REINFORCEMENT LEARNING (RL)

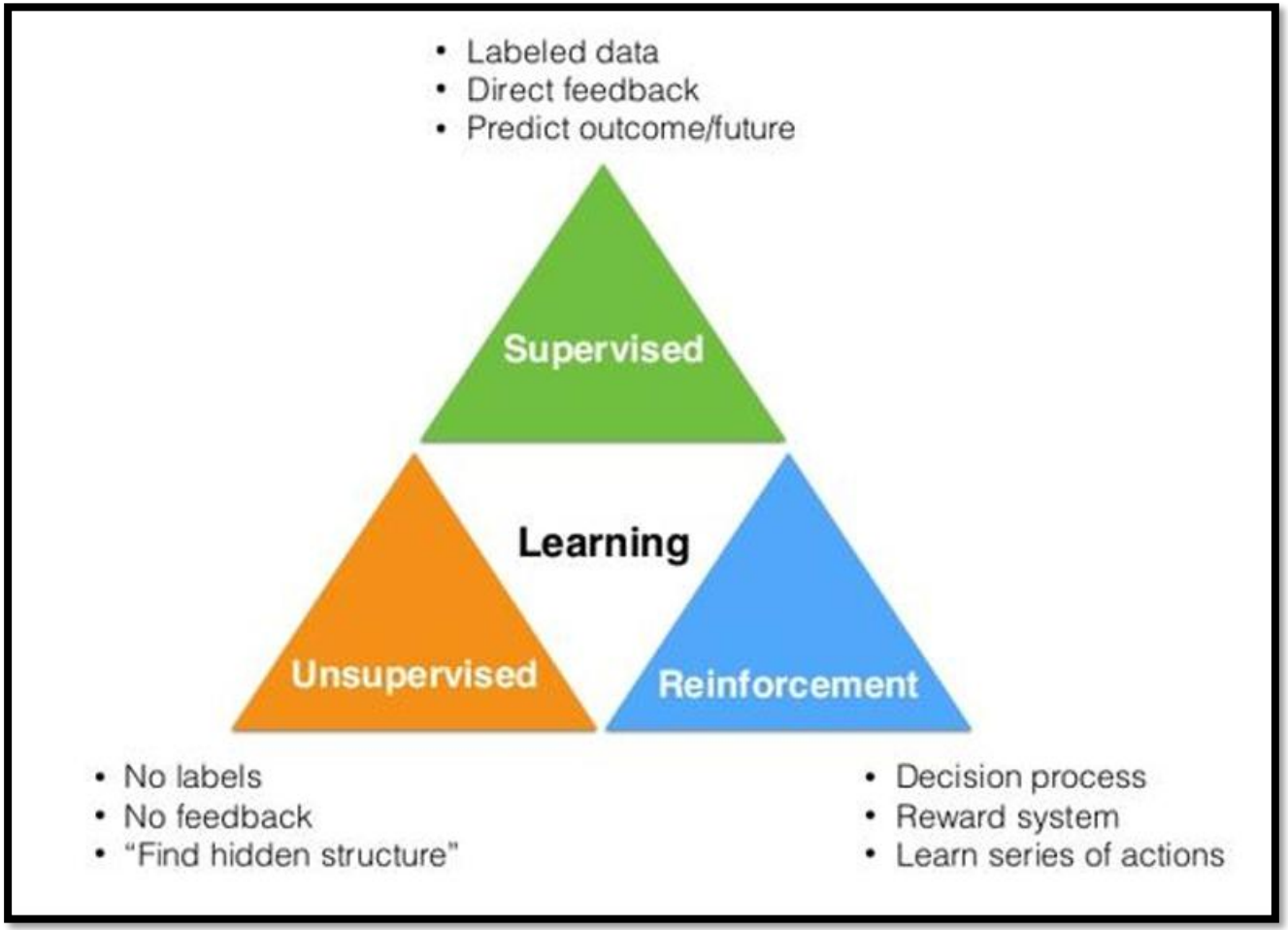
EC6301 – Artificial Intelligence

Ms. Yugani Gamlath
Department of Electrical & Information Engineering
Faculty of Engineering
University of Ruhuna

