Models of Higher Brain Function
PT Project 7: Stochastic Ion Channels

Reinforcement learning model of spatial navigation

Lukas Dippold
Evert de Man

supervised by
Mathias Schmerling

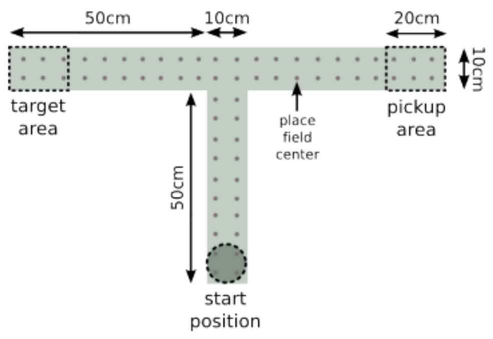
July 31, 2018

1 Introduction

Learning by reinforcement is done by interacting with our environment - we naturally repeat rewarding behavior and avoid behavior that causes a penalty. Reinforcement learning is widely studied in computational science, too. Here, an agent seeks to maximize its numerical reward. For this, the agent must in some cases explore the options it has (exploration) and in other cases keep on taking the action which it deems the most rewarding (exploitation).

1.1 Stick throwing game

For this paper, we simulated a stick growing game for mice traditionally played with dogs.



We let the agent (a rat) explore the above maze, starting in the start position and only receiving a reward of 20 arbitrary units in the target area if the pickup area had been visited beforehand. A penalty was imposed whenever the agent hit one of the walls.

For this, we simulate an input layer containing two populations of place cells with their centers as in Fig 1 indexed $\beta \in \{0, 1\}$. Population 0 is active whenever the pickup area has not been visited yet ($\alpha = 0$) and population 1 is active when $\alpha = 1$. The activity of each place neuron is calculated as

$$r_{j\beta}(s) = \delta_{\alpha\beta} \exp\Big(- \frac{-(x_j - x)^2 + (y_j - y)^2}{2\sigma^2} \Big). \tag{1}$$

This input layer fed to an output layer containing $N_A = 4$ cells, simulating the direction in which the agent could move. With $a \in \{1, 2, 3, 4\}$, each direction was $\frac{2\pi a}{N_A}$, meaning up, down, left and right.

During learning, we use ϵ-greedy action selection. If $\epsilon = 0$, the rat will always choose the action that has the highest expected reward (Q), which is pure exploitation. With nonzero values for ϵ, the animal will choose one of the other Q-values with a chance of ϵ per step. By changing ϵ, we can thus alternate the balance in the exploitation-exploration dilemma.

These expected reward or Q-values were calculated by calculating the activity of one of the N_a output neurons. For this, the animal learned to update the weight matrix w between input and output neurons.

In this paper, we analyze our model with the following calculations. We let 10 rats explore the model and plot how many steps they need to finish the game over trials. Then, we calculate what rats in certain positions would do (e.g. where they would go if they were in a certain position). In addition, we assessed which values for parameters ϵ, η and λ worked best for our model.

2 Analysis of the computer model

2.1 Escape latency

Averaged over 10 rats, the time to successfully complete a trial is plotted as in Figure 1. It takes the rat about 30 trials to fully learn the task.

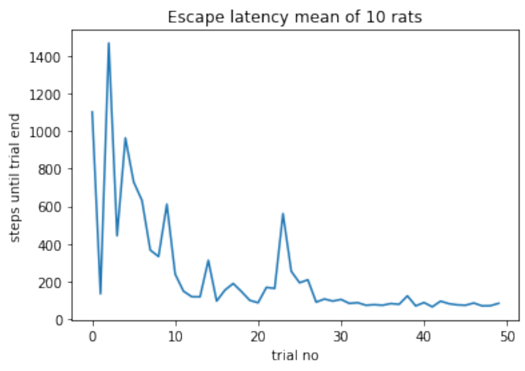


Figure 1: The escape latency of 10 rats in the maze. It is clearly seen the rats take more time to successfully complete the task in the first trial(s), after which the amount of steps saturate. Here, we chose $\eta = 0.01$ and $\lambda = 0.5$.

2.2 Navigation map

[H] When the animal has not trained yet, it needs a lot of steps to reach the end of the trial (Figure 2). After, its path resembles a straight line from the start to the pickup area and onto the finish.

