MLA03 REINFORCEMENT LEARNING

Software Required: Python IDLE
Library Required: GYM

UNIT-I

1. An autonomous cleaning robot navigates a 5x5 grid where certain cells contain dirt (reward: +1) and obstacles (penalty: -1). The robot starts at the top-left corner and must find an optimal policy to clean the entire grid efficiently. Implement the grid environment as an MDP and write a Python program to simulate the robot's navigation using different policies.

2. A robot navigates a warehouse to pick and place items. Define states (locations in the warehouse), actions (move in four directions), and rewards (picking an item: +2, reaching the goal: +5, hitting an obstacle: -2). Implement a policy evaluation algorithm to determine the value function for a given policy in Python.

3. An online retailer uses a multi-armed bandit approach to set prices dynamically. Simulate different pricing strategies using epsilon-greedy, UCB, and Thompson Sampling. Write a Python script to compare which strategy maximizes revenue over a series of pricing decisions.

4. A delivery drone needs to find the shortest path from a warehouse to multiple delivery points in a city represented as a grid. Implement a policy iteration algorithm using dynamic programming to find the optimal route policy in Python.

5. In a taxi dispatching system, define an MDP where states are locations, actions are move directions, and rewards are based on reaching pick-up points quickly. Write a Python program to use value iteration to find the optimal dispatch policy.

6. An online platform uses bandit algorithms to decide which advertisements to show to users. Implement epsilon-greedy, UCB, and Thompson Sampling algorithms. Use a Python script to determine which algorithm results in the highest click-through rate over time.

7. A delivery robot operates in a warehouse with predefined delivery points. Using Bellman equations, compute the state-value function for navigating to each delivery point. Implement this in Python and visualize the value function for different policies.

8. Simulate an autonomous car navigating a simple road network with intersections. Design policies for the car to follow traffic rules and reach the destination safely. Implement these policies in Python and evaluate their effectiveness.

9. A call center wants to optimize the assignment of customer service representatives to incoming calls. Implement a Monte Carlo simulation to estimate the value function for different assignment policies in Python.

10. A financial institution wants to optimize its investment strategy. Use a basic policy gradient method to simulate and optimize the investment policy for maximum returns. Implement this in Python.

MLA03 REINFORCEMENT LEARNING

11. Implement epsilon-greedy and softmax exploration strategies for an AI agent playing a simple game (e.g., Tic-Tac-Toe). Compare the performance of these strategies in terms of win rates and computational steps in Python.

12. A robot navigates a grid to perform tasks. Use Bellman's optimality equation to compute the optimal state-value function for the robot's navigation tasks. Implement this in Python and demonstrate the optimal path.
13. Set up an environment using OpenAI Gym and implement a policy to solve the MountainCar problem. Utilize Python libraries like Keras or TensorFlow to build and train the policy.

14. Simulate an RL framework to optimize a manufacturing process, where actions represent different machine settings and rewards are based on product quality. Implement the environment, policy, and value function in Python.

15. Use Monte Carlo methods to evaluate a policy for predicting customer churn in a subscription-based service. Implement this policy evaluation in Python and analyze the results.

16. Implement an epsilon-greedy strategy to optimize content recommendations on an online learning platform. Write a Python script to simulate and analyze its performance over multiple runs.

17. Use the Upper Confidence Bound (UCB) algorithm to dynamically select content for users on a streaming platform. Implement the UCB algorithm in Python and compare its effectiveness against other strategies.

18. Simulate a k-armed bandit problem to optimize marketing campaign choices. Implement epsilon-greedy, UCB, and Thompson Sampling algorithms in Python and evaluate their performance.

19. Explain the relationship between policy and value functions using a practical gridworld example. Implement this relationship in Python and visualize how different policies affect the value function.

20. In an inventory management system, use Bellman's equation to find the optimal policy for ordering stock. Implement this in Python and demonstrate how the optimal policy minimizes costs

UNIT II

1. Design a dynamic programming solution to optimize the traffic light timings at an intersection to minimize vehicle wait times. Define the states, actions, rewards, and transitions, and implement the policy iteration algorithm in Python.
2. An autonomous drone navigates a city grid to deliver packages. Using Monte Carlo control methods, implement a policy to optimize the drone's route to minimize delivery times and fuel consumption. Write the Python code to simulate this environment.

