A network model for action learning built upon Endotaxis

Zeyu Jing[a]

[a]*Computation & Neural Systems, California Institute of Technology, Pasadena, CA 91125*

Abstract

Mice might employ other methods than trial-and-errors in the later stage of navigation in maze. In this project, we built a three layer neural network—two feedforward layers with Hebbian plasticity and one middle recurrent layer with anti-Hebbian plasticity–to explain the late stage navigation behavior, which we termed *action learning*. We tested the model on three different environments: grid world, Hanoi tower, and binary maze. Overall, the network was able to implement state-action association for most possible states after training.

1. Introduction

The Meister Lab's recent behavioral studies of spatial navigations shows that mice are able to navigate toward target locations (e.g., a water port or home location) from anywhere in the maze in an accurate and confident way at their late stage of learning. This observation suggests that "expert mice" might employ other methods than trial-and-errors, as proposed in the Endotaxis model.

We conjecture that the mouse consolidated its memory of specific actions taken during successful bouts in the form of associative memory, thereby waived the process of trial-and-errors. Namely, during early successful forays toward the targets, the mouse memorized the actions it had taken so that it would be able to initiate the same action next time it encountered the same situation. Note that this type of associative learning differs from classical-conditioning-type associative learning by its contextual nature. Namely, a given spatial stimulus does not uniquely prescribe an action that should be taken. Instead, the correct action depends on the animal's goal, which could

be classified as a type of internal contextual signal. We call this method of spatial navigation *action learning*.

As mentioned above, the prerequisite for the animal to complete such action learning is to have a number of successful forays toward the targets. The Endotaxis provides such successful forays by implementing trial-and-errors with its (virtual) odor gradient ascent algorithm. It is due to this reason that our model is considered as the next learning stage of the Endotaxis model.

2. Model description and methods

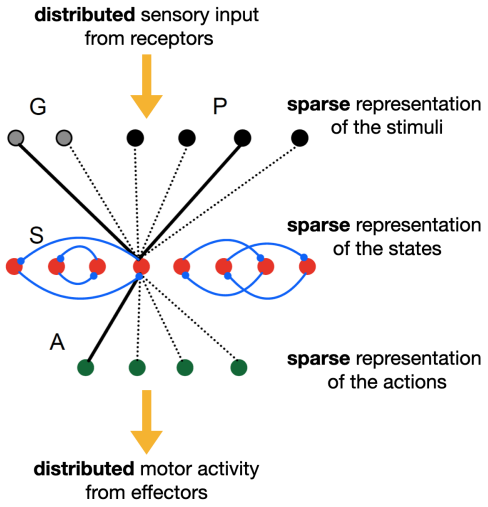

Figure 1: Model architecture.

2.1. Architecture

We formulate the action learning problem mathematically as follows: let G be the space of goals, P be the space of locations in the maze, and A be the space of actions, action learning lears the mapping

$$f : G \times P \to A$$

such that $f(g, p) = a$.

How to implement this mapping using biologically realistic neural networks? Observe that each element (g, p) from the product space $G \times P$

completely characterizes an agent's *state* as far as the navigation task is concerned. Therefore, we can include an unsupervised layer of state neurons, each of which learns a particular (g, p) pair, and an additional supervised layer of action neurons that learns the state-action mapping (Figure 1).

2.2. Neurons

The model neurons are all binary, i.e., they are either in on-state or off-state at each time step. The activity of the goal neurons and location neurons depends on the agent's current goal and location, i.e., (g, p) while the activity of action neurons depends on the action taken at the current state. Note that in the simulation, the activity of goal/location/action neurons is prescribed by the Endotaxis module rather than dynamic equations, whereas the state neurons' activity is given by the stable state of a non-linear recurrent network specified by a dynamical system.

2.3. Connections

The connections from G or P to S (with matrices SG, SP, respectively) and from S to A (AS) are all-to-all feed-forward. They are trained with Hebbian learning rules. The connections within the S layer is recurrent and trained with anti-Hebbian rules. It is because of the anti-Hebbian dynamics that the state cells are able to recognize and encode all possible (g, p) pairs rather than showing some highly correlated response patterns.

