# Lab Assignment No.: 8 Reinforcement Learning

Problem Statement: Implement reinforcement learning and make a simple demonstrative game to show reinforcement learning

## Theory

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.

## Implementation

This code demonstrates a simple example of reinforcement learning in a grid world environment. Reinforcement learning is a type of machine learning where an agent learns to make a sequence of decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on its actions. Over time, the agent learns an optimal policy that maximizes its cumulative reward.

Grid World Environment:

The environment is represented as a grid world, which is a common scenario in reinforcement learning. The grid world is a 2D space where the agent can move around to reach a goal while avoiding obstacles. In this example, the grid is represented as a linear grid for simplicity, where the agent moves from one state to another.

Training the Agent:

The agent starts at a specific initial state and learns to navigate to the goal state. It uses a Q-learning algorithm to update its Q-values based on the rewards and penalties it receives during its exploration. The Q-values represent the agent's estimation of the expected cumulative rewards for each state-action pair.

Exploration and Exploitation:

The agent has a trade-off between exploration and exploitation. It explores the environment by taking random actions with a certain probability and exploits its learned Q-values to make optimal decisions.

```
import numpy as np

# Define the grid world and RL parameters

GRID_SIZE = 5
START = 0
GOAL = 24  # Assuming a linear grid world
```

```
OBSTACLES = [6, 12, 13, 18]
NUM_STATES = GRID_SIZE * GRID_SIZE
NUM_ACTIONS = 4  # Up, Down, Left, Right
NUM_EPISODES = 1000
LEARNING_RATE = 0.1
DISCOUNT_FACTOR = 0.9
EXPLORATION_PROB = 0.2

# Create a Q-table

q_table = np.zeros((NUM_STATES, NUM_ACTIONS))

# Function to take an action

def take_action(state, action):
    if action == 0:  # Up
        return max(0, state - GRID_SIZE)
    elif action == 1:  # Down
        return min(NUM_STATES - 1, state + GRID_SIZE)
    elif action == 2:  # Left
        return max(0, state - 1)
    elif action == 3:  # Right
        return min(NUM_STATES - 1, state + 1)

# Q-learning algorithm

for episode in range(NUM_EPISODES):
    state = START
    done = False

    while not done:
        if np.random.uniform(0, 1) < EXPLORATION_PROB:
            action = np.random.choice([0, 1, 2, 3])  # Choose a random
action
        else:
            action = np.argmax(q_table[state])

        next_state = take_action(state, action)

        if next_state in OBSTACLES:
            reward = -1
        elif next_state == GOAL:
            reward = 1
        else:
            reward = 0

        q_table[state][action] = (1 - LEARNING_RATE) * q_table[state]
[action] + \
            LEARNING_RATE * (reward + DISCOUNT_FACTOR *
np.max(q_table[next_state]))
```

```
        state = next_state

        if state == GOAL:
            done = True

# Test the trained agent

state = START
path = []

while state != GOAL:
    action = np.argmax(q_table[state])
    state = take_action(state, action)
    path.append(state)

print("Agent's path:", path)
```

```
Agent's path: [5, 10, 11, 16, 17, 22, 24]
```

Conclusion:

Reinforcement learning is a powerful paradigm that allows agents to learn from their interactions with the environment. In this code, the agent learns to navigate the grid world by trial and error, gradually discovering the most rewarding path while avoiding obstacles. The Q-learning algorithm is a fundamental technique within reinforcement learning that helps the agent make informed decisions. This example serves as a basic introduction to the principles of reinforcement learning in a grid world context.