MLA03 REINFORCEMENT LEARNING

3. Implement the TD(0) algorithm to solve a maze where the agent must reach the goal while avoiding traps. Write a Python script to simulate the maze environment and apply the TD(0) algorithm to find the optimal path.

4. A robot vacuum cleaner navigates a house with various rooms and obstacles. Use the SARSA algorithm to learn the optimal cleaning policy that maximizes the cleaned area while minimizing energy usage. Implement this in Python.

5. Implement Q-learning to develop an AI agent that plays a simple grid-based game (e.g., a basic version of Pac-Man). The agent should learn to collect rewards (e.g., food) and avoid penalties (e.g., ghosts). Write a Python program to train and evaluate the AI agent.

6. An autonomous vehicle navigates a simulated highway environment. Use Deep Q-Networks (DQN) to train the vehicle to drive efficiently and safely. Implement the DQN algorithm using TensorFlow or Keras in Python.

7. Implement Double DQN to optimize a stock trading strategy. The agent should learn to buy, sell, or hold stocks to maximize profits. Write a Python script to simulate the trading environment and train the agent.

8. Use Dueling DQN to train an AI agent to play a simple video game (e.g., Pong). Implement the dueling architecture in Python and evaluate the agent's performance compared to standard DQN.

9. A robotic arm needs to learn to pick and place objects efficiently. Implement Prioritized Experience Replay in a DQN setup to improve the learning efficiency of the robotic arm. Write a Python program to simulate and train the robotic arm.

10. Implement dynamic programming methods to solve a gridworld navigation problem where the agent must reach the goal with the least number of steps while avoiding obstacles. Use Python to simulate the environment and policy iteration.

11. A call center uses Monte Carlo methods to optimize the assignment of customer service representatives to incoming calls. Implement Monte Carlo policy control in Python to minimize average call handling time.

12. Use the TD(0) algorithm to optimize a financial portfolio by learning the best asset allocation strategy. Implement this in Python to simulate the environment and learn the optimal policy.

13. Implement the SARSA algorithm to develop an AI agent that learns to play a simple board game (e.g., Tic-Tac-Toe). Write a Python program to train and evaluate the agent's performance.

14. An autonomous delivery robot navigates a warehouse to deliver packages. Implement Q-learning to train the robot to find the shortest and safest path to its destinations. Write the Python code for this simulation.

15. Use DQN to solve the CartPole balancing problem in OpenAI Gym. Implement the DQN algorithm using TensorFlow or Keras in Python and train the agent to balance the pole on the cart.

16. Implement Double DQN to optimize energy management in a smart home. The agent should learn to control devices to minimize energy consumption while maintaining comfort. Write a Python script to simulate and train the agent.

17. Use Dueling DQN to train an AI agent for optimal navigation in a complex gridworld environment. Implement the dueling architecture in Python and compare its performance with standard DQN.

18. An autonomous car navigates a city grid. Implement Prioritized Experience Replay in a DQN setup to improve the learning efficiency of the car. Write a Python program to simulate and train the car.

19. Use dynamic programming to optimize a trading strategy for a financial market. Implement policy iteration in Python to maximize long-term profits

20. Train a simulated humanoid robot to walk and navigate complex terrains using Proximal Policy Optimization (PPO). Implement PPO in Python to learn locomotion policies that enable the robot to maintain balance and traverse diverse environments.

<u>Unit III</u>

1. Implement an autonomous cleaning robot that navigates a 5x5 grid with cells containing dirt (reward: +1) and obstacles (penalty: -1). The robot starts at the top-left corner and must find an optimal policy to clean the entire grid efficiently using the REINFORCE algorithm.

2. Model a self-driving car navigating a 4-way intersection with traffic lights. The car must minimize wait time (reward: -1 per second) and avoid collisions (penalty: -10) using an Actor-Critic (A2C) algorithm to simulate the car’s navigation.

3. Implement a warehouse robot that picks items from shelves and delivers them to a packing station, avoiding collisions and optimizing its path using a Proximal Policy Optimization (PPO) algorithm.

4. Develop an agent to play Tic-Tac-Toe against a human opponent, with rewards for winning and penalties for losing, and train the agent using Vanilla Policy Gradient methods.