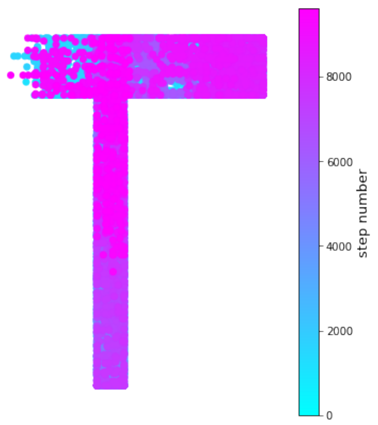


Figure 2: The path of a rat in its first trial.

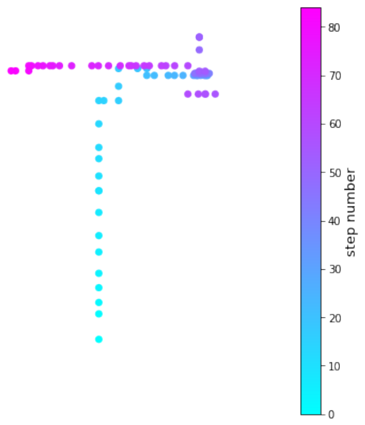


Figure 3: The path of a rat after learning the task.

2.3 Learning curves for different λ and η

Our model updates the weight between the input neurons and output neurons $w_{aj\beta}$ according to the SARSA TD(λ) update rule. We simulated the model with different values for learning rate η, namely 0.001, 0.0005 and 0.0001. In addition, we simulated the model with different values for the decay rate λ. This decay rate decided how much of the memory in the eligibility trace is remembered during updates of the weight.

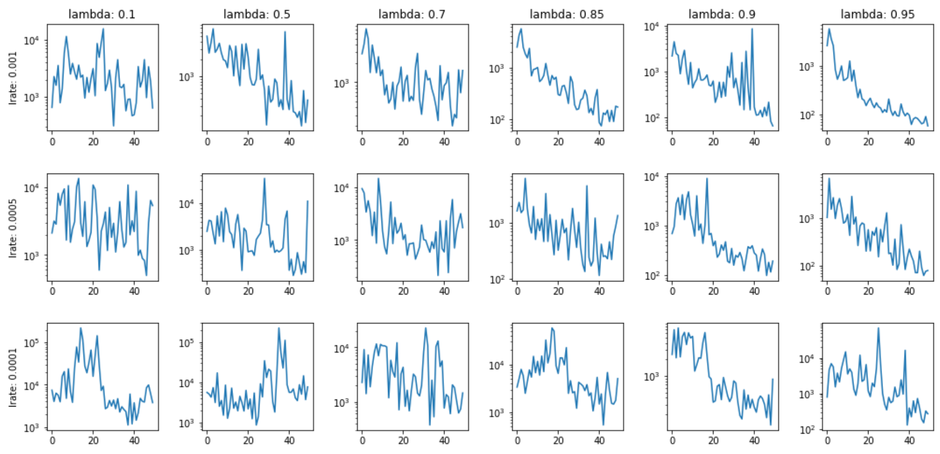


Figure 4: The escape latency for one rat with different values for η and λ. Each rat did 50 trials.

We observe that high λ-values produce a much better learning curve. The eligibility trace is the memory of past actions and adjusting how much the weights should be updated according to their position. The decay rate λ is regarding the agents memory. A high λ means that the agent will only slowly forget the past actions and states whereas a low λ means the agent is really forgetful.

2.4 The influence of parameter ϵ

In a further simulation, we assessed the influence of the parameter ϵ, or the greediness of the rat. A value of 0 means absolute greediness, and means in computational terms the rat will always choose the action in its position that yields the highest expected reward $Q(s,a)$. As the rat learns, it is desirable it uses an exploration strategy first (higher ϵ) and is more greedy after (low ϵ). We simulated the following situations of ϵ-decay.

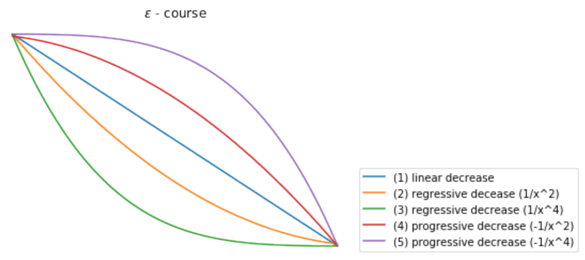


Figure 5: Varying decay curves for ϵ from 0.999 to 0.1.