2.4. Equations

The Hebbian learning rules for matrices SG and SP are prescribed by

$$\Delta SG_{ij} = \beta_{SG} \cdot s_i \cdot (g_j - \alpha_{SG} \cdot SG_{ij}) \tag{1}$$

$$\Delta SP_{ij} = \beta_{SP} \cdot s_i \cdot (p_j - \alpha_{SP} \cdot SP_{ij}) \tag{2}$$

where s_i, g_i, p_i are the firing rates (vectors) of state cells, goal cells, location cells, respectively; β_{SG} and β_{SP} determine the overall Hebbian learning rate, while α_{SG} and α_{SP} control the depression rate. Both SG and SP are randomly initialized before learning.

The anti-Hebbian learning rules within the state cell layer is given by

$$\Delta SS_{ij} = -\beta_{SS} \cdot s_i s_j \tag{3}$$

where the matrix SS is initialized to all zeros. Self-inhibition is prohibited throughout the learning process. Note that this form of negative learning

rules effectively discourages simultaneous firing of state cells, thereby encourage state cells to find different statistical regularities from their input patterns.

Importantly, the activity of state cells are computed by the following two steps:

1. Solve for stable fixed point from the dynamical system

$$\frac{ds}{dt} = -s + F(SG \cdot g + SM \cdot m + SS \cdot s) \tag{4}$$

where F is the sigmoid function

$$F(x) = \frac{1}{1 + \exp(-k(x - \theta)} \tag{5}$$

2. Set $s = 1$ if $s > 0.5$; $s = 0$ otherwise.

This system (equation 4) is guaranteed to settle into a stable sate after an initial transient. Note that the above two steps are implemented each time we need to compute the activity of state cells. In other words, we only care about the stable state of the system that gives us the firing rates of state cells, while we disregard the specific dynamic evolution of the system. This is consistent with our choice of binary activity for all the model neurons.

2.5. Training

The training set consists of a sequence of N runs, each of which has a random starting location and a random goal. As mentioned in the introduction, these runs are the output of the Endotaxis model, thus are perfectly accurate. Equivalently, this is to generate N shortest-paths with random start and end. Thus, each sequence looks like

$$p_1 \xrightarrow{a_1} p_2 \xrightarrow{a_2} \dots \xrightarrow{a_{k-1}} p_k$$

where actions a_i connect consecutive locations p_i and p_{i+1}, and p_k is the goal location.

2.6. Learning dynamics

After the agent arrives at location p with goal g, p and g cells fire simultaneously in the input layer. Suppose now a state cell s from fires due to its large synaptic weight with g and p, its connections with cells g and p get

strengthened whereas its connections with other *inactive* cells get weakened. Therefore, next time the agent arrives at the same location g with goal p, this s cell would be more likely to fire because of the previous strengthening; while if the agent arrives at other locations or has a different goal, this particular s cell would be less likely to fire due to the previous depression.

In a less preferred scenario where multiple state cells fire in response to a particular combination of p and g, by developing mutual symmetric inhibition, the state cell(s) with stronger connections with g and p would win the competition and strengthen its encoding of this particular (g, p) pair; while the state cell(s) that lost the competition would seek other patterns in the following training stream.

Remarkably, the anti-Hebbian plasticity implements a form of *competitive learning* such that simultaneous firing of multiple state cells for a given input pair (g, p) is discouraged. This would force different state cells to find different patterns in the input. As a result, the state cell layer's representational capacity increases.

3. Simulation results

We tested our model on three graphs: grid world, Hanoi tower, and a binary maze.

3.1. Grid world

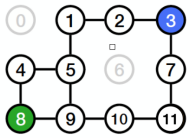

Figure 2: The grid world environment. Actions are defined to be movements in four allocentric directions, i.e., north, east, west, south. Note that location 11 is no longer a target location in our simulation–node 3 and 8 are the only two goals in the simulation.