5. Model a smart grid that manages energy consumption and production to minimize costs and balance supply and demand using Trust Region Policy Optimization (TRPO) to optimize energy management.
6. Develop a recommendation system for a streaming service to suggest movies based on user feedback, implemented as an MDP and trained using a Deep Deterministic Policy Gradient (DDPG) algorithm.

MLA03 REINFORCEMENT LEARNING

7. Implement an agent that manages a financial portfolio, choosing stocks to maximize returns and minimize risk using an Actor-Critic (A3C) method to optimize investmentImplement a robot that navigates a maze to reach the exit, with rewards for reaching the exit and penalties for hitting walls, and use REINFORCE to find the optimal navigation policy.

8. Model traffic signals at an intersection that adapt to changing traffic conditions to minimize wait times and congestion using PPO to optimize signal timing.

9. Implement an e-commerce platform that adjusts product prices dynamically based on demand and competition using the TRPO algorithm to optimize pricing strategies.

10. Model a drone that delivers packages to various city locations, navigating around buildings and no-fly zones using DDPG to optimize the drone's delivery routes.

11. Develop a system to recommend personalized treatment plans for patients based on medical history using an A2C method to optimize treatment recommendations.

12. Implement an autonomous vehicle that changes lanes on a highway to avoid slower traffic and reach its destination faster using PPO to optimize lane-changing decisions.

13. Model a retail store that manages its inventory to meet customer demand while minimizing costs using TRPO to optimize inventory levels.

14. Develop an AI agent for a real-time strategy game to gather resources, build units, and defeat opponents using DDPG to train the agent.

15. Implement an app that helps users manage personal finances by suggesting budgeting and investment decisions using PPO to optimize financial health.

16. Model a smart home system that adjusts heating and cooling to maintain comfort while minimizing energy usage using the REINFORCE algorithm to optimize temperature settings.

17. Implement a robot that plays soccer, learning to score goals while avoiding opponents using an Actor-Critic (A3C) method to train the robot.

18. Develop a bot to trade stocks autonomously, maximizing profits while managing risks using a DDPG algorithm to optimize trading strategies.

19. Implement an online learning platform that adapts to students' progress, providing personalized learning paths using an A2C method to optimize learning paths.

20. Develop an AI agent to play a real-time strategy game (e.g., Age of Empires) using Actor-Critic methods. Implement the actor and critic networks in Python and train the agent to build structures, gather resources, and engage in strategic combat.

21. Train an AI agent to compete in autonomous vehicle racing competitions using Advantage Actor-Critic (A2C) methods. Implement A2C in Python to learn aggressive driving policies that optimize lap times and race performance.

22. Train a virtual character to create engaging content (e.g., storytelling, interactive experiences) within a simulated virtual world using policy gradient methods. Implement the policy gradient algorithm in Python to optimize the character's behavior for maximum audience engagement.

23. Develop an automated trading system using the REINFORCE algorithm. Implement REINFORCE in Python to learn trading strategies that maximize profit and minimize risk in financial markets.

Unit IV

1. Consider a robotic arm tasked with reaching a target position in 3D space. Implement an analytic gradient computation approach to optimize the control policy of the robotic arm using model-based RL. Write a Python script to simulate the robotic arm's motion and demonstrate the optimization process.

2. An autonomous exploration robot needs to navigate and map an unknown environment. Implement a sampling-based planning algorithm (e.g., RRT, RRT*, or PRM) to plan collision-free paths for the robot to explore efficiently. Write a Python program to simulate the robot's exploration process and visualize the generated paths.

3. A retail company aims to optimize its inventory management strategy using model-based RL. Develop a data generation model that simulates customer demand patterns and inventory dynamics. Use Python to generate synthetic data and evaluate different inventory management policies based on the simulated environment.

4. Implement a value-equivalence prediction model to estimate the long-term performance of different investment portfolios. Use historical financial data and machine learning techniques to predict the value equivalence of alternative portfolio allocations. Write a Python program to analyze and compare the predicted performances of various investment strategies.

5. A dynamic pricing platform aims to optimize its pricing strategy using model-based RL. Develop a predictive model that forecasts customer demand and price sensitivities based on historical sales data. Use Python to train the predictive model and implement a model-based policy optimization algorithm to dynamically adjust prices in response to changing market conditions.

