Reinforcement Learning

Introduction to Reinforcement Learning (RL)

1. What is Reinforcement Learning?

Reinforcement Learning is a type of Machine Learning where an agent learns by interacting with an environment and receiving rewards or penalties for its actions.

2. Key Characteristics
- No labeled data
- Learning through trial and error
- Goal is to maximize cumulative reward

3. Where RL is Used
- Games
- Robotics
- Recommendation systems
- Self-driving cars
- Resource optimization

RL Components

1. Agent
The learner or decision maker.

2. Environment
Everything the agent interacts with.

3. State (S)
Current situation of agent.

4. Action (A)
Possible moves agent can make.

5. Reward (R)
Feedback from environment.

6. Policy
Strategy used by agent.

7. Episode
One complete sequence of interaction.

Lesson 3: Types of Reinforcement Learning

1. Positive Reinforcement
Reward for good action.

2. Negative Reinforcement
Penalty for bad action.

3. Model-Free RL
Learns without knowing environment model.

4. Model-Based RL
Uses environment model.

Lesson 4: Simple RL Algorithm - Q Learning

Q Learning updates values using:
Q(s,a) = Q(s,a) + alpha * (reward + gamma * max(Q(next_state)) - Q(s,a))

Where:
alpha = learning rate
gamma = discount factor

Lesson 5: Demo – Simple Grid World using Q Learning

Agent moves in 1D grid from position 0 to position 4.
Goal is to reach position 4.

Step 1: Import Libraries

```
import numpy as np
```

Step 2: Create Environment

```
states = 5
actions = 2  # 0 = left, 1 = right
Q = np.zeros((states, actions))
```

Step 3: Define Parameters

```
alpha = 0.8
gamma = 0.9
episodes = 1000
```

Step 4: Training Loop

```
for episode in range(episodes):
    state = 0
    while state != 4:
        action = np.random.randint(0,2)
        if action == 0:
            next_state = max(0, state-1)
        else:
            next_state = min(4, state+1)

        reward = 1 if next_state == 4 else -0.01

        Q[state, action] = Q[state, action] + alpha * (reward + gamma *
np.max(Q[next_state]) - Q[state, action])

        state = next_state
```

Step 5: View Q Table

```
print(Q)
```

Step 6: Test Trained Agent

```
state = 0
while state != 4:
    action = np.argmax(Q[state])
    if action == 0:
        state = max(0, state-1)
    else:
        state = min(4, state+1)
    print(state)
```

Agent learns to move right continuously to reach goal.

Lesson 7: Classroom Exercise

1. Change goal position to 6.
2. Add obstacle with negative reward.
3. Increase grid size.
4. Try different learning rates.

Lesson 8: Mini Assignment

Create a grid of size 10.
Place goal at position 9.
Train agent using Q-learning.
Submit Q-table and path output.