From figure 3, we can see from column 1 and 2 that the the state cells learned all the combinations of goals and locations. The block-diagonal pattern in column 3 (SS) suggests that state cells encoding the same goal tend to inhibit each other more when compared to state cells encoding different goals, because they are more likely to fire together due to the common goal cell input. The last two columns are simply the result of few-shot action

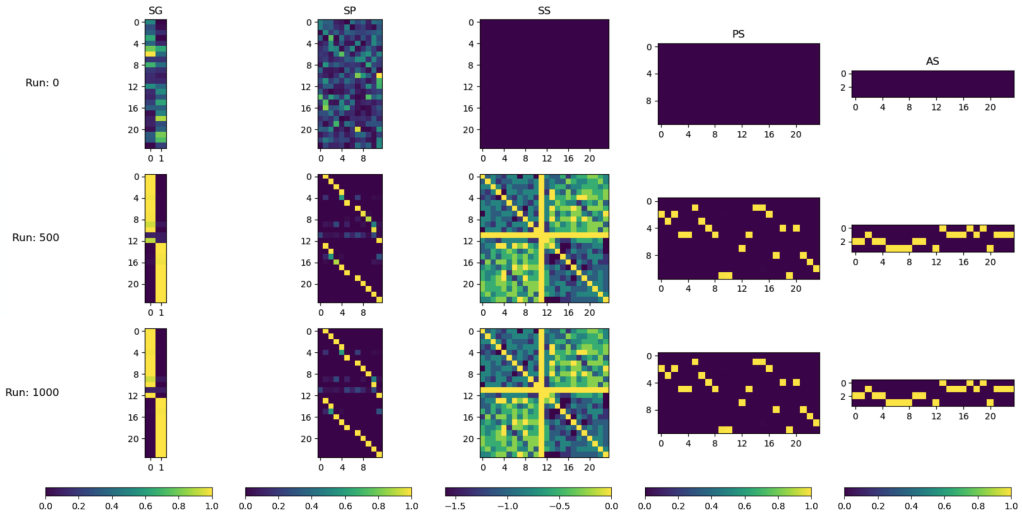

Figure 3: The grid world simulation result. Each row is at a particular run and each column is representing a particular synaptic weight matrix. From left to right columns: SG, SP, SS, PS, AS. Note that the fourth column (PS) is not mentioned in the text because it is equivalent to AS–it represents the next position, which is equivalent to the action.

learning–namely, once state cell representation is learned, learning the action association only needs a few experiences. This is consistent with the observation that sparse representation is preferential for associative learning.

3.2. Hanoi Tower

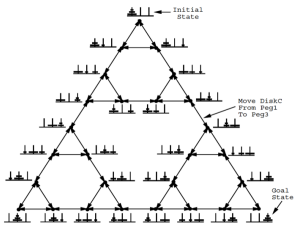

Figure 4: The Hanoi tower graph structure. Actions are defined to be the six disk movements, one from each pole. We chose two of the three vertices as our goals in the simulation. Note that the graph structure is highly symmetric.

From figure 5, we see that the learning result is almost perfect, which is particularly clear from the second column (SP). This is precisely because of the perfect symmetry in the graph structure of the Hanoi tower

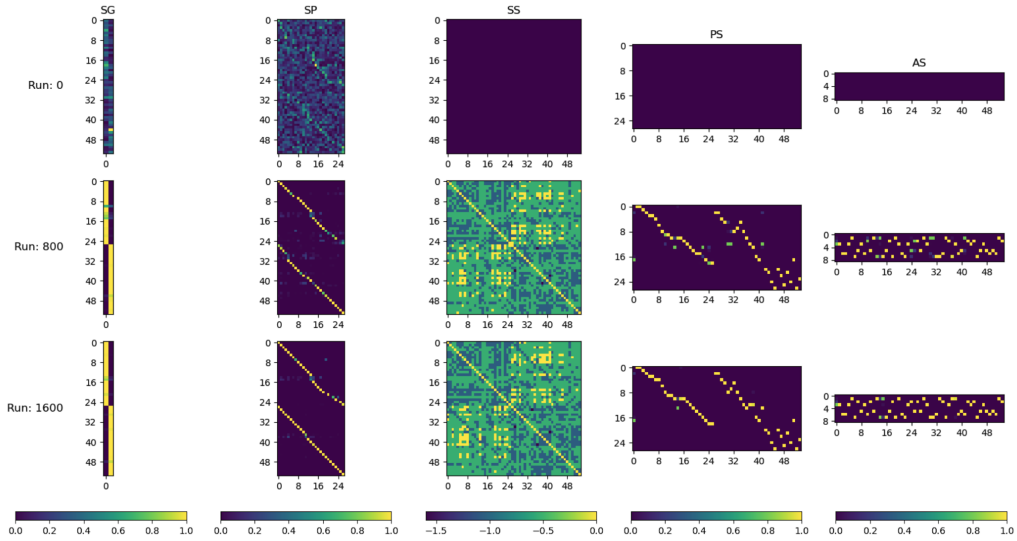

Figure 5: The Hanoi tower simulation results. The arrangement of plot is the same as the grid world. Note the almost perfect development of state cell representation seen from the first two columns.

game (Figure 4). As a reminder, our training sequences are generated using shortest-paths from a randomly selected location in the graph to a random goal location. The distribution of (g, p) affects the learning result. If each combination is visited at the same probability (uniform), the state cell learning result will be perfect, though this also depends on the randomness from the weight initialization process. We will see this in the binary maze case below.

3.3. Binary maze

As we can see from the first two columns of Figure 7, the state cell representation is not as good as before. The reason is that our training runs are now very skewed because of the goals we have chosen. Intuitively, the existence of a leaf node target leads to some extremely long runs, for instance, the one from the bottom leaf node to the blue node in figure 6. Also, there are nodes that are visited way more frequently than others, for instance, the agent has to visit the two nodes at the first level in order to reach the green node from anywhere in the graph. In comparison, any non-goal leaf node only gets visited when it's selected as the starting point.

We predict that in reality, if a location does not involve enough features of the animal's interest (e.g., water, food, etc.) and is at a location that's to

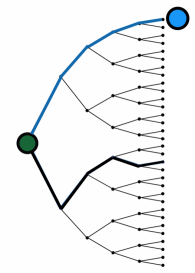

Figure 6: The binary maze graph. Actions are defined in terms of the three tree operations: up to the parent, down to the left child, and down to the right child. Note that the green and blue nodes are the two goals in the simulation. The thick black line is not important here. Also note that the average length of a run from a random starting node to the blue node is bigger than to the green node. This creates the asymmetry in the training sequence, which in turn explains the imperfectness of the state cell learning result.

the periphery of the graph, the animal would have less neural representation of the particular location and also less accurate/confident navigation behavior. It would probably fall back into its trial-and-errors strategies, i.e., the Endotaxis. On the contrary, if a node is very frequently visited (supposed in the center of the environment), its neural representation should be dense and strong, and the action learning is also more effective there.

To test our hypothesis that the distribution of (g, p) pairs affects the model's performance in state learning (thus also state-action mapping), we uniformly generate a sequence of unit length runs, namely, a sequence of (g, p) pairs, without any action–we simply look at the state learning. From figure 8, we see from the last row and first two columns that the state cell learning is almost perfect. This supports our statement that the reason figure 7 has learning defects is that the distribution of (g, p) pairs is skewed.

4. Discussion

We implemented an action learning model that was built upon the Endotaxis to explain the late stage navigation behavior of the mice. After learning, most state cells encode a particular pairing of goal and location. In the meantime, each encoding state cell is associated with an action that was previously given by the Endotaxis model as a result of odor gradient ascent. In summary, the model implements the state action mapping so that later when the agent encounters the same state, the corresponding state neuron will fire and activate the corresponding action, without trial-and-erros.

