UseCase1

December 20, 2024

1 Module 12: Reinforcement Learning

1.1 Case Study – 1: Optimal Path for Logistics

Objective The goal is to optimize delivery paths using RL, minimizing fuel costs and delays.

Approach

1. **Graph Representation**:
   - Delivery routes are represented as a **graph**:
     - **Nodes**: Delivery points (e.g., warehouses, customer locations).
     - **Edges**: Routes between delivery points, weighted by **distances** (in km).
2. **Reward Matrix**:
   - The distances between nodes are converted into a **reward matrix ((R))**: R[i][j] = int(100/distance)
   - **Shorter distances yield higher rewards.**

[12]:
```
import numpy as np

# Define distances (in km) as per the provided graph
distances = np.matrix([
    [-1, 25, 20, 50, -1, -1],
    [25, -1, 30, -1, 20, -1],
    [20, 30, -1, -1, 5, 45],
    [50, -1, -1, -1, -1, 70],
    [-1, 20, 5, -1, -1, 10],
    [-1, -1, 45, 70, 10, -1]
])

# Convert distances to rewards using the formula int(100 / distance)
R = np.where(distances > 0, np.int32(100 / distances), -1)

# Initialize Q-matrix
Q = np.matrix(np.zeros(R.shape))

# Parameters
gamma = 0.8  # Discount factor
```

```
alpha = 0.5  # Learning rate
num_episodes = 10000  # Number of training iterations

# Q-learning algorithm
for episode in range(num_episodes):
    state = np.random.randint(0, R.shape[0])  # Random initial state

    while True:
        # Get possible actions for the current state
        actions = np.where(np.array(R[state]).flatten() >= 0)[0]

        # Choose a random action from the possible actions
        action = np.random.choice(actions, 1)[0]

        # Calculate the maximum Q-value for the next state
        next_state = action
        max_next_q = np.max(Q[next_state,])

        # Update Q-value using the Q-learning formula
        Q[state, action] = Q[state, action] + alpha * (
            R[state, action] + gamma * max_next_q - Q[state, action]
        )

        # Move to the next state
        state = next_state

        # End episode if goal state (state 5) is reached
        if state == 5:
            break

# Normalize the Q-matrix
Q_normalized = Q / np.max(Q) * 100

# Display the trained Q-matrix
print("Trained Q-Matrix:")
print(Q_normalized)



def optimal_path_with_constraints(start_state):
    steps = [start_state]
    current_state = start_state
    visited = set()  # Keep track of visited nodes

    while current_state != 5:  # Goal state is 5
        visited.add(current_state)  # Mark the current state as visited
```

```
        # Find the next state with the highest Q-value that is not visited
        q_values = Q[current_state].A1  # Flatten the Q-row
        #print(q_values)
        mask = np.array([i in visited for i in range(len(q_values))])  # Create a boolean mask
        q_values[mask] = -np.inf  # Set visited nodes' Q-values to -inf to avoid revisiting
        #print(q_values)

        # Select the best unvisited state
        next_state = np.argmax(q_values)
        if next_state in visited or q_values[next_state] == -np.inf:
            print("No valid path found. Stuck!")
            return steps  # Return the path so far if stuck

        steps.append(int(next_state))
        current_state = next_state

    return steps




# Find the optimal path starting from node 0
path = optimal_path_with_constraints(0)
print("Optimal Path with Constraints:", path)
```

```
Trained Q-Matrix:
[[  0.   72.   85.   60.4   0.    0. ]
 [ 72.    0.   83.    0.   85.    0. ]
 [ 73.   71.    0.    0.  100.   74. ]
 [ 70.    0.    0.    0.    0.   73. ]
 [  0.   73.  100.    0.    0.   82. ]
 [  0.    0.   82.   59.4  90.    0. ]]
Optimal Path with Constraints: [0, 2, 4, 5]
```

[ ]:

2 Case Study – 1: Optimal Path for Logistics

2.1 Domain: Logistics

2.2 Focus: Optimal Path

2.2.1 Business Challenge/Requirement

BluEx is a leading logistics company in India, known for its efficient delivery of packets to customers. However, van drivers are taking suboptimal delivery paths, resulting in delays and higher fuel costs. As an ML expert, you are tasked with creating a Reinforcement Learning (RL) model to identify

efficient delivery paths and reduce costs.

2.2.2 Key Issues

- Delivery routes have multiple attributes (distances, cost) to optimize.
- Classification of routes could be tricky due to the dynamic nature of logistics.

2.2.3 Considerations

- Reinforcement Learning is complex; a sample flow is expected for this assignment.
- Full-fledged implementation will be completed later by the team.
- No data volume is provided; sample data is hardcoded.

2.2.4 Sample Solution

Objective The goal is to optimize delivery paths using RL, minimizing fuel costs and delays.

Approach

1. **Graph Representation**:
   - Delivery routes are represented as a **graph**:
     - **Nodes**: Delivery points (e.g., warehouses, customer locations).
     - **Edges**: Routes between delivery points, weighted by **distances** (in km).
2. **Reward Matrix**:
   - The distances between nodes are converted into a **reward matrix ((R))**:
     R[i][j] = int(100/distance)
   - **Shorter distances yield higher rewards.**
3. **Reinforcement Learning with Q-Learning**:
   - **Q-learning algorithm** is used to train the model:
     - The **Q-matrix** is updated iteratively to store the best cumulative rewards (lowest costs) for moving between nodes.
4. **Training Process**:
   - The agent explores the graph for 10,000 iterations.
   - Each iteration improves the Q-matrix by learning which routes lead to better rewards.
5. **Extracting the Optimal Path**:
   - After training, the model identifies the **shortest path** by following the highest Q-values from the starting node to the goal node.

2.2.5 Simplified Example

- **Graph**:

  - Nodes: (0, 1, 2, 3, 4, 5) (represent delivery points).

  - Edges: Distances between nodes (e.g., (25) km from node (0) to (1)).
- **Sample Output**:
  - **Optimal Path**: From node (0) to node (5), the shortest path might be ($0 \rightarrow 2 \rightarrow 4 \rightarrow 5$).

2.2.6 Why RL is Relevant

The sample demonstrates how RL solves BluEx's challenge by:
1. Modeling delivery points as nodes and distances as edges.
2. Using rewards to represent shorter distances and fuel savings.
3. Dynamically learning the best routes through Q-learning.

This solution provides a foundational **sample flow** to address BluEx's logistics problem.

[ ]: