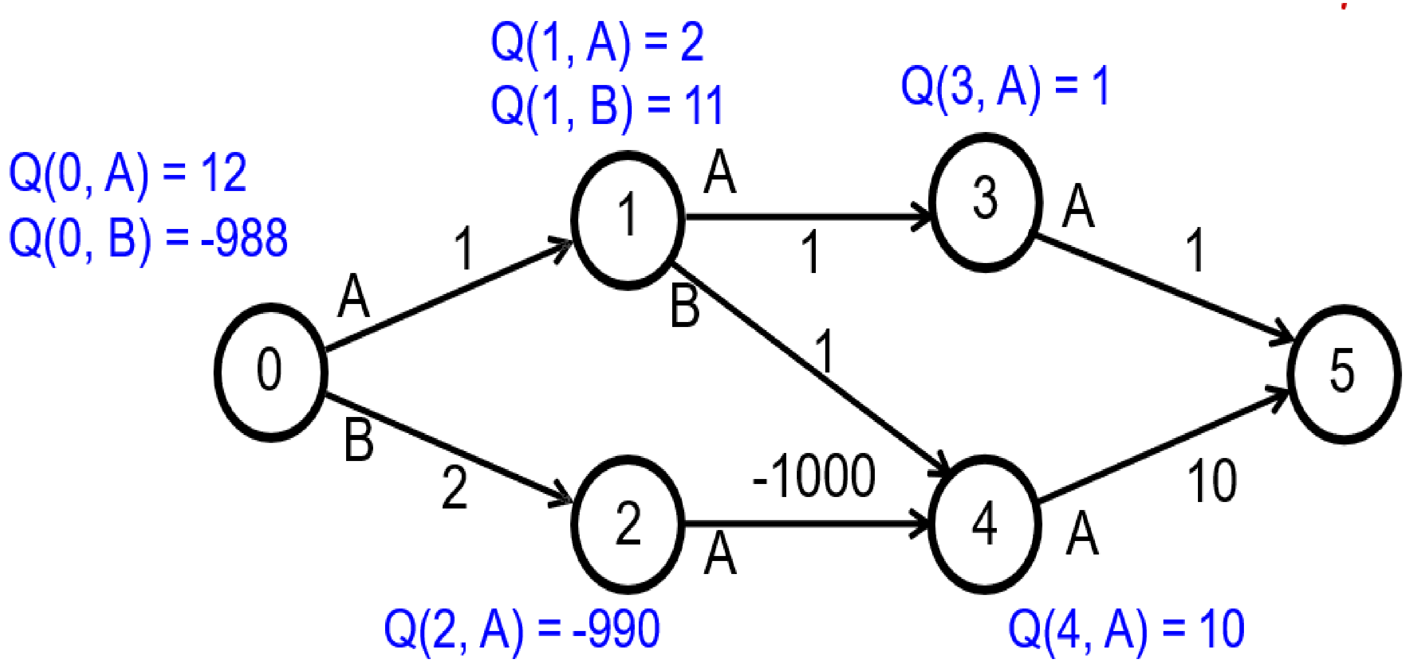
- How good is it to run policy *π*
- State value function, V



Value Functions with State Action : Q Value

Value without specifying the policy

- Specify the value of taking an action *a* from state *s* and then performing optimally
- State-action value function, Q



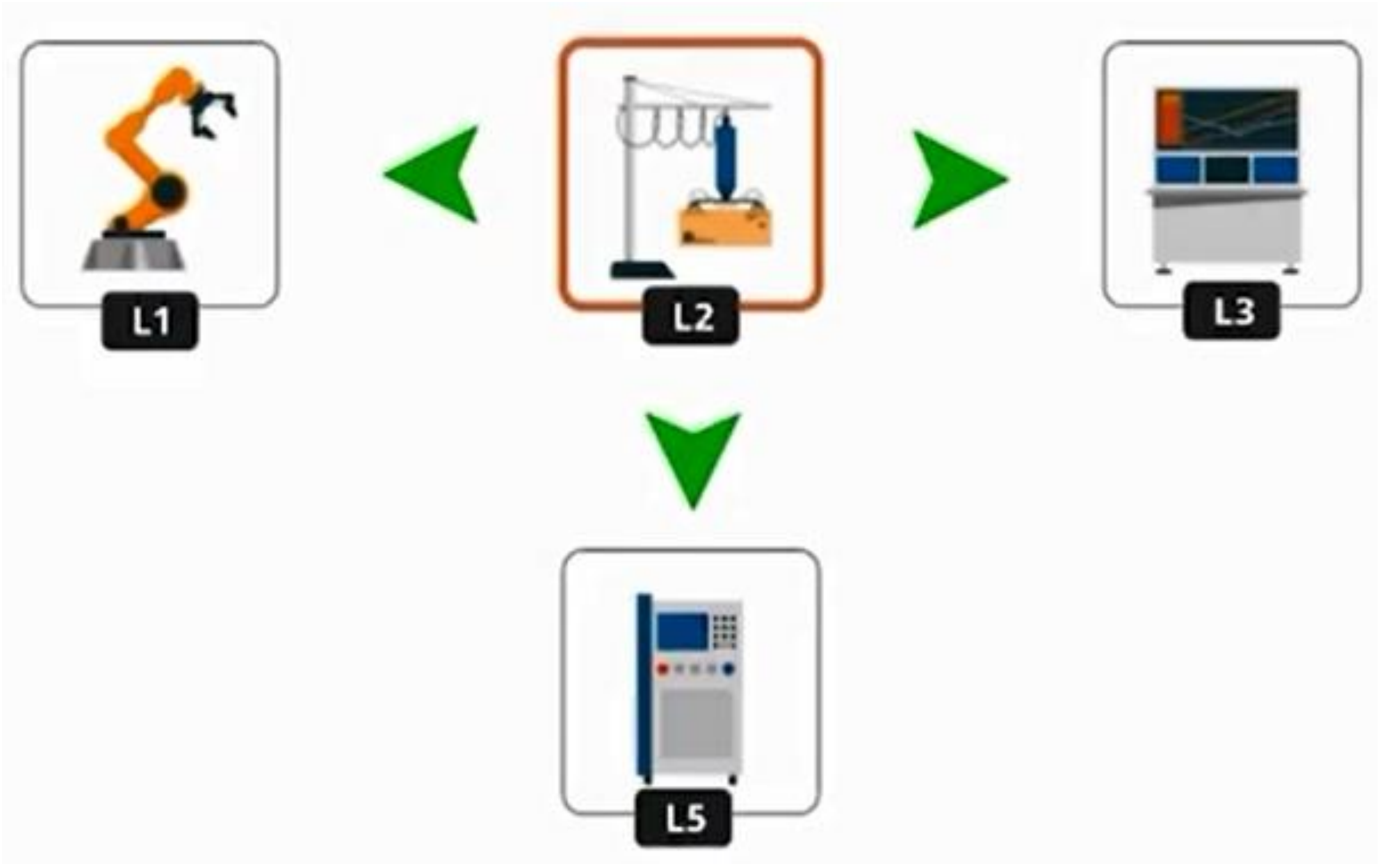
Q-Learning

Q-Learning-Problem Statement

Q-Learning-Problem Statement

Q-Learning-Problem Statement

Q-Learning-Action



Q-Learning-Action

Q-Learning-The Rewards

LIST OF STATES

$$S = 0, 1, 2, 3, 4, 5, 6, 7, 8$$

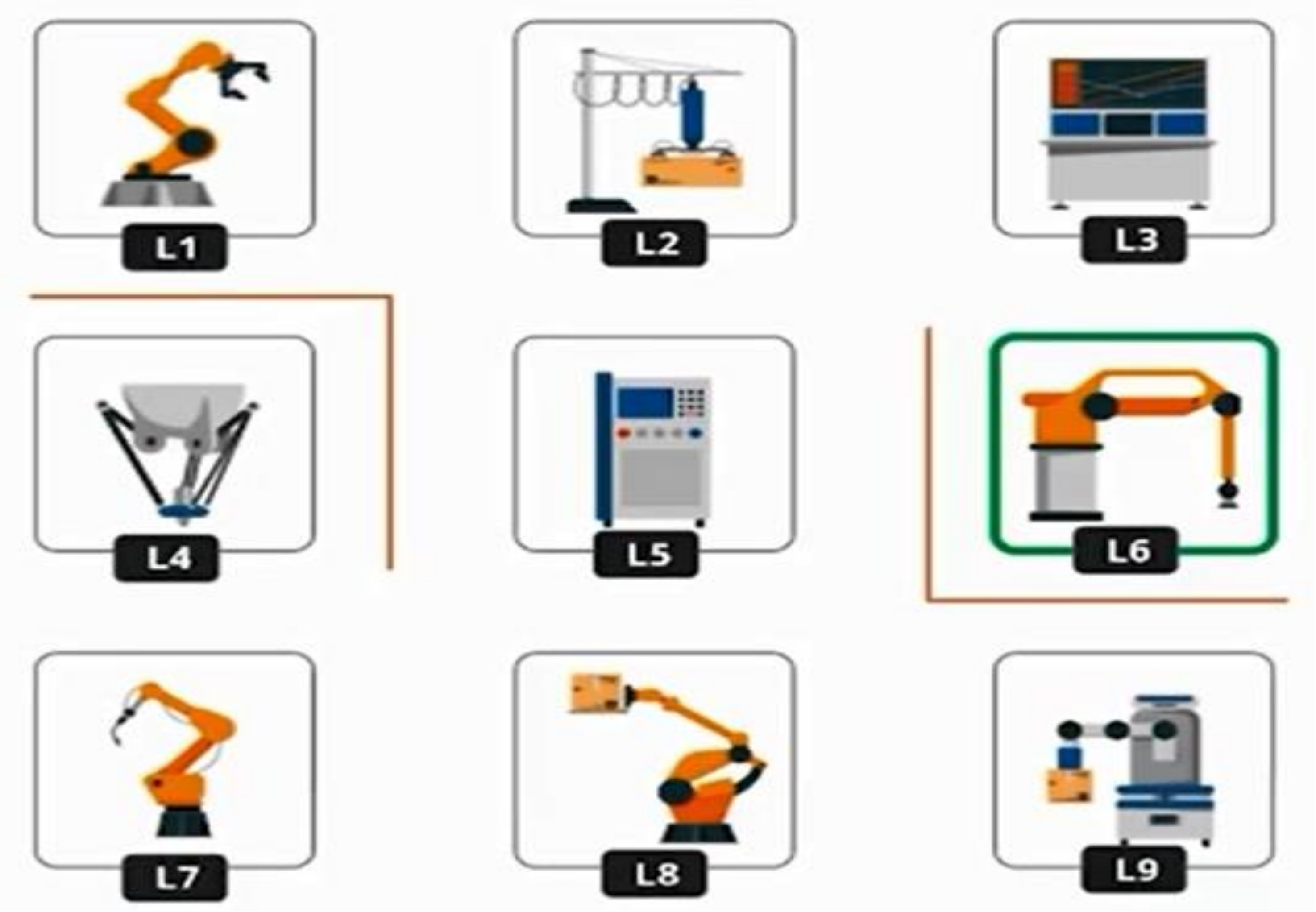
SET OF ACTIONS

$$A = 0, 1, 2, 3, 4, 5, 6, 7, 8$$

Q-Learning-Reward Table

	L1	L2	L3	L4	L5	L6	L7	L8	L9
L1	0	1	0	0	0	0	0	0	0
L2	1	0	1	0	0	0	0	0	0
L3	0	1	0	0	0	1	0	0	0
L4	0	0	0	0	0	0	1	0	0
L5	0	1	0	0	0	0	0	1	0
L6	0	0	1	0	0	0	0	0	0
L7	0	0	0	1	0	0	0	1	0
L8	0	0	0	0	1	0	1	0	1
L9	0	0	0	0	0	0	0	1	0