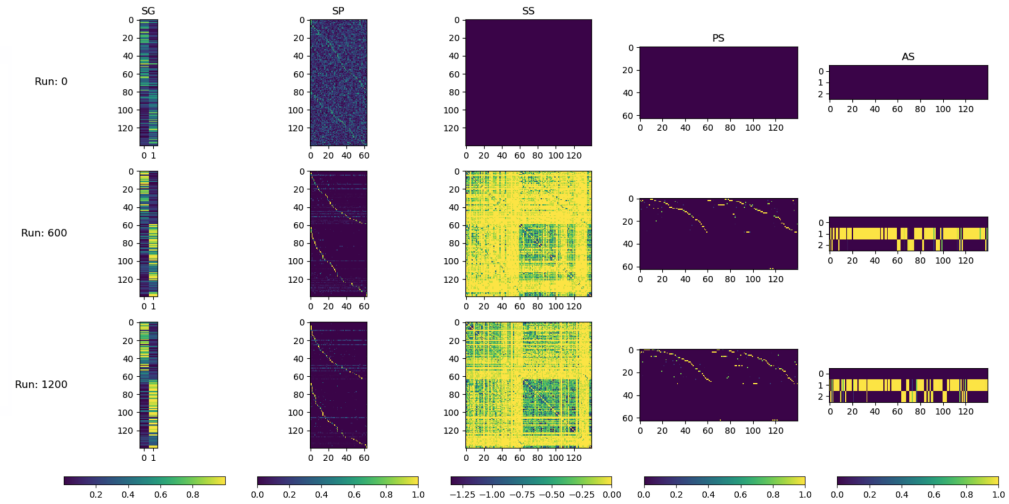

Figure 7: The binary maze with 5 levels/63 nodes simulation results. The arrangement of columns and rows is the same as above. Note that now the first two columns are suggesting the imperfection of the learning, i.e., there are multiple state cells responding to the same combination of goal and location, and there are also some combinations that don't have any state cells associated with.

The most important ingredient of the model is the anti-Hebbian plasticity rule. It provides an effective measure to decorrelate weight vector (with input cells) of state cells in our model, which enables to them learn all the input patterns. Although we have obtained a qualitative understanding of the anti-Hebbian learning rules, a comprehensive mathmatical analysis of the rules is not beyond reach and should be done in the future.

Similarly, the statistical elements such as the random initialization and input distribution should also be made mathematically rigorous in order to obtain a full understanding of the computational capability of the model.

References

Please see the accompanying slides.

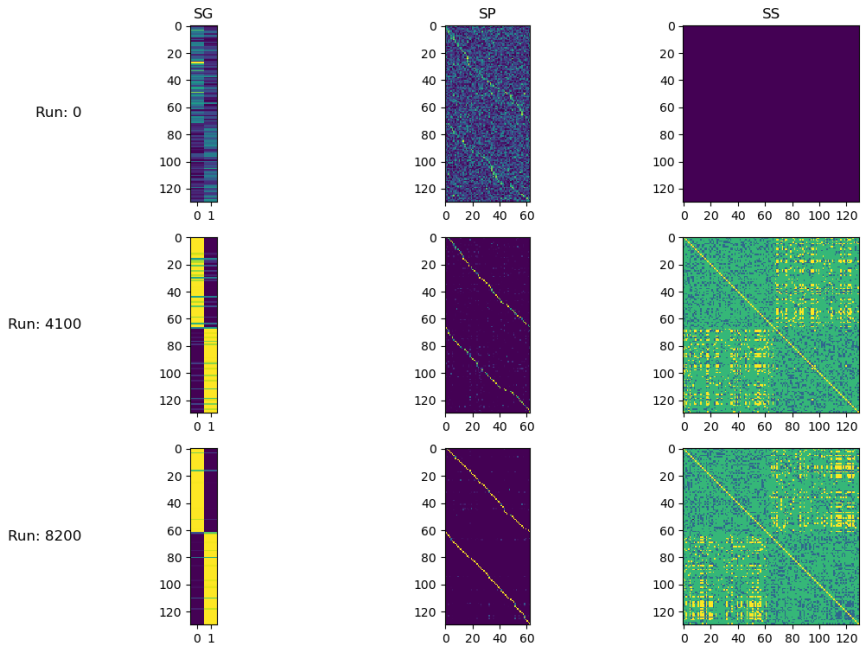

Figure 8: The binary maze with 5 levels/63 nodes simulation results with uniform training runs. The number of runs (now length-1) on the left was chosen so that the total number of node visits are the same as figure 7. Here we have only three columns–the first three columns from previous figures. As the last row indicates, the state representation is now perfect, as in the Hanoi tower or the simple grid world environments.