6. A mobile robot navigates through a cluttered environment to reach a goal position while avoiding obstacles. Develop an analytic gradient computation method to optimize the robot's path planning trajectory using model-based RL. Implement this approach in Python and visualize the robot's trajectory in the simulated environment.

7. An unmanned aerial vehicle (UAV) is deployed for surveillance missions in a dynamic urban environment. Implement a sampling-based planning algorithm (e.g., RRT, RRT*,

or PRM) to plan collision-free flight paths for the UAV while maximizing coverage and minimizing detection latency. Write a Python program to simulate the UAV's surveillance mission and visualize the planned paths.

8. A logistics company aims to optimize its supply chain operations using model-based RL. Develop a data generation model that simulates order fulfillment processes, inventory flows, and transportation networks. Use Python to generate synthetic data and evaluate different supply chain management policies based on the simulated environment.

9. Implement a value-equivalence prediction model to estimate the long-term performance of different cryptocurrency investment portfolios. Use historical cryptocurrency price data and machine learning techniques to predict the value equivalence of alternative investment strategies. Write a Python program to analyze and compare the predicted performances of various cryptocurrency portfolios.

10. A utility company aims to optimize its smart grid management strategy using model-based RL. Develop a predictive model that forecasts electricity demand and renewable energy generation based on historical consumption patterns and weather data. Use Python to train the predictive model and implement a model-based policy optimization algorithm to optimize energy dispatch and pricing decisions.

11. A cloud service provider dynamically allocates resources (e.g., virtual machines, storage) to meet varying customer demands while minimizing operational costs. Develop an analytic gradient computation method to optimize resource allocation decisions using model-based RL. Implement this approach in Python and evaluate its performance in a simulated cloud environment.

12. An autonomous agricultural robot is deployed for field monitoring and crop management tasks. Implement a sampling-based planning algorithm (e.g., RRT, RRT*, or PRM) to plan efficient trajectories for the robot to navigate through agricultural fields and perform tasks such as soil sampling and weed detection. Write a Python program to simulate the robot's operations and visualize the planned trajectories.

13. A healthcare facility aims to optimize its resource allocation decisions (e.g., staff scheduling, equipment deployment) using model-based RL. Develop a data generation model that simulates patient admission rates, treatment durations, and resource utilization patterns. Use Python to generate synthetic data and evaluate different resource allocation policies based on the simulated environment.

14. Implement a value-equivalence prediction model to estimate the long-term returns of different real estate investment portfolios. Use historical property price data and machine learning techniques to predict the value equivalence of alternative investment strategies. Write a Python program to analyze and compare the predicted performances of various real estate portfolios.

15. A city transportation department aims to optimize its traffic management strategy using model-based RL. Develop a predictive model that forecasts traffic flow, congestion levels, and travel times based on historical traffic data and real-time sensor readings. Use Python to train the predictive model and implement a model-based policy optimization algorithm to optimize traffic signal timings and route guidance decisions.

16. A robotic manipulator arm performs complex manipulation tasks in a manufacturing environment. Develop an analytic gradient computation method to optimize the control policy of the manipulator arm using model-based RL. Implement this approach in Python and evaluate its performance in simulating precise and efficient manipulation actions.

17. An autonomous underwater vehicle (AUV) explores the ocean floor to survey marine ecosystems and geological features. Implement a sampling-based planning algorithm (e.g., RRT, RRT*, or PRM) to plan collision-free paths for the AUV to navigate through underwater terrain and collect data. Write a Python program to simulate the AUV's exploration mission and visualize the planned paths.

18. An asset management firm aims to optimize its investment decisions using model-based RL. Develop a data generation model that simulates financial market conditions, asset price movements, and economic indicators. Use Python to generate synthetic data and evaluate different investment strategies based on the simulated environment.

19. Implement a value-equivalence prediction model to estimate the long-term performance of diversified investment portfolios. Use historical financial market data and machine learning techniques to predict the value equivalence of alternative portfolio diversification strategies. Write a Python program to analyze and compare the predicted performances of various diversified portfolios.

20. An energy management system aims to optimize its energy production and distribution strategy using model-based RL. Develop a predictive model that forecasts energy demand, renewable energy generation, and grid stability based on historical consumption patterns and weather forecasts. Use Python to train the predictive model and implement a model-based policy optimization algorithm to optimize energy production, storage, and distribution decision.