Q-Learning-Action

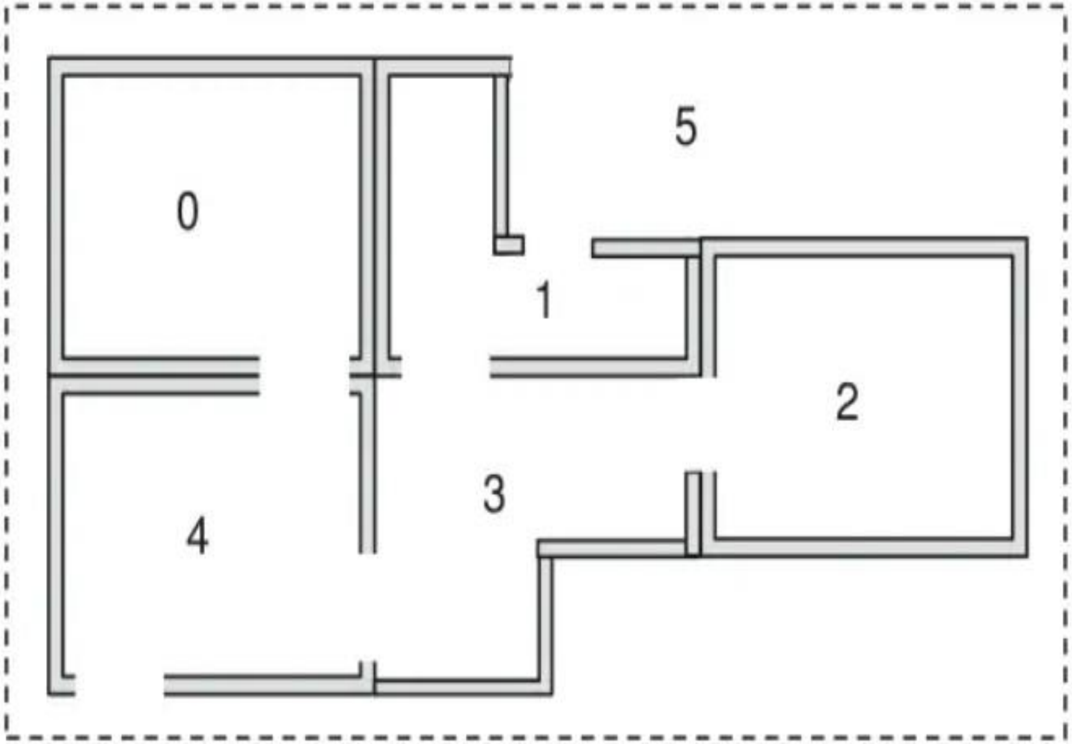


Q-Learning-Reward Table

	L1	L2	L3	L4	L5	L6	L7	L8	L9
L1	0	1	0	0	0	0	0	0	0
L2	1	0	1	0	0	0	0	0	0
L3	0	1	0	0	0	1	0	0	0
L4	0	0	0	0	0	0	1	0	0
L5	0	1	0	0	0	0	0	1	0
L6	0	0	1	0	0	999	0	0	0
L7	0	0	0	1	0	0	0	1	0
L8	0	0	0	0	1	0	1	0	1
L9	0	0	0	0	0	0	0	1	0

Understanding Q-Learning with Example

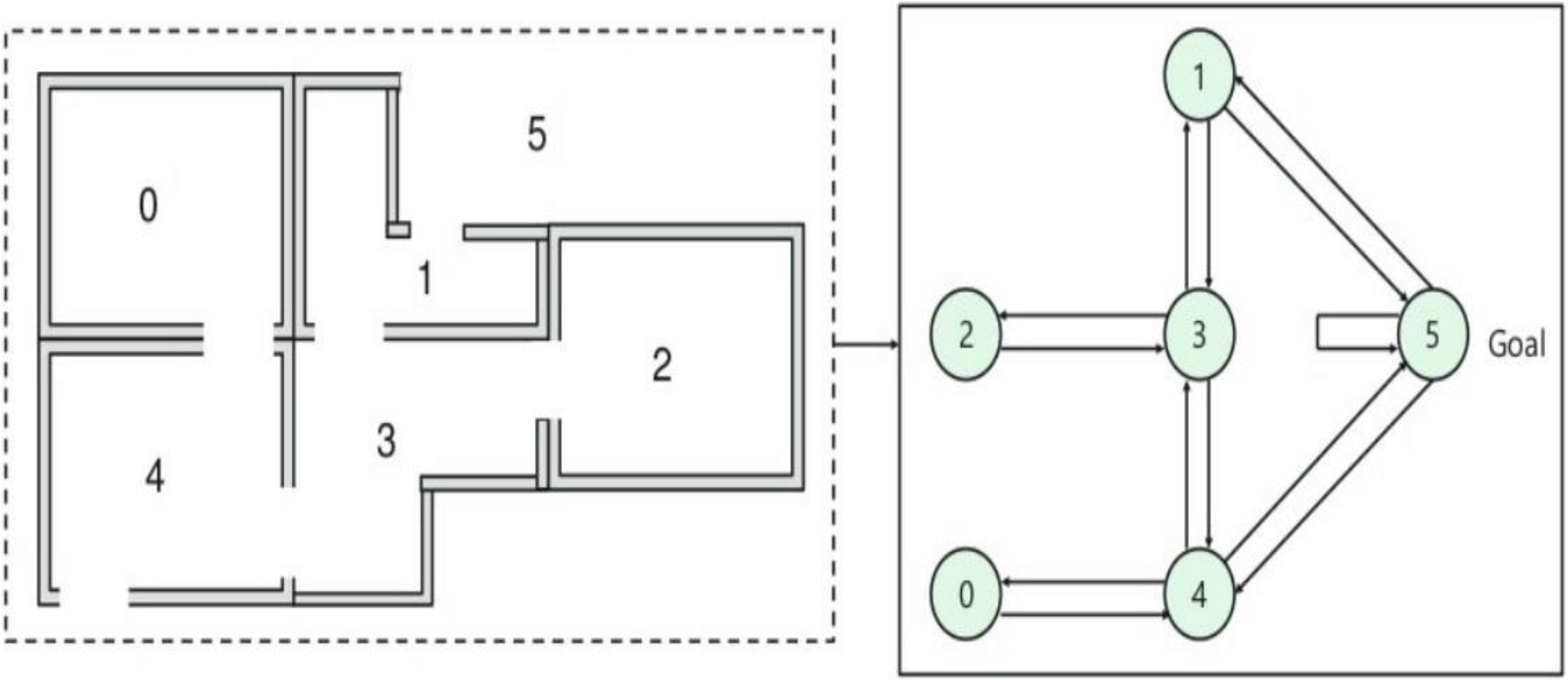
Place an agent in any one of the rooms (0,1,2,3,4) and the goal is to reach outside the building (room 5)



- 5 rooms in a building connected by doors
- each room is numbered 0 through 4
- The outside of the building can be thought of as one big room (5)
- Doors 1 and 4 lead into the building from room 5 (outside)

Understanding Q-Learning with Example

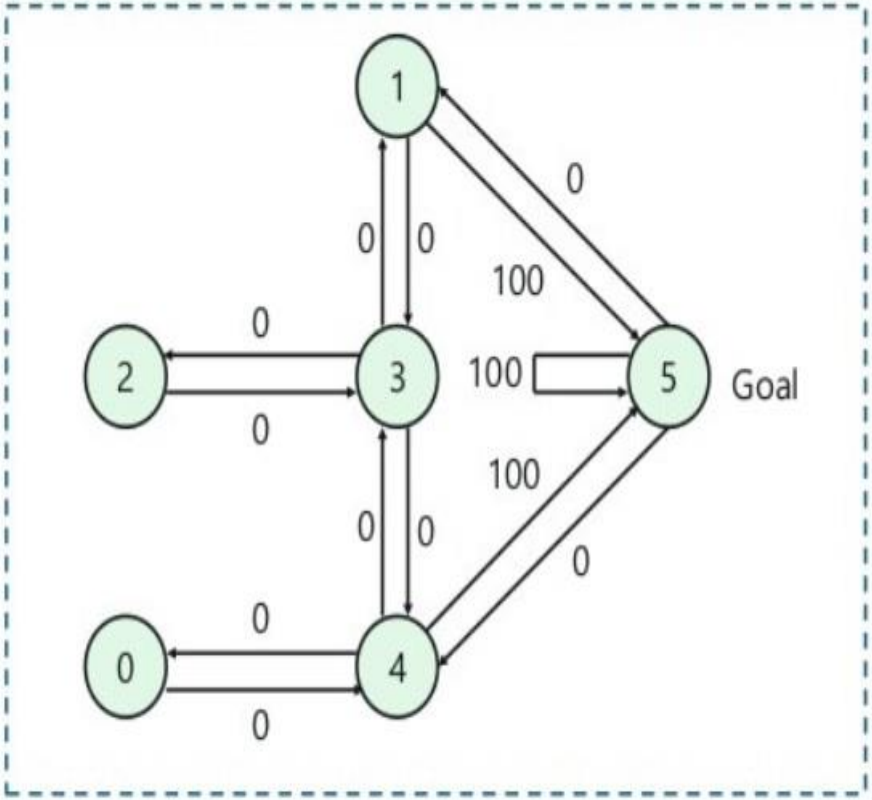
Let's represent the rooms on a graph, each room as a node, and each door as a link



Understanding Q-Learning with Example

Next step is to associate a reward value to each door:

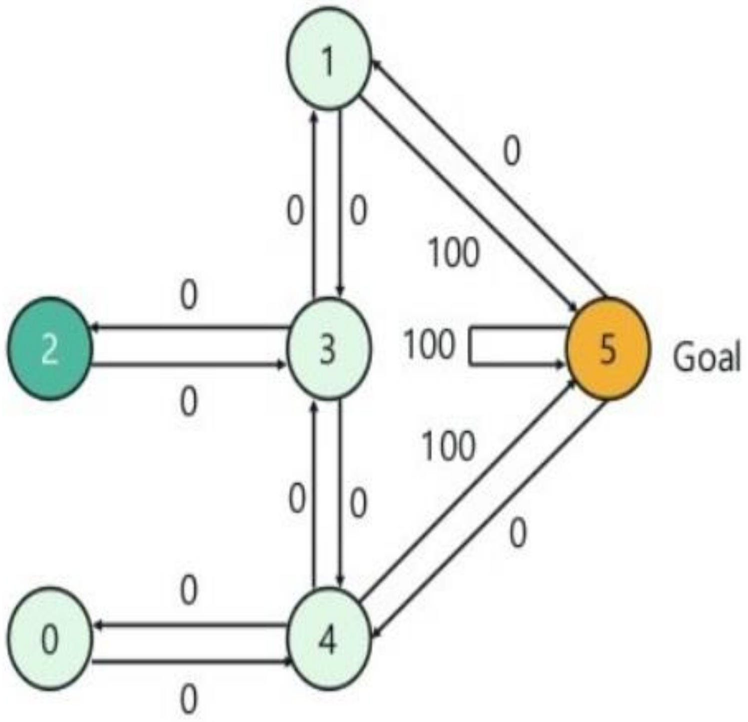
- doors that lead directly to the goal have a reward of 100
- Doors not directly connected to the target room have zero reward
- Because doors are two-way, two arrows are assigned to each room
- Each arrow contains an instant reward value



Understanding Q-Learning with Example

The terminology in Q-Learning includes the terms state and action:

- Room (including room 5) represents a state
- agent's movement from one room to another represents an action
- In the figure, a state is depicted as a node, while "action" is represented by the arrows

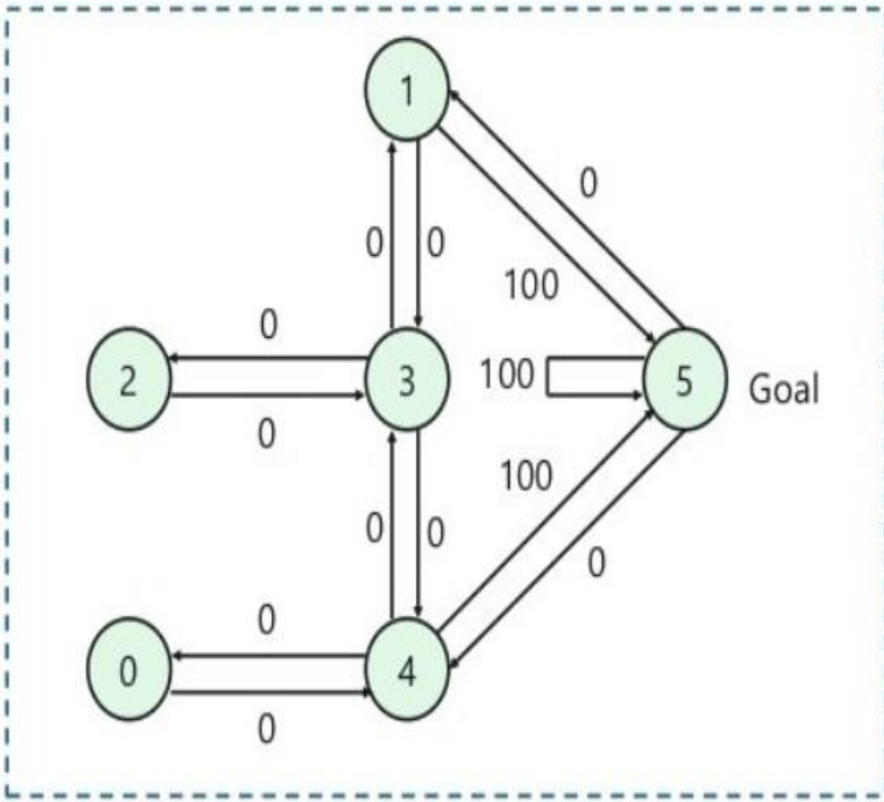


Example (Agent traverse from room 2 to room5):

1. Initial state = state 2
2. State 2 -> state 3
3. State 3 -> state (2, 1, 4)
4. State 4 -> state 5

Understanding Q-Learning with Example

We can put the state diagram and the instant reward values into a reward table, matrix R.



$$
R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\text{Action} \\
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\end{array}
$$

The -1's in the table represent null values

Understanding Q-Learning with Example

Add another matrix Q, representing the memory of what the agent has learned through experience.

- The rows of matrix Q represent the current state of the agent
- columns represent the possible actions leading to the next state
- Formula to calculate the Q matrix:

*Q(state, action) = R(state, action) + Gamma * Max [Q(next state, all actions)]*

Note

The Gamma parameter has a range of 0 to 1 (0 <= Gamma > 1).

- If Gamma is closer to zero, the agent will tend to consider only immediate rewards.
- If Gamma is closer to one, the agent will consider future rewards with greater weight

Q-Learning Algorithm

1. Set the gamma parameter, and environment rewards in matrix R
2. Initialize matrix Q to zero
3. Select a random initial state
4. Set initial state = current state
5. Select one among all possible actions for the current state
6. Using this possible action, consider going to the next state
7. Get maximum Q value for this next state based on all possible actions
8. Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
9. Repeat above steps until current state = goal state

Q-Learning Example

First step is to set the value of the learning parameter Gamma = 0.8, and the initial state as Room 1.

Next, initialize matrix Q as a zero matrix:

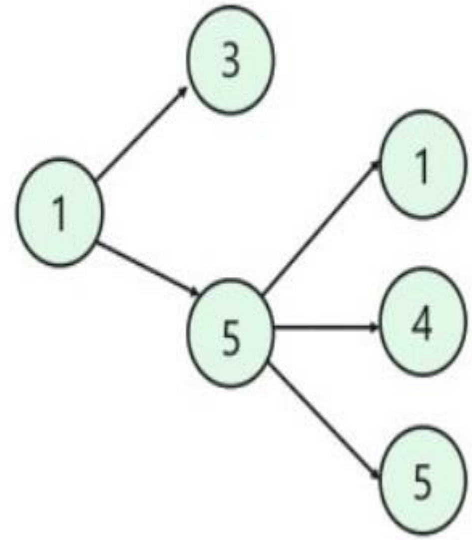
- From room 1 you can either go to room 3 or 5, let's select room 5.
- From room 5, calculate maximum Q value for this next state based on all possible actions:

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(1,5) = R(1,5) + 0.8 * Max[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$R = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$



Q-Learning Example

First step is to set the value of the learning parameter Gamma = 0.8, and the initial state as Room 1.

