Neural modeling

Action learning

Arne Gittel

Prepared for, 13.12.2023

Contents

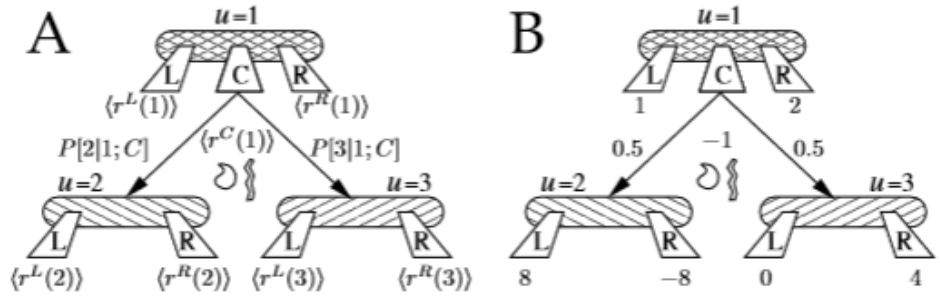

Figure 1: Schematic overview of the problem from the book: A sequential decision problem. The monkey starts at the top u= 1, and can choose to press L or R for a chance of reward, or press C to incur a reward or a shock and then being offered a chance at state u = 2 or u = 3 to get a reward or a shock by choosing L or R there. (A) defines the notation for the rewards and transitions. (B) shows an example in which the best choices are (C,L,R) at states (1,2,3). Here, all the rewards and costs are binary ±1, and the numbers under the levers indicate the probabilities of achieving those binary outcomes. Positive and negative utilities are assumed to add.

1 Introduction

First we want to learn the general reward (r) that is expected from the state. Here v(u) denotes the predicted reward in any given state u. The probabilities to change from any given state to the next are uniform distributed at first and do not change at first so at first the prediction of the reward is ideally the mean of the complete reward that is achievable from this state forward:

$$v(2) = \frac{1}{2}(r_l(2) - r_r(2)) = 0 \tag{1}$$

$$v(3) = \frac{1}{2}(r_l(3) + r_r(3)) = 2 \tag{2}$$

$$v(1) = \frac{1}{3}\left(r_l(1) + r_r(1) + \left(r_c(1) + \frac{1}{2}(v(2) + v(3))\right)\right) = 1 \tag{3}$$

Here we can use the critical learning rule to update our knowledge about predicted reward for each state. When action a at location u leads to reward $r_a(u)$ and changes location to u', the temporal difference rule modifies the prediction w(u) with the learning grate ϵ and the prediction error. Most importantly, our prediction error δ is calculated with:

$$\delta = r_a(u) + v(u') - v(u) \tag{4}$$

$$v(u) \longleftarrow v(u) + \delta * \epsilon \tag{5}$$

2 Implementing the evaluation of the states

I used a graph representation of the problem where each state is a node and each change in state is an edge with a given probability and reward. I used two additional states D to represent the decision point that leads to states u2 and u3 and and state T to represent the termination of the task (Fig. 2):

This class also contains a step function to change nodes based on the probability of the outgoing edges and return the reward and the chosen edge. In the first task all probabilities of the edges stay the same and I created a policy agent class that tracks and updates the predicted rewards for all states u. In the end the resulting predictions for each state are shown in Fig. 3. It is to note that in each trial the agents to not seem to converge towards the calculated predictions for each state but the mean over many repetitions converge. The learning rate was set to 0.2.

