21AIE311 Reinforcement Learning

Project Report

Finding the Shortest Path using Q-Learning Algorithm

Group-5

Group Members:

Addanki Veerababu – AM.EN. U4AIE20004

Durgapu Sathvik – AM.EN. U4AIE20024

Kolla Dorasanaiah – AM.EN. U4AIE20041

Abstract:

The report "Finding Shortest Path using Q-Learning Algorithm" discusses the application of the Q-learning algorithm, a model-free reinforcement learning algorithm, to find the shortest path between two vertices in an undirected graph. This report begins by introducing the basics of Q-learning and its application to finding the shortest path in a graph. An example is provided to demonstrate the use of Q-learning in finding the shortest path between two vertices in a graph. The report also discusses the advantages and disadvantages of using Q-learning for this purpose. The report states that Q-learning is a powerful tool for solving various problems, including finding the shortest path in a graph.

Introduction:

A graph is a mathematical structure used to represent relationships between objects. It consists of vertices (also called nodes) and edges that connect pairs of vertices. An undirected graph is a type of graph in which the edges have no direction, meaning that the relationship between the connected vertices is bidirectional.

Q-learning is a model-free reinforcement learning algorithm used to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations. The agent uses a Q-table to take the best possible action based on the expected reward for each state in the environment. A Q-table is a data structure of sets of actions and states, and we use the Q-learning algorithm to update the values in the table.

Q-learning algorithm involves an agent, a set of states and a set of actions per state. It uses Q-values and randomness at some rate to decide which action to take. Q-values are initialized and updated according to the reward and possible future rewards of an action taken.

Mathematically,

Q values can be calculated using the equation:

$Q(s,a) <= (1-\alpha)Q(s,a) + \alpha(R(s,a)+\gamma \max Q(s',a'))$, where

- Q(s,a) is the Q value of an action a from state s to s'.
- α is the learning rate.
- R(s,a) is the reward of doing action a from state s to s'.
- γ is the discount factor.
- maxQ(s',a') is the max Q value among all possible actions a' in the next state s'.

Advantages of Q-Learning Algorithm:

- The algorithm can learn from experience and improve its performance over time.
- The algorithm can handle problems with stochastic transitions and rewards.
- The algorithm can find the shortest path in any graph, regardless of its size or structure.

Dis-advantages of Q-Learning Algorithm:

- The algorithm can be computationally expensive, especially for large graphs.
- The algorithm may not always find the optimal solution, especially if the graph is not well-connected.
- The algorithm may require a large number of training iterations to converge.

Methodology:

1. Import the necessary libraries i.e., networkx, numpy, matplotlib.
2. Define the graph with the edges between the nodes and draw it using the network library.
3. Initialize the Reward Matrix and Q Matrix.
4. Define a function which takes a starting node and returns the next node. Define the exploration rate for random exploration in the function.
5. Define a function to update the Q-value of the action taken.
6. Define a function to improve Q-values by starting at random nodes and making a walk in the graph.
7. Define a function to find shortest path between the initial node and final node by choosing highest Q-value from Q-Matrix when deciding an action.
8. Return the shortest path between the initial node and final node.

Results:

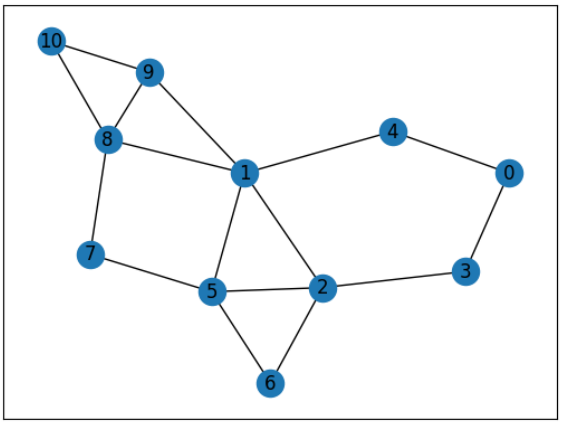

Figure 1: Graph with defined nodes and edges

	0	1	2	3	4	5	6	7	8	9	10
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 2: Reward Matrix

	0	1	2	3	4	5	6	7	8	9	10
0	-100.0	-100.0	-100.0	0.0	0.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0
1	-100.0	-100.0	0.0	-100.0	0.0	0.0	-100.0	-100.0	0.0	0.0	-100.0
2	-100.0	0.0	-100.0	0.0	-100.0	0.0	0.0	-100.0	-100.0	-100.0	-100.0
3	0.0	-100.0	0.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0
4	0.0	0.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0
5	-100.0	0.0	0.0	-100.0	-100.0	-100.0	0.0	0.0	-100.0	-100.0	-100.0
6	-100.0	-100.0	0.0	-100.0	-100.0	0.0	-100.0	-100.0	-100.0	-100.0	-100.0
7	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	-100.0	-100.0	0.0	-100.0	-100.0
8	-100.0	0.0	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	-100.0	0.0	0.0
9	-100.0	0.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	-100.0	0.0
10	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	0.0	-100.0

Figure 3: Q Matrix

	0	1	2	3	4	5	6	7	8	9	10
0	-100.0	-100.0	-100.0	110.0	138.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0
1	-100.0	-100.0	138.0	-100.0	138.0	138.0	-100.0	-100.0	218.0	218.0	-100.0
2	-100.0	174.0	-100.0	110.0	-100.0	138.0	110.0	-100.0	-100.0	-100.0	-100.0
3	110.0	-100.0	138.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0
4	110.0	174.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0
5	-100.0	174.0	138.0	-100.0	-100.0	-100.0	110.0	174.0	-100.0	-100.0	-100.0
6	-100.0	-100.0	138.0	-100.0	-100.0	138.0	-100.0	-100.0	-100.0	-100.0	-100.0
7	-100.0	-100.0	-100.0	-100.0	-100.0	138.0	-100.0	-100.0	218.0	-100.0	-100.0
8	-100.0	174.0	-100.0	-100.0	-100.0	-100.0	-100.0	174.0	-100.0	218.0	274.0
9	-100.0	174.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	218.0	-100.0	274.0
10	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	218.0	218.0	-100.0

Figure 4: Updated Q Matrix

```
shortest_path(0,10)
```

```
[0, 4, 1, 8, 10]
```

Figure 5: Shortest Path between Initial Node & Final Node

Conclusion:

The Q-learning algorithm is a powerful tool that can be used to solve a variety of problems, including finding the shortest path in a graph. The algorithm works by iteratively updating a table of Q-values, where each Q-value represents the expected reward for taking a particular action in a particular state. The algorithm starts by randomly exploring the state space and gradually learns the optimal policy for finding the shortest path.

The Q-learning algorithm is a valuable tool that can be used to find the shortest path in a variety of situations. However, it is important to note that the algorithm can be computationally expensive, especially for large graphs. In addition, the algorithm may not always find the optimal solution, especially if the graph is not well-connected.

Despite these limitations, the Q-learning algorithm is a powerful tool that can be used to solve a variety of problems.