21. A retail chain manages inventory across multiple stores and warehouses. Develop an analytic gradient computation method to optimize inventory replenishment decisions using model-based RL. Implement this approach in Python and assess its effectiveness in reducing stockouts and excess inventory levels.

22. A team of autonomous robots is deployed for disaster response tasks in a simulated disaster scenario (e.g., earthquake aftermath). Implement a sampling-based planning algorithm (e.g., RRT, RRT*, or PRM) to plan coordinated paths for the robots to search for survivors and deliver supplies in a hazardous environment. Write a Python program to simulate the robots' operations and visualize the planned paths.

23. A public health agency aims to predict the spread of infectious diseases and assess the effectiveness of intervention strategies using model-based RL. Develop a data generation model that simulates disease transmission dynamics, population movement patterns, and healthcare system capacities. Use Python to generate synthetic data and evaluate different intervention policies based on the simulated environment.

24. Implement a value-equivalence prediction model to estimate the long-term profitability of different cryptocurrency trading strategies. Use historical cryptocurrency market data and machine learning techniques to predict the value equivalence of alternative trading

approaches. Write a Python program to analyze and compare the predicted performances of various trading strategies.

25. A smart building system aims to optimize its climate control strategy using model-based RL. Develop a predictive model that forecasts indoor temperature, occupancy patterns, and energy consumption based on historical data and weather forecasts. Use Python to train the predictive model and implement a model-based policy optimization algorithm to adjust heating, ventilation, and air conditioning (HVAC) settings dynamically to maintain comfort while minimizing energy costs

<u>Unit – V</u>

1. An autonomous vehicle navigates through a complex urban environment with varying levels of granularity. Implement a hierarchical reinforcement learning approach to enable the vehicle to navigate efficiently at both high-level decision-making (e.g., route planning) and low-level control (e.g., obstacle avoidance). Write a Python program to simulate the vehicle's navigation process and visualize the hierarchical decision-making.

2. A team of autonomous robots collaborates to complete a task in a dynamic environment with multiple subtasks and dependencies. Implement hierarchical abstract machines (HAMs) to coordinate the robots' actions at different levels of abstraction, enabling them to achieve complex objectives efficiently. Write a Python program to simulate the robots' coordination process and visualize the hierarchical task decomposition.

3. Multiple agents collaborate to solve a cooperative task in a simulated environment with hierarchical structure and interdependencies. Implement the MAXQ framework to decompose the task into hierarchically organized subtasks and learn policies for each level of the hierarchy. Write a Python program to simulate the agents' interactions and evaluate their performance in achieving the overall task objectives.

4. An adaptive control system aims to adapt its control policy to different operating conditions and environmental changes without explicit retraining. Implement a meta-learning approach to enable the control system to learn how to adapt its parameters and structure based on past experiences and performance feedback. Write a Python program to simulate the control system's adaptation process and evaluate its performance under various conditions.

5. A team of autonomous robots collaborates to accomplish a complex task (e.g., search and rescue, package delivery) in a dynamic and uncertain environment. Implement multi-agent reinforcement learning algorithms (e.g., MADDPG, COMA) to enable the robots to coordinate their actions and achieve collective objectives efficiently. Write a Python program to simulate the robots' interactions and evaluate their performance in completing the task.

6. An autonomous robot navigates through a partially observable environment with limited sensor information and uncertainty. Implement a partially observable Markov decision process (POMDP) framework to enable the robot to localize itself and plan navigation actions robustly under partial observability. Write a Python program to simulate the robot's navigation process and evaluate its performance in different scenarios.

7. An autonomous system is designed to make decisions that may have ethical implications (e.g., autonomous vehicles, medical diagnosis systems). Implement ethical considerations into the system's decision-making process using reinforcement learning techniques, such as incorporating fairness constraints or avoiding harm to humans. Write a Python program to simulate the system's decision-making and evaluate its ethical performance.

8. Apply reinforcement learning techniques to optimize healthcare management processes, such as patient scheduling, resource allocation, or treatment planning. Develop a simulation environment that models patient flow, resource constraints, and treatment outcomes, and use reinforcement learning algorithms to learn policies that maximize patient outcomes while minimizing costs or resource utilization. Write a Python program to simulate the healthcare management processes and evaluate the learned policies.

