COIT20277 Introduction to Artificial Intelligence

Week 4 - Tutorial

- Reinforcement Learning
- Responsible AI

Acknowledgement of Country

I respectfully acknowledge the Traditional Custodians of the land on which we live, work and learn. I pay my respects to the First Nations people and their Elders, past, present and future

CQUniversity AUSTRALIA

BE WHAT YOU WANT TO BE
cqu.edu.au

Reinforcement Learning

- In Part 1 of this tutorial, you will apply Reinforcement Learning, specifically ***Q-Learning***, to solve a path planning problem in Python.

- The objective is to understand how the Q-learning algorithm finds the shortest path from an initial node to the goal in a graph.

- A Python program will be developed step-by-step through this tutorial. Your tasks include:
  - Input and execute the code in Jupyter Notebook.
  - Answer the accompanying questions.

Step 1: Setup

- Import necessary libraries:

```
import numpy as np
```

- In this example, only the numpy package is needed.

Step 2: Define the Graph

- Define the graph as an adjacency matrix.
- Include the rewards for transitioning from one state to another.

```
# Define the graph and rewards
graph = np.array([
    [-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]
])
```

- **Question 1**: What could each value in the graph array represent?

Step 3: Initialize Q-table

- Create a Q-table with dimensions corresponding to the number of states and actions.

```
# Initialize Q-table with zeros
q_table = np.zeros_like(graph, dtype=np.float32)
```

- **Question 2**: What is the shape of the Q-table? How does it relate to the number of states and actions in the graph?

Step 4: Define Hyperparameters

- Set hyperparameters for the Q-learning algorithm.

```
# Hyperparameters
learning_rate = 0.8
discount_factor = 0.95
num_episodes = 1000
```

- **Question 3**: What role do learning rate and discount factor play in Q-learning?

Step 5: Implement Q-Learning Algorithm

- Implement the Q-learning algorithm to update the Q-values.

```
# Q-learning algorithm
for episode in range(num_episodes):
    current_state = np.random.randint(0, graph.shape[0])
    while current_state != 5:  # Goal state
        possible_actions = np.where(graph[current_state] >= 0)[0]
        action = np.random.choice(possible_actions)
        next_state = action
        future_rewards = []
        for next_action in range(graph.shape[0]):
            future_rewards.append(q_table[next_action, :].max())
        q_table[current_state, action] = graph[current_state, action] + \
            learning_rate * (discount_factor * max(future_rewards) - q_table[current_state, action])
        current_state = next_state
```

- **Question 4**: Explain the purpose of each step in the Q-learning algorithm.

Step 6: Extract the Optimal Policy

- Extract the optimal policy from the learned Q-table.

```
# Extract the optimal policy
current_state = 0
optimal_path = [current_state]
while current_state != 5:  # Goal state
    action = np.argmax(q_table[current_state])
    current_state = action
    optimal_path.append(current_state)
```

- **Question 5**: How does the agent use the Q-table to determine the optimal policy?

Step 7: Visualize the Optimal Path

- Visualize the optimal path found by the Q-learning algorithm.

```
# Visualize the optimal path
print("Optimal Path:", optimal_path)
```

- **Question 6**: What does the optimal path represent? How does it relate to the goal of finding the shortest path?

Responsible AI

- In Part 2 of this tutorial, you will practice Responsible AI principles on a simple synthetic dataset involving variables like age and gender, which is common in many analyses in the real-world.

- The objective is to appreciate where biases might occur and how to use descriptive statistics and visualization to understand these biases.

- A Python program will be developed step-by-step through this part. Your tasks include:
  - Input and execute the code in Jupyter Notebook.
  - Answer the accompanying questions.

Step 1: Setup

- Import necessary libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- **Question 1**: Which libraries are imported in the code, and what is their purpose in analyzing the data?

Step 2: Generate data

```
# Generate synthetic dataset with bias
np.random.seed(0)
num_samples = 1000

# Generate gender distribution with bias
gender = np.random.choice(['Male', 'Female'], size=num_samples, p=[0.7, 0.3])

# Generate age distribution with bias
age_range = range(18, 65)
age_probs = [0.01] * (65 - 18)  # Equal probability for each age initially
age_probs = np.array(age_probs) / np.sum(age_probs)  # Normalize
probabilities
age = np.random.choice(age_range, size=num_samples, p=age_probs)

# Create DataFrame
df = pd.DataFrame({'Gender': gender, 'Age': age})
```

Step 3: Display summary statistics

```
# Display summary statistics
print("Summary Statistics:")
print(df.describe())
```

- **Question 3**: What information do the summary statistics provide about the data?

Step 4: Visualize distributions

```
# Visualize distributions
plt.figure(figsize=(10, 5))

# Plot gender distribution
plt.subplot(1, 2, 1)
df['Gender'].value_counts().plot(kind='bar', color=['blue', 'pink'])
plt.title('Gender Distribution')

# Plot age distribution
plt.subplot(1, 2, 2)
df['Age'].plot(kind='hist', bins=20, color='green', edgecolor='black')
plt.title('Age Distribution')

plt.tight_layout()
plt.show()
```

Additional Questions to Consider

- **Question 5**: Why is it important to be aware of biases in data generation and analysis?
- **Question 6:** How can we mitigate bias in data analysis?
- **Question 7:** What are other ways to visualize data distributions?

THANK YOU

TIME FOR DISCUSSION & QUESTIONS