Definition of Reinforcement Learning

"Reinforcement is a class of machine learning where an agent learns how to behave in the environment by performing actions and thereby drawing intuitions and seeing the results"

Invented by Rich Sutton and Andrew Barto


• Labeled data
• Direct feedback
• Predict outcome/future
Supervised
Learning
Unsupervised
Reinforcement
• No labels
• No feedback
• "Find hidden structure"
• Decision process
• Reward system
• Learn series of actions

The Story

- In 2016, Deep Mind developed their original AlphaGo agent.
- Which played against, and beat, the 18-time world champion.
- In 2017, David Silver and others explained a new approach for training reinforcement learning agents to play adversarial games.
- They create a new agent for Go called AlphaGo Zero, that is totally self-trained.
- When fully trained, it beat the original AlphaGo agent 100 games to zero.

The Story

- https://youtu.be/tXlM99xPQC8
- AlphaGo - The Movie | Full award-winning documentary

Applications of Reinforcement Learning

- Self Driving Cars/Autonomous Driving
- Gaming
- Robotics
- Recommendation System
- Advertising and Marketing
- Industrial Logistics

Why Reinforcement Learning?

Where has this Reinforcement Learning come from when we have a good number of Machine Learning and Deep Learning techniques available at hand?

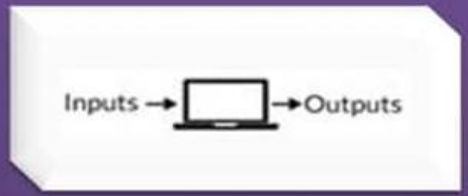
Reinforcement Learning supports automation by learning from the environment it is present in…

Supervised Learning

- Makes machine Learn explicitly
- Data with clearly defined output is given
- Direct feedback is given
- Predicts outcome/future
- Resolves classification and regression problems

Unsupervised Learning

- Machine understands the data (Identifies patterns/structures)
- Evaluation is qualitative or indirect
- Does not predict/find anything specific



Reinforcement Learning

- An approach to AI
- Reward based learning
- Learning form +ve & +ve reinforcement
- Machine Learns how to act in a certain environment
- To maximize rewards