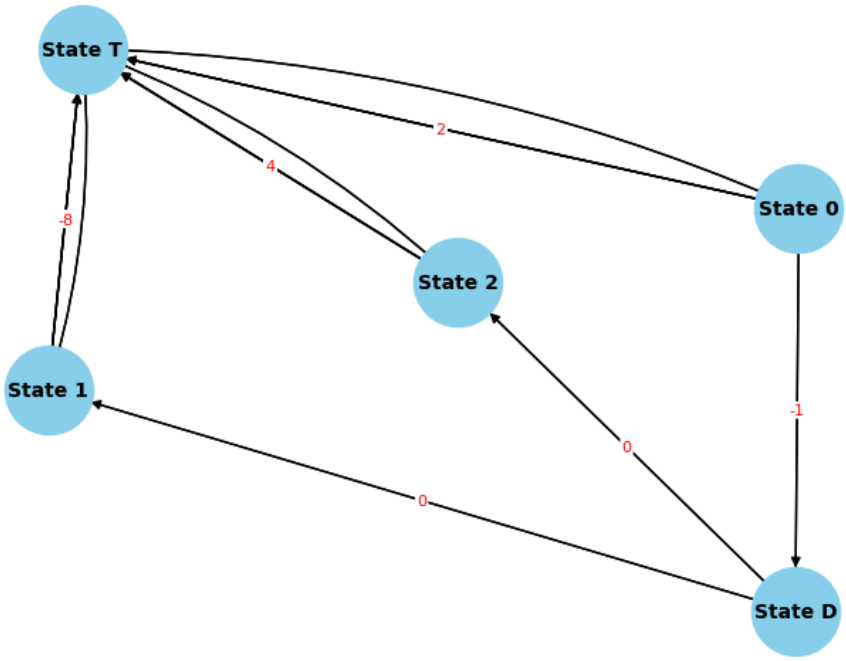

Figure 2: Schematic overview of the graph (even tho it is not shown, all edges are directed and have the correct reward)

3 Implementing action based learning

In the next step we want to update our probabilities for each edge depending on the predicted reward of each action. Thus we also implement the actor learning rule with another actor learning rate for all possible edges b from a node a. With the softmax function of each m we can then reevaluate the probabilities in each edge after we have taken it.

$$m_b(u) \longrightarrow (1 - \text{decay}) m_b(u) + \epsilon_a * \delta \tag{6}$$

In my code i used a class for the actor critic that tracks and updates both m and v via the update actor function and updates the probabilities in the edges via the update

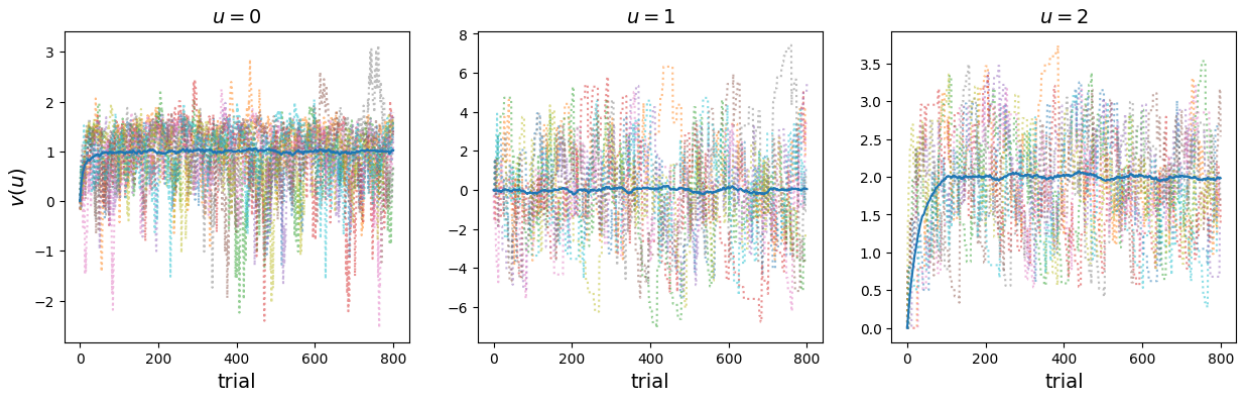

Figure 3: Learning of state rewards

step function. The resulting critic actor can learn the correct choice in each node over the course of many trials and repetitions as shown in Fig. 4. After 1000 trials not all curves have converged and it takes on average around 800 trials for the actor critic to learn that the small punishment in the first choice leads to possible greater reward later on. Still the predictions of each state do not seem to converge for all repetitions. For the parameters I used 0 decay and $\beta = 1$ for the softmax function. The learning rate for my critic was 0.2 and therefore higher than the actor learning rate 0.075.

4 Maze task

The actor critique can also solve the maze tasks that we had in the lecture Fig. 5

Simply by correcting the edges and nodes and changing my dictionary for m to account for the two choices of State 0 we can learn the maze, see Fig. 6:

Which results in the learning curves shown in Fig. 7 (hyper parameters were not changed). The actor critique quickly learns how to get to the cheese and therefore the predicted reward of the initial state is the same as the maximal reward (4).

