Neuroscience of Learning, Memory and Cognition
Reinforcement Learning Project

December 23, 2023

1 Structure

- This project is a bonus to the final grade. Hence, it must be done individually.
- You must read the instructions carefully and implement the ϵ-greedy algorithm. The introduction to the algorithm and basic concepts are explained later.
- You may run the code multiple times and experiment with various parameters and training episodes. The goal is to achieve the best result provided in later sections. Your work will be graded based on the efficiency of your final answer along with the implementation of the desired concepts.
- The algorithm must be implemented from scratch. Utilization of third-party libraries for the matter is prohibited.
- You must submit your code along with a full report of the experiments carried out and the final plots and figures corresponding to the optimum result.
- The final figures are twofold: **1-** a heat-map of the grid with the best route highlighted **2-** the dynamics of the cumulative reward across all training episodes
- Any brainstorming and discussion with others is allowed and encouraged, but the final work must belong to you and your efforts specifically.

2 Introduction

You have been introduced to some dynamic programming algorithms like Q-Learning to train a model and devise policies based on rewards and punishments. Now, it's time for you to get started and implement what you've learned!

The game is simple! Imagine a **N-by-N** grid which is filled with integers where N represents the number of rows and columns. Each integer represents the amount of reward you'll get if you enter a specific cell. (of course, negative rewards are equivalent to punishments.) Your job is to find the most suitable route in order to get from the bottom-left cell to the top-right one. However, you may only move 1 cell upwards or rightwards at each step. The data regarding the configuration of the grid is provided in the file '**Grid.xslx**'. You can load this file into your code and start digging around!

Suggestion: Since the 30-by-30 grid would be too large to start with, you could implement the algorithm on smaller grids and then try optimizing the routes for the given size.

-5	-5	-5	-5	-5	10
-1	-1	-1	-1	-1	-1
-1	15	-1	-1	5	-1
-1	-1	-1	-10	-1	-1
-5	-5	-5	-5	-5	-5
-10	-1	-1	-1	-1	-1

Table 1: A simpler 6-by-6 grid

3 Concepts

Worry not if you're not familiar with Q-Learning! The algorithm iteratively tries to enhance a policy in order to maximize the reward. Some concepts you might need in the process can be observed below.

- **States :** Each cell in the grid is a state.
- **Actions :** You may choose to move rightwards or upwards at each step. This decision defines your action, which leads to a newer state.
- **Q-Values :** Q-Learning revolves around the concept of Q-Values. The Q-value of a state-action pair represents the expected cumulative reward you can achieve by starting from that state, taking a particular action, and then following a specific policy.
- **Learning rate :** Controls how much the newly-gained information will affect your policy
- **Discount factor:** Controls the importance of future rewards rather than the immediate ones.

4 Briefing

So, you must implement the learning algorithm and improve your policy in order to determine the most rewarding route. The update rule you may use in order to do so is as follows.

$$Q(s,a) = (1-\alpha)\ Q(s,a) + \alpha\Big(R(s) + \gamma\ (max[Q(s^{'},a^{'})])\Big)$$

Where Q represents the Q-Value corresponding to a specific pair of states and actions, α is the learning rate, R represents the immediate reward of a certain state, and γ is the discount factor.

5 Optimum result

In case you need to evaluate your results, the best route is highlighted in the figure below. Following this path will yield a total cumulative reward of **385**.

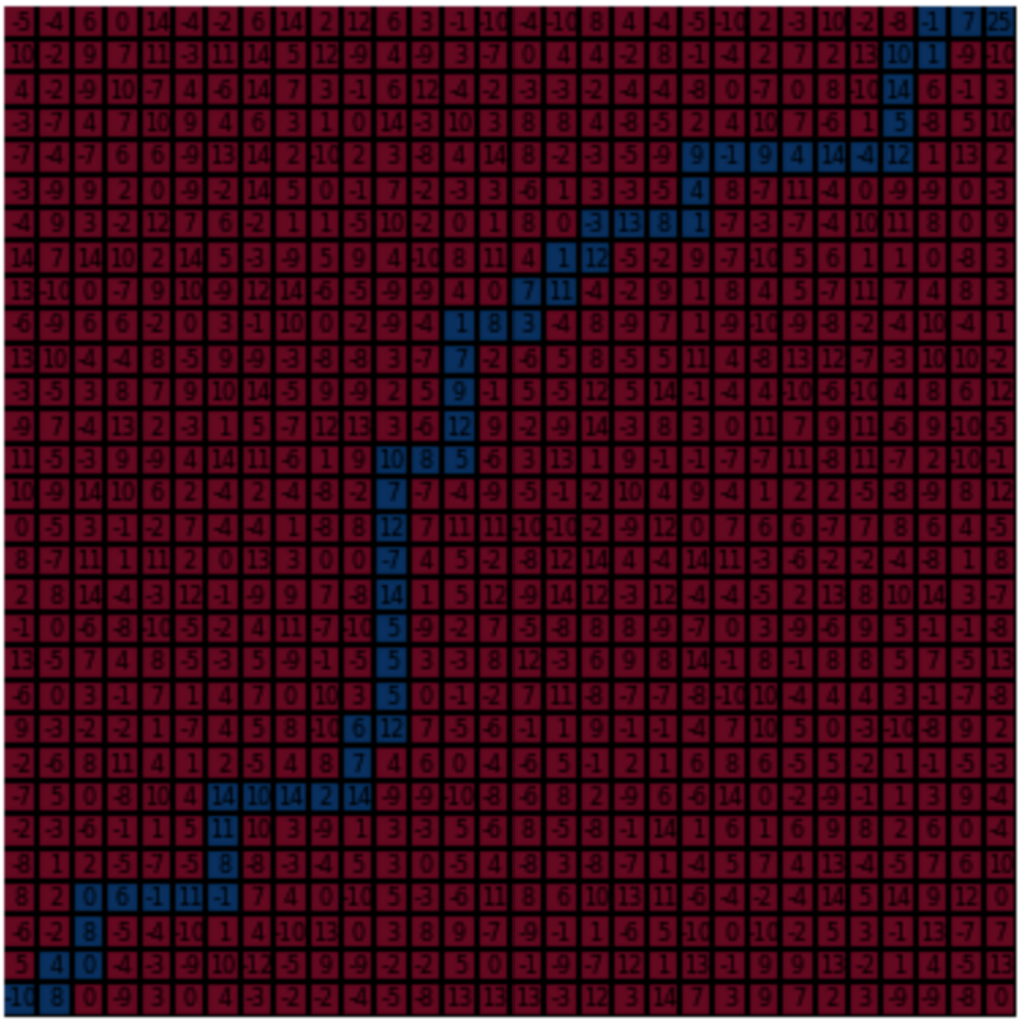

Figure 1. The best possible route highlighted in blue