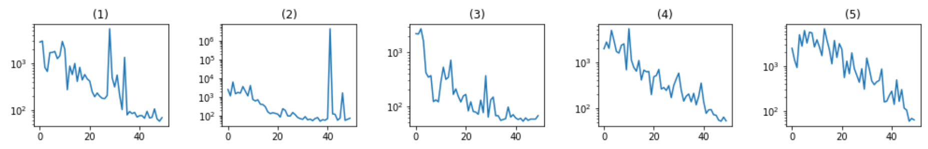


Figure 6: Varying learning curves corresponding to the different decay curves of Figure 5.

Figure 5 shows the 5 different scenarios of the time course of the exploration/exploitation parameter. The linear decrease (1) is the standard case in our model. in this exercise we look at different options to handle the exploration/exploitation dilemma. Scenarios (2) & (3) are quicker switching to exploitation whereas (4) & (5) have a longer emphasis on exploration. The decreasing learning curves of the different options in Figure 6 show all rats are learning the task. However, (4) & (5) are producing a nicer and smoother learing curve. Outlier trials with a lot of steps are rare. (2) & (3) are quicker converging towards $> 10^2$ steps but we observe more outlier trials. For detailed results more testing averaged over more rats is necessary like that what we did in bullet point one. But because of the computational load we cannot provide this for this task.

2.5 The number of action cells N_a

In this task we increase the number of possible directions the rat can do. In the standard case there are 4 different actions: up, right, down, left. When the number of actions is increasing, the directions are changing depending on the equation $\frac{2\pi a}{N_A}$ for each a. We compare the results of 50 trials with 4,5,6,7 or 8 possible actions.

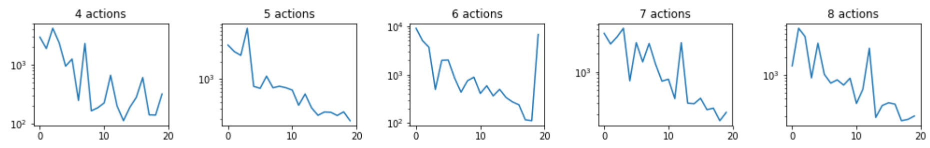


Figure 7: Learning rate for one rat with different possible actions (directions).

More possible actions (higher N_a) lead to more steps when the rat is untrained. That makes sense for our task because only up and right are good actions for the task. In another environment this could be different. But when it comes to training, the differences are not too great, as the 5 training graphs show for the learning curves look more or less similar.

In addition, we plotted the path of an exploring rat that has the option to move in 6 directions ($N_a = 6$) as shown in Figure 9.

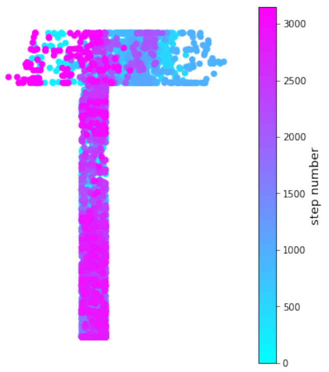


Figure 8: Path of an exploring rat that has the option to take steps in 6 directions. It is clearly visible the steps are not in the up-down-left-right configuration as shown in Figure 2.

2.6 Additional notes

In the end we will have a look at a mistake we did within the process of building our model. It is a common observation in machine learing that you build a model and the model is producing nice results although the algorithm was implemented wronly. In our case we used one eligibility matrix instead of two that would be corresponding to the 2 population of neurons. The first population is guiding the rat to the pickup area and the second population is learning the way from the pickup area to the goal. It does not make sense to mix this up, but as the graph below shows, it is also working and the rat is learning the task.

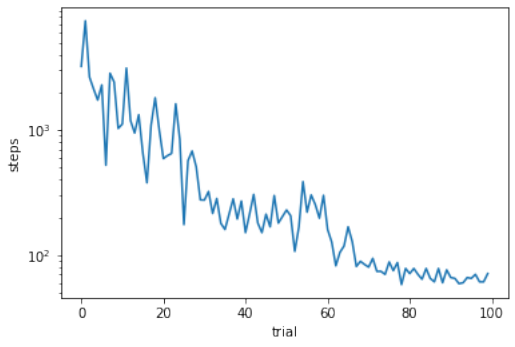


Figure 9: Learning curve with 100 trials. The rat is learning the task also with one eligibility trace matrix.