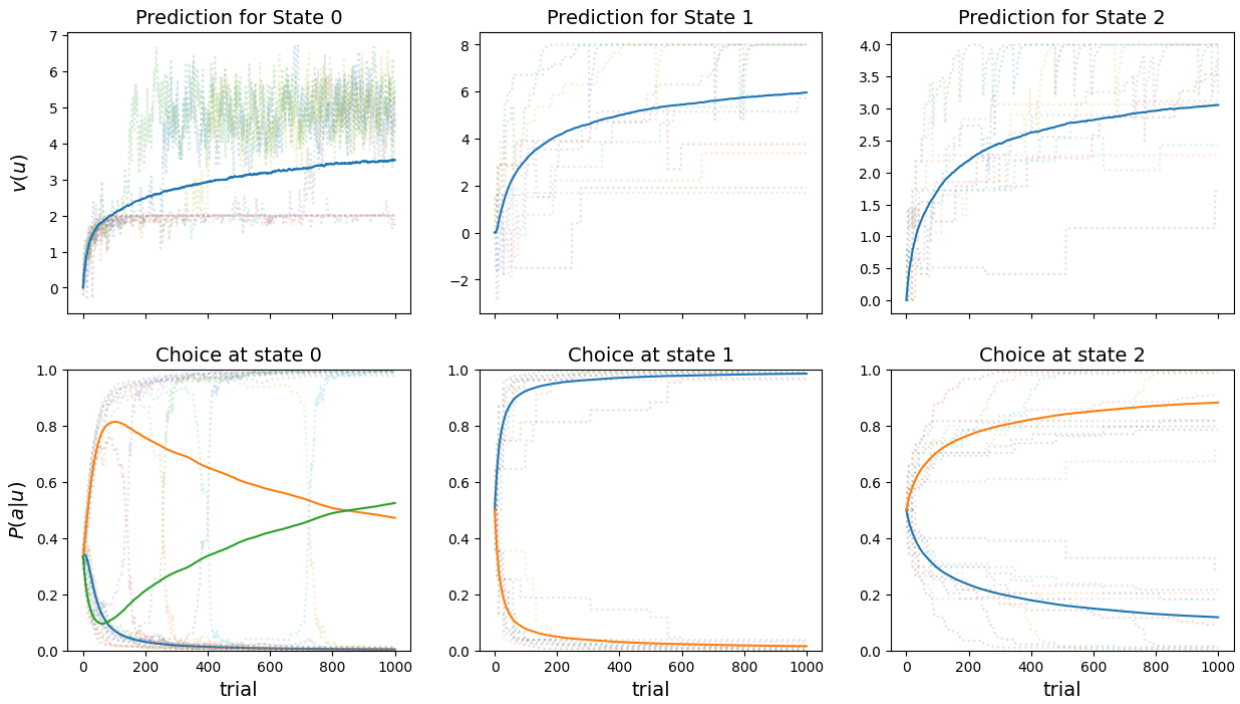

Figure 4: Graphs of the predicted reward values in each state as well as the choice probabilities in outgoing edge. With the mean over all repetitions and 20 sample trials

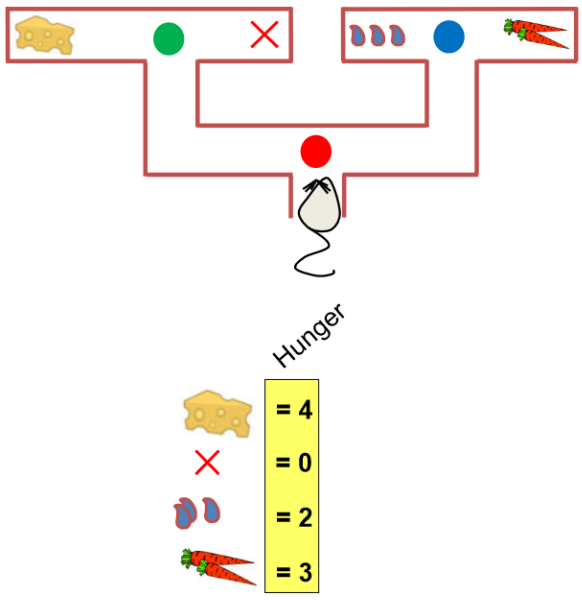

Figure 5: Maze task from lecture

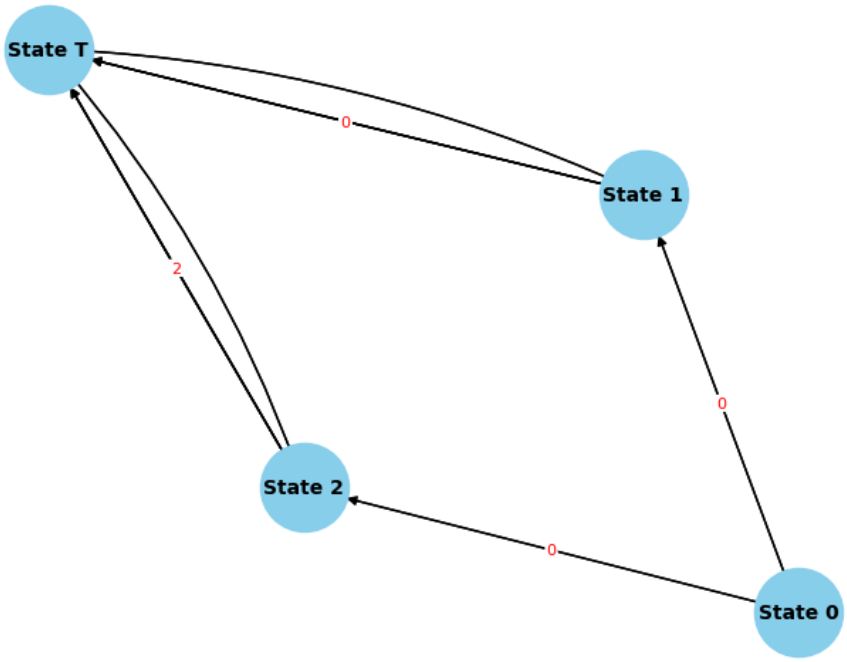

Figure 6: Graph for maze task (missing rewards 3 from state 2 and 4 from state 1)

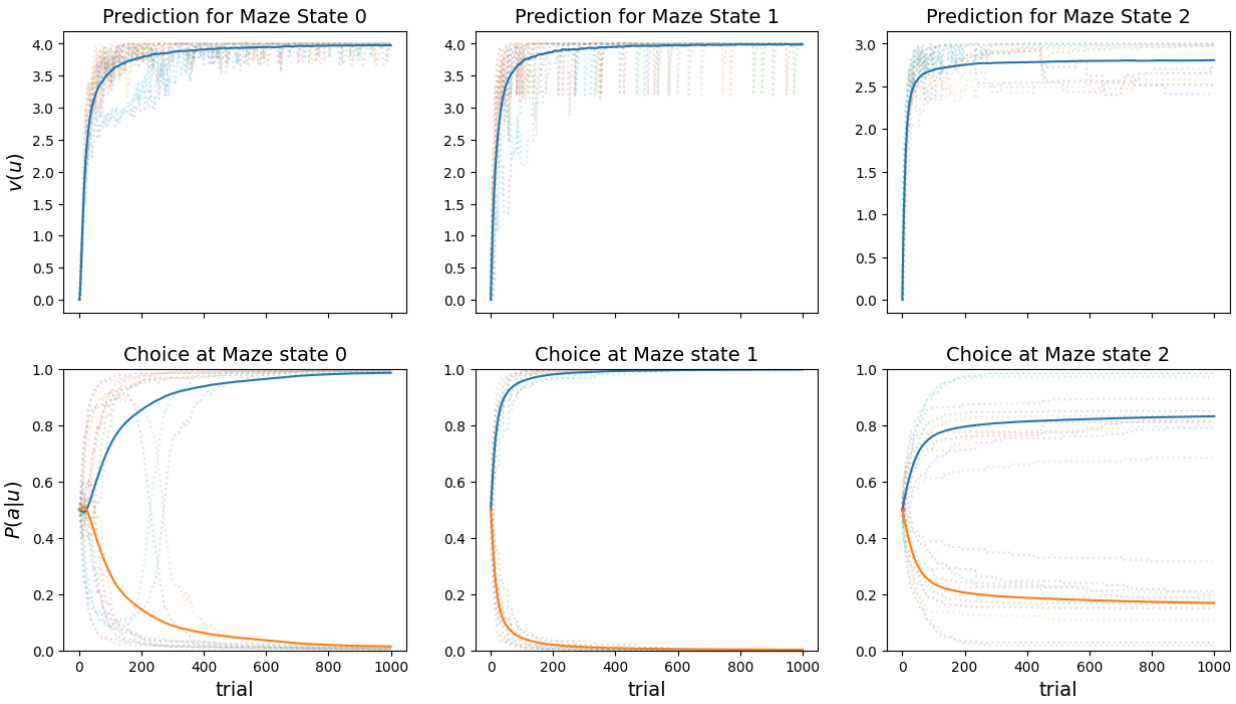

Figure 7: Learning in the maze task