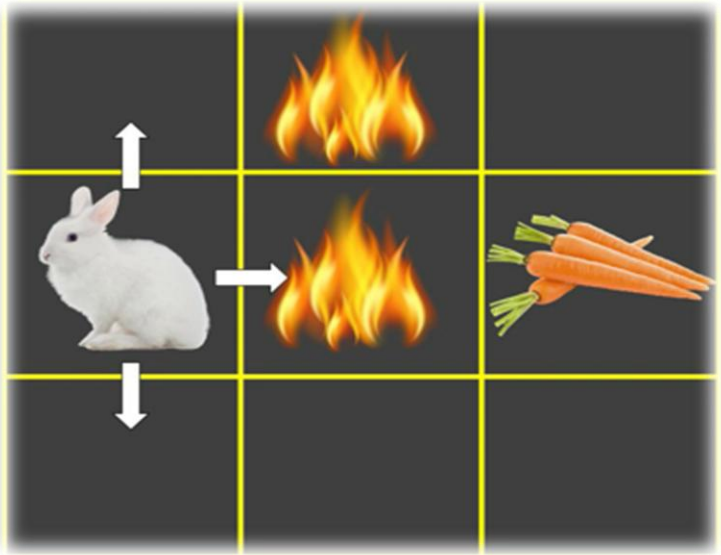
Let’s Start…

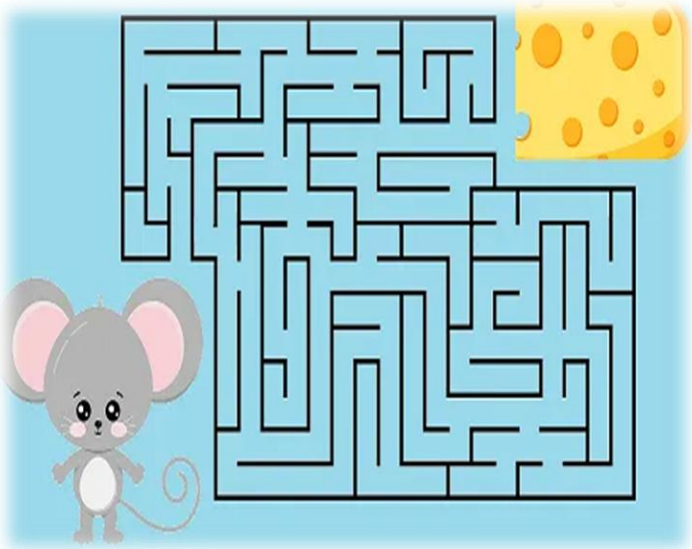
- It’s very much like the natural learning process wherein, the process/the model would be receiving feedback as to whether it has performed well or not.

Let’s Start…

- Deep Learning and Machine Learning, are most focused on finding patterns in the existing data.
- Reinforcement Learning, does this learning by trial-and-error method, and eventually, gets to the right actions or the global optimum.
- The significant additional advantage of Reinforcement Learning is that we need not provide the whole training data as in Supervised Learning.

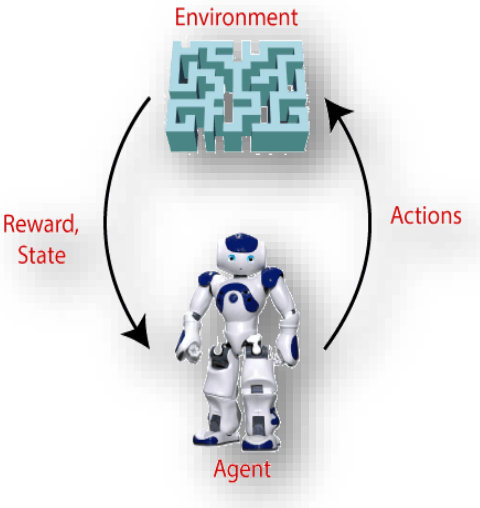
Introduction to Reinforcement Learning





Introduction to Reinforcement Learning

- Reinforcement Learning is a feedback-based Machine Learning approach.
- In here an agent learns to which actions to perform by looking at the environment and the results of actions.
- For each correct action, the agent gets positive feedback.
- For each incorrect action, the agent gets negative feedback or penalty.



Introduction to Reinforcement Learning

- Agent interacts with the environment and identifies the possible actions he can perform.
- Primary goal of an agent in reinforcement learning is to perform actions by looking at the environment and ***get the maximum positive rewards.***
- 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.
- Reinforcement learning is used to solve specific type of problem where decision making is sequential, and the goal is long-term, such as playing game, robotics.

