Neural Modelling exercise 3: Temporal learning and acting

Hand-in by **Friday, 22.11.24 (midnight)** to neuralmodelling24@gmail.com

If you are handing in as a two person team, make sure to put both of your names on your solution (please hand in only one report per team). Besides your responses to the questions in text form, your submission should contain your code. You could e.g. link to a Github repo or submit your report as a Jupyter notebook.

TD Learning (Lecture 4)

- Recreate figure 9.2 (page 15) of the book chapter we provided
- Experiment with following parameters. Plot and briefly describe your observations for each.
  - Reward timing
  - Learning rate
  - Multiple rewards
  - Stochastic rewards

  (Note that you will have to think about how to represent the time between the stimulus and the reward).

Successor learning (Lecture 4)

We will learn the successor representation (discounted future state occupancies) for a simple grid-world under a random walk policy. We will use the grid-world of the shown figure, refer to the Python code we provided for an implementation of this grid.

- Implement a random walk policy (up, down, left, or right with equal probability), filling in the provided function stub. Given a maze, a starting location, and a number of steps, perform the specified number of random moves from the starting location. Make sure to exclude impossible moves (don't just stay at the current spot when such a move is attempted, but pick a different one instead). Use the provided function to plot such a trajectory.
- Write a function which takes a trajectory of grid positions and the current state of your learned successor representation (SR; for this environment it is practical to implement the SR as a matrix, corresponding to the grid). Then, based on the provided trajectory, update the successor representation matrix of the starting state, being sure to discount future states appropriately. Repeat this a number of times with different examples, as

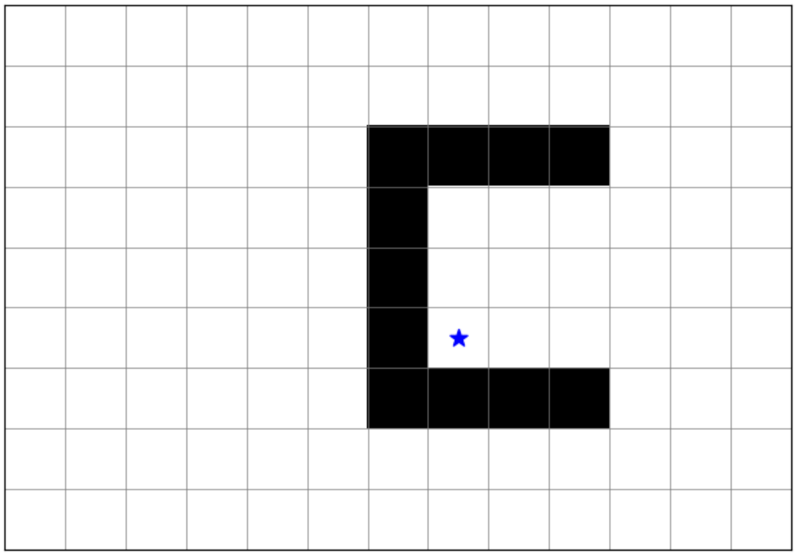

Figure 1: Simple grid-world, the blue star marks the starting location.

outlined in our code, to learn a representation from examples. Plot the learned representation.

- Compute the overall transition matrix, based on the maze layout. Note that this matrix has size 117×117 (as 9*13=117), to represent the probability for transitioning from any state into any other. (Rows which correspond to an impossible state, a wall, contain only zeros, make sure to remove NaNs here).
- Recompute the successor representation at the starting position by repeatedly applying the transition matrix you computed (for this it is opportune to turn the SR matrix into a vector).
- (Bonus) Compute the successor representation one last time, for all states at once, using the analytical equation which involves the transition matrix (in a way that does not require the repeated application of the transition matrix, as you did in the previous exercise). Extract and plot the successor representation of the starting state, as well the representation of the state on the opposite side of the wall from the starting state (2 steps to the left).