9. Apply reinforcement learning techniques to optimize the management of natural resources (e.g., water, energy, forests) or sustainable infrastructure systems (e.g., smart grids, transportation networks). Develop a simulation environment that models resource dynamics, environmental constraints, and policy interventions, and use reinforcement learning algorithms to learn policies that balance resource utilization, economic efficiency, and environmental sustainability. Write a Python program to simulate the resource management processes and evaluate the learned policies.

10. Apply reinforcement learning techniques to personalize educational experiences for students in online learning platforms or intelligent tutoring systems. Develop a simulation environment that models student learning dynamics, curriculum content, and teaching interventions, and use reinforcement learning algorithms to learn policies that adaptively tailor educational content and interventions to individual student needs and preferences. Write a Python program to simulate the educational processes and evaluate the personalized learning policies.

11. An autonomous robot performs complex manipulation tasks (e.g., object assembly, tool manipulation) in a dynamic environment with multiple subgoals and dependencies. Implement hierarchical reinforcement learning techniques to decompose the manipulation task into hierarchically organized subtasks and learn policies for each level of the hierarchy, enabling the robot to achieve the overall task efficiently. Write a Python program to simulate the robot's manipulation process and visualize the hierarchical task decomposition.

12. A team of autonomous robots collaborates to explore and map an unknown environment with complex spatial structures and obstacles. Implement the MAXQ framework to decompose the exploration and mapping task into hierarchically organized subtasks and learn policies for each level of the hierarchy, enabling the robots to coordinate their exploration efforts and efficiently build a map of the environment. Write a Python program to simulate the robots' exploration and mapping process and evaluate the effectiveness of the MAXQ-based approach.

13. A swarm of autonomous robots collaborates to accomplish a collective task (e.g., collective transport, area coverage) in a dynamic and uncertain environment. Implement meta-learning techniques to enable the swarm robots to adapt their collective behaviors and coordination strategies based on past experiences and performance feedback,

enabling them to achieve the task objectives efficiently under varying conditions. Write a Python program to simulate the swarm robots' interactions and evaluate the effectiveness of the meta-learning-based approach.

14. A team of autonomous drones collaborates to perform a cooperative task (e.g., surveillance, search and rescue) in a dynamic and uncertain environment. Implement multi-agent reinforcement learning algorithms (e.g., MADDPG, COMA) to enable the drones to coordinate their actions and achieve collective objectives efficiently, while considering constraints such as communication limitations and collision avoidance. Write a Python program to simulate the drones' interactions and evaluate the performance of the multi-agent reinforcement learning approach.

15. An autonomous robot interacts with humans in a real-world environment with partial observability and uncertainty. Implement a partially observable Markov decision process (POMDP) framework to model the robot's interaction with humans, enabling it to infer human intentions, adapt its behavior, and make decisions that facilitate effective human-robot collaboration and communication. Write a Python program to simulate the human-robot interaction scenarios and evaluate the effectiveness of the POMDP-based approach.

16. Multiple autonomous robots collaborate to perform tasks in a warehouse automation system. Implement the MAXQ framework to facilitate hierarchical task decomposition and coordination among the robots. Write a Python program to simulate the warehouse operations and analyze the effectiveness of the MAXQ approach in multi-agent coordination.

17. An e-learning platform aims to personalize course recommendations for individual learners based on their learning preferences and performance history. Develop a meta-learning framework to adaptively adjust the recommendation algorithm to each learner's characteristics. Use Python to implement the meta-learning approach and evaluate its effectiveness in improving learning outcomes.

18. A city's transportation department employs multiple traffic management agents to optimize traffic flow and reduce congestion. Implement multi-agent reinforcement learning to enable the agents to coordinate traffic signals, lane assignments, and route guidance strategies. Write a Python program to simulate traffic scenarios and assess the performance of the multi-agent RL approach in improving traffic efficiency.

19. An autonomous robot explores and maps an environment with limited sensor visibility and noisy observations. Implement a partially observable Markov decision process (POMDP) framework to enable the robot to localize itself and build accurate maps despite partial observability. Use Python to simulate the robot's exploration and mapping tasks and evaluate the performance of the POMDP approach.