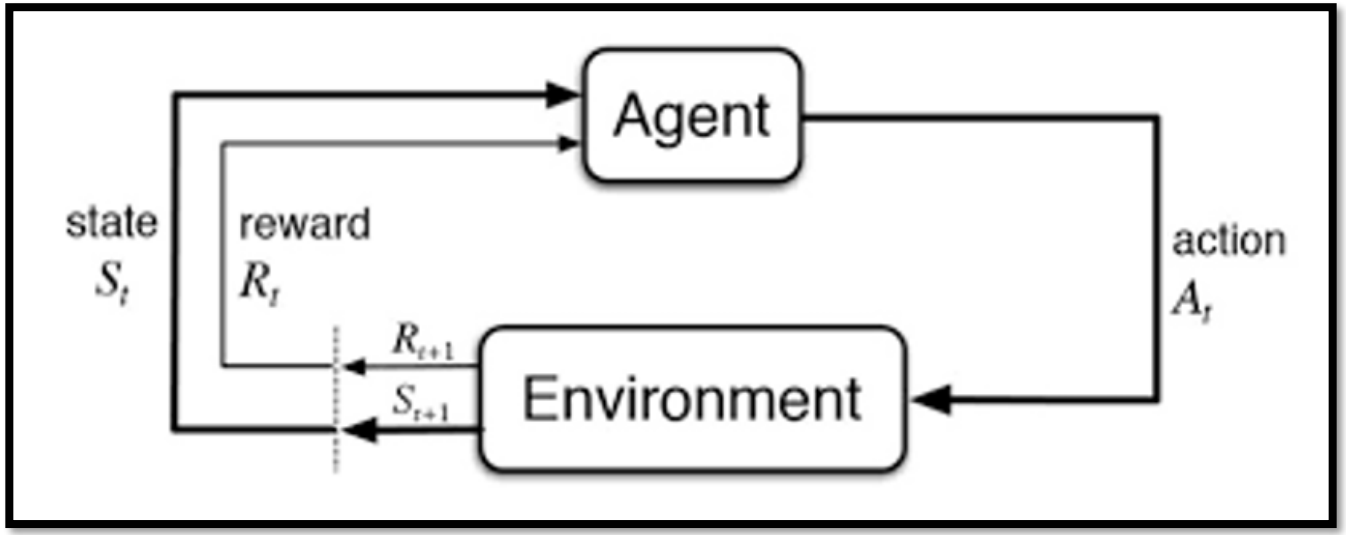
Reward maximization will be our end goal…

Key Terminalogies...

- Agent: The RL Algorithm that learns from trial and error.
- Action (a): All the possible moves that the agent can take.
- Environment (e): The world through which the agent moves.
- State (s): Current situation/condition returned by the environment.
- Reward (R): An immediate return sent back from the environment to evaluate the last action by the agent.
- Policy (∏): The approach that the agent uses to determine the next action based on the current state.

Markov Decision Process (MDP)

Mathematical approach for mapping a solution in Reinforcement Learning.



Q-Learning Algorithm

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state s
Do forever:

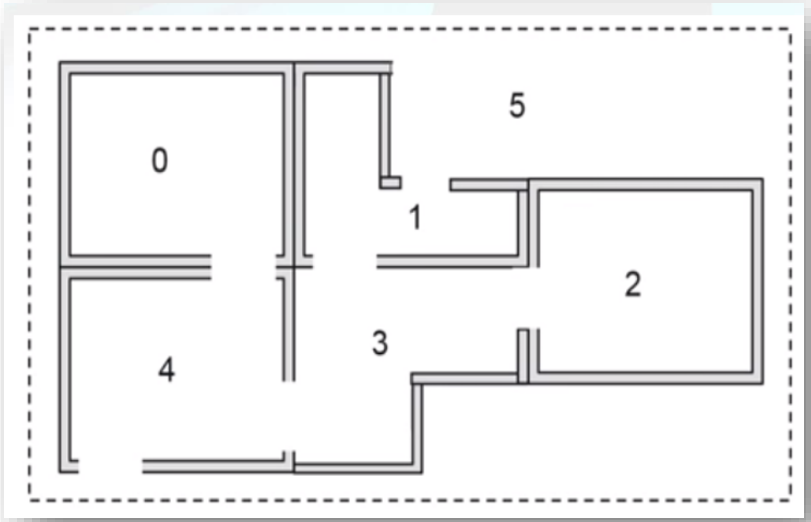
- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