Next, initialize matrix Q as a zero matrix:

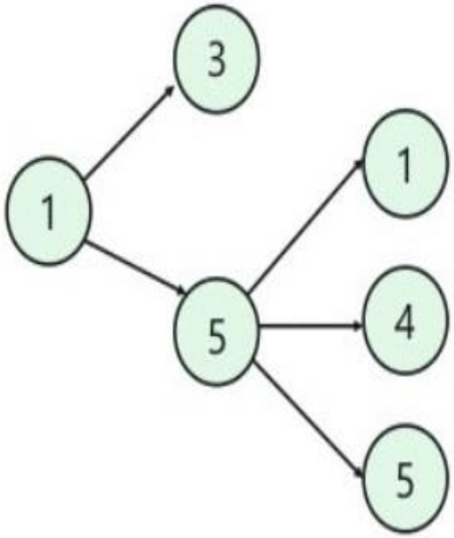
- From room 1 you can either go to room 3 or 5, let's select room 5.
- From room 5, calculate maximum Q value for this next state based on all possible actions

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(1,5) = R(1,5) + 0.8 * Max[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$R = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$



Q-Learning Example

For the next episode, we start with a randomly chosen initial state, i.e. state 3

- From room 3 you can either go to room 1,2 or 4, let's select room 1.
- From room 1, calculate maximum Q value for this next state based on all possible actions:

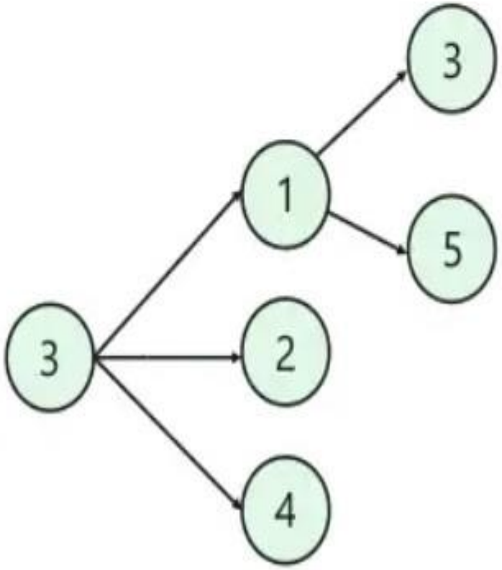
*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(3,1) = R(3,1) + 0.8 * Max[Q(1,3), Q(1,5)]= 0 + 0.8 * [0, 100] = 80

The matrix Q get's updated

$$Q = \begin{array}{cc} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$R = \begin{array}{cc} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$



Q-Learning Example

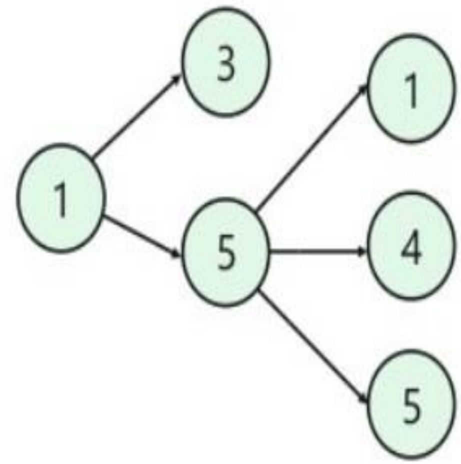
For the next episode, the next state, 1, now becomes the current state. We repeat the inner loop of the Q learning algorithm because state 1 is not the goal state.

- From room 1 you can either go to room 3 or 5, let's select room 5.
- From room 5, calculate maximum Q value for this next state based on all possible actions:

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

Q(1,5) = R(1,5) + 0.8 * Max[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8 * 0 = 100