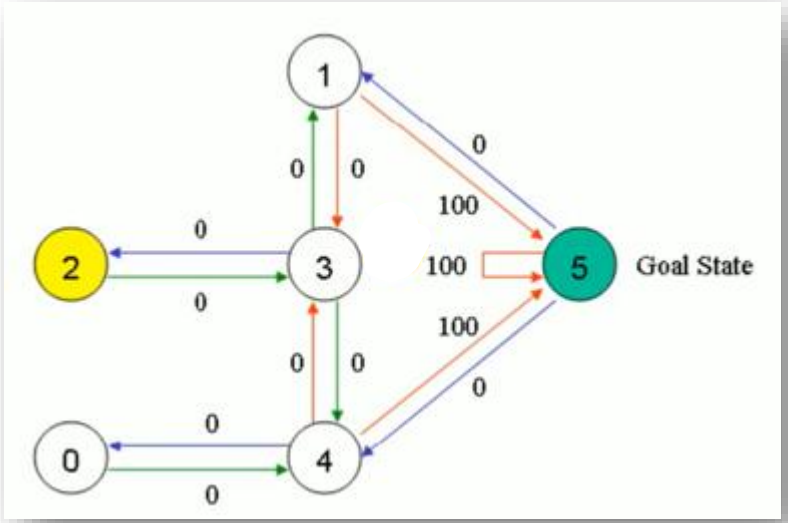
- $s \leftarrow s'$

Q-Learning Algorithm

- Place an agent in any one of the rooms.
- Goal is to reach outside of the building (state 5).

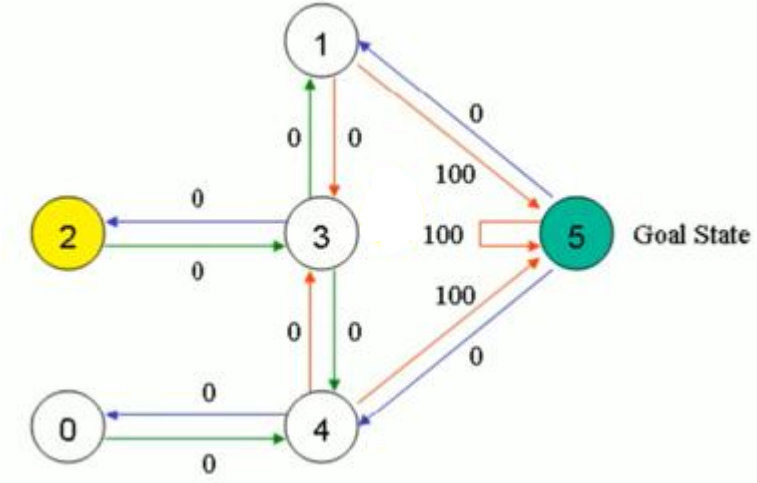


Q-Learning Algorithm



Q-Learning Algorithm

We can put the state diagram and the instant reward values into the following reward table (R). The -1's in the table represent null values. (There isn't a link between nodes)



$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Q-Learning Algorithm

- Take Gamma = 0.8 and the initial state as 1.
- Initialize matrix Q as a zero matrix.

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

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

Q-Learning Algorithm

- Look at the second row (state 1) of matrix R.
- There are 2 possible actions for the current state 1: go to state 3 or 5.
- By random selection, we select to go to 5 as our action.

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

Q-Learning Algorithm

Now let's imagine what would happen if our agent were in state 5.

Look at the sixth row of the reward matrix R (i.e. state 5).

It has 3 possible actions: go to state 1, 4 or 5.

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

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

Q-Learning Algorithm

- The next stage 5 now becomes the current state.
- Because 5 is the goal state, we've finished one episode.

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

Q-Learning Algorithm

For the next step, we randomly choose the initial state as 3 (can go to 1,2 and 4)

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

Q-Learning Algorithm

Now we imagine that we are in state 1 (next state).

Look at the second row of reward matrix R (i.e. state 1).

It has 2 possible actions: go to state 3 or state 5.