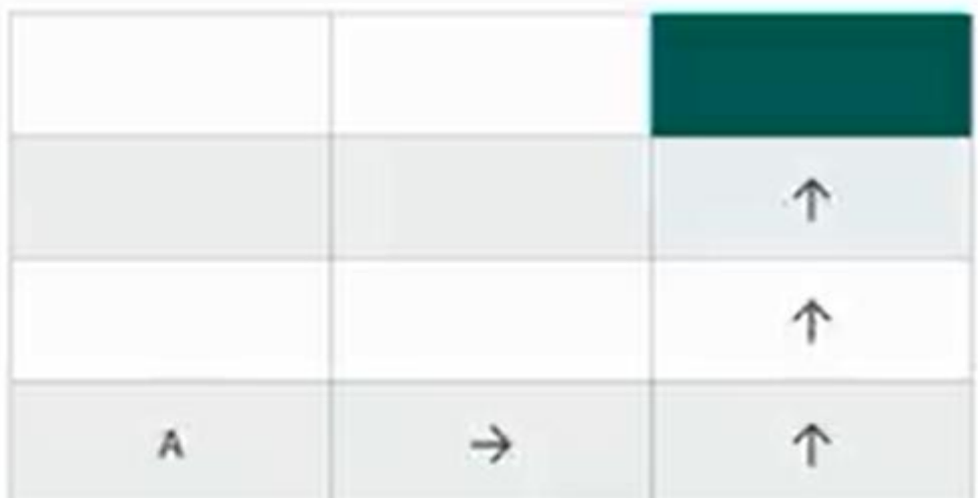
The matrix Q remains the same since, Q(1,5) is already fed to the agent



$$Q = \begin{array}{cc} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$R = \begin{array}{cc} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Bellman Equation



Bellman Equation

		1
		1
1	1	1

Bellman Equation

		1
		1
	1	A

Bellman Equation

$$V(s) = \max_a \left(R(s,a) + \gamma V\left(s'\right) \right)$$

s = a particular state (room)

a = action

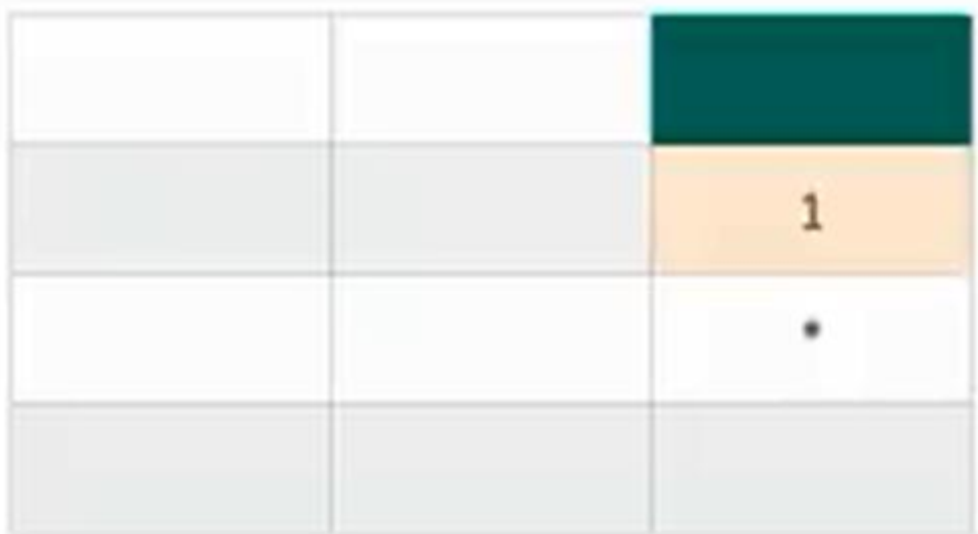
s' = state to which the robot goes from s

γ = discount factor

R(s, a) = a reward function which takes a state s and action a and outputs a reward value

V(s) = value of being in a particular state

Bellman Equation



Bellman Equation

$$V(s) = \max_a \left(R(s,a) + \gamma V\left(s'\right)\right)$$

Discount Factor = 0.9

$$V(s) = \max_a (0 + 0.9 * 1) = 0.9$$

Bellman Equation

		1
		0.9
START	0.729	0.81

Bellman Equation

0.9	1	
0.81	0.9	1
0.729	0.81	0.9
0.66	0.729	0.81

Applications

- Self Driving Cars
- Industry

  Automation

  Trading and

  Finance
- NLP : Text stigmatization. question answering and Language translation
- RL in Healthcare : Dynamic treatment regimes(DTRs)

Thank You
Merci
Salamat
Kop
Tack
Hvala
Gra
Xie
Diolch
Kiitos
Sheun
umesc
Kasih
Todah
Mamnoon
Shnorhakalutiun
Shokriya
Dziękuję
Shokran
Spaas
Mul
Ači
Gamsahapnida
Teşekkür
Dekuju/Dekujeme
Ngiyabonga
Dank
Takk
Cam
or
al
Dhanyavad
Grazie
Faleminderit
Dakujem
Waad
Daw
Dhanyavaadalu
Dankie
Go
Dhanyavad
Khopjai
Kruthagnathalu
kun
Arigatou
krap
Shukriya
Or
Dhonnobaad
ederim
Asante
Grazzi
raibh
Gracias
Nandree
Blagodariya
Gomapsupnida
Fyrir
Terima
Enkosi
Danke
dank
Euxaristo
Kun
Hain
Dhan
daa