Then, we compute the Q value:

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

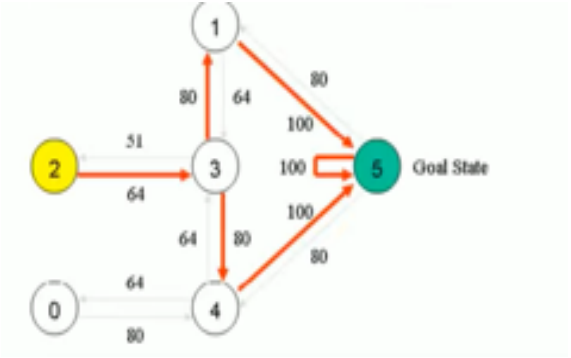
Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 3), Q(1, 5)] = 0 + 0.8 * Max(0, 100) = 80

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

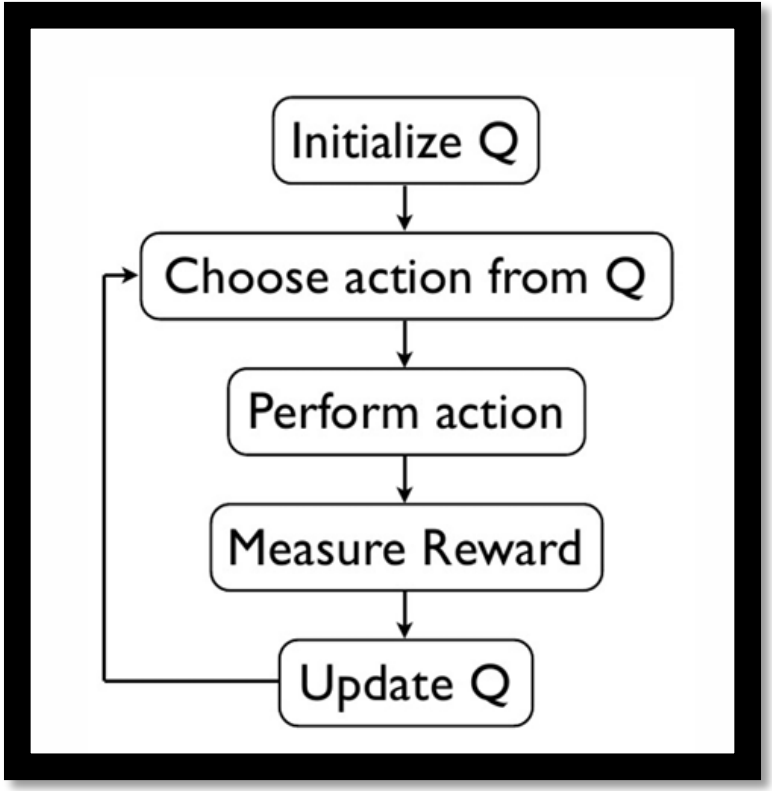
Q-Learning Algorithm

- If agent learns more through further episodes, it will finally reach convergence values in matrix Q.
- Tracing the best sequence of states is as following the links with highest values at each state.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$



Q-Learning Algorithm



When Not to Use RL?

- When you have enough data to solve the problem with a supervised learning method.
- RL is computing-heavy and time consuming when the action space is large.

Self Learning Activities...

- Supervised Learning vs Reinforcement Learning (Need to create a table)
- Find about 3 Different Approaches in Reinforcement Learning (Value based, Policy Based, Model Based).
- Find advantages and limitations related to the Reinforcement Learning and list down them.

Workshop 01
Q-Learning Algorithm Implementation

Write a Python program to perform Q-learning for the given 6×6 reward matrix (R).

The program should:

- Initialize the Q-matrix.
- Identify available actions from each state.
- Randomly select the next action from available actions.
- Update the Q-values using the Q-learning formula $Q(s, a) \leftarrow R(s, a) + \gamma \max Q\,(s', a')$.
- Normalize and print the final Q-matrix after 1000 iterations.

Thank You..