Project 2: Learning of Grid Cells

Claus Lang, Can Eren Sezener, Claudia Winklmayr

09.02.2016

Abstract: Grid cells, located in rats medial entorhinal cortex (mEC), show increased firing activity when the animal enters specific regions of the environment. The resulting firing maps show a characteristic hexagonal grid pattern. In our project we implement a computational model for the development of grid cells, introduced by Kropff and Treves. We simulate a rat exploring a square environment at constant speed. The x and y coordinates of its loactions serve as input for a layer of 400 place units. The output layer then recieves a weighted sum of the place cells activity to compute the activation of 100 grid cells.

1 Introduction

There are various types of neurons that serve to encode an animals spacial location. Place cells - located in hippocampus - were discovered in the 1970s by O'Keefe and Dostrovsky. Place cells show increased activity when the animal enters a specific region of the environment - the so called place field.
Grid cells - located in medial entorhinal cortex (mEC) were then discovered in 2005. In contrast to place cells they are activated at various locations and their firing maps show a characteristic hexagonal grid pattern. Different grid cells can be distinguished by their spacing, orientation and spacial phases. Neighboring grid cells usually show similar and orientation.
It is assumed that grid cells perform a path integration task, taking into account the rats position, speed and direction. Visual information on the other hand plays at most a secondary role in the formation of the grid pattern.
The Kropff and Treves model assumes places cells as the basis for grid cell development. The place cells take as input the x and y coordinates of a virtual rat exploring a square environment of 125cm length. Every output unit then receives input from all place cells and the weights connecting input and output neurons are updated by a Hebbian learning rule.

2 Model

The model consists of a layer of 400 input neurons (simulating place cells) and 100 ouput neurons (simulation grid cells), every unit of the input layer being connected to every unit of the output layer. By w_{ij} we denote the weight of the connection between the j-th input unit and the i-th output unit.

model_overview.png

2.1 Spacial Information

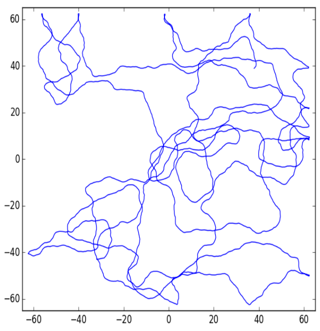


Figure 1: trajectory of the virtual rat exploring for 50 seconds

To create the spacial coordinates that serve as input for the first layer of neurons, we simulate a rat exploring a square environment of size 125cm $\times$ 125cm at a speed of 0.4 m/s. At each timestep τ =10ms we chose a new direction by randomly drawing an angle ϕ from a normal distribution, which is centered around the direction at the prvious timestep and has a standard deviation of 0.2rad.

2.2 Input layer

The input layer consists of 400 place units processing information about the rats location $\boldsymbol{x}(t)$ at a given time t. The firing $r_j^{in}(t)$ of each place unit is modeled by an exponential function centered around the cells prefered location $\boldsymbol{x}_j$. The respective fireing field's width is $\sigma_p = 5$cm.

$$r_j^{in}(t) = \exp\left(\frac{-|\boldsymbol{x}^t - \boldsymbol{x}_j|}{2\sigma_p^2}\right) \tag{1}$$

Figure 2: From left to right: (i) distribution of place fields across the environment, (ii) trajectory of the virtual rat and (iii) corresponding activity of place cells.

2.3 Output layer

At each timestep every output unit recieves as input a weighted sum h of the activations of all place cells. For the i-th output unit this takes the form:

$$h_i(t) = \sum_j w_{ij} r_j^{in}(t). \tag{2}$$

This input is then subject to an dynamic adaptaion process, during which we also determine parameters g and μ which controll the mean firing activity across neuons as well as the sparsity. To calculate the final output $r_i^{out}(t)$, we then use a nonlinear activation function:

$$r_i^{out}(t) = f(r^+(h_i(t)); g, \mu). \tag{3}$$

The crucial point for the developement of grid cells is that we do not directly use the value $h_i(t)$ to calculate the output laver activity but first subject it to the following adaptation process:

$$\begin{aligned} \tau^+ \frac{d}{dt} r^+ &= h_i(t) - r^+(t) - r^-(t) && (4) \\ \tau^- \frac{d}{dt} r^- &= h_i(t) - r^-(t) && (5) \end{aligned}$$

The control parameters g and μ are determined by itereation. g serves to keep the mean activity across neurons - defined via $a = \frac{1}{400}\sum_i r_i^{out}$- within a 10%-range from the value a_0. The parameter μ respectively controls the sparsity $s = \frac{(\sum_i r_i^{out})^2}{\frac{1}{400}\sum_i (r_i^{out})^2}$ and keeps it within a 10% error bound of the value s_0. At each iteration step g and μ are updated in the following way:

$$\begin{aligned} \mu^{t,l+1} &= \mu^{t,l} + b_\mu(a^l - a_0) && (6) \\ g^{t,l+1} &= g^{t,l} + b_g g^{t,l}(s^l - s_0) && (7) \end{aligned}$$

l indicates the step of the iteration. The updating process is stopped if both a and s fall within the 10 % error bound of a_0 and s_0. In any case the updating process stops after 100 iterations.

output_function.png

Figure 3: Activation function for output neurons

a_0	s_0	$\mu^{0,0}$	$g^{0,0}$	b_μ	b_g	τ^+	τ^-
0.1	0.3	0	4.5	0.01	0.1	0.1	0.3

Figure 4: Default parameters for adaptation dynamics

Using the above introduced expressions we can now write down the activation function in its complete form:

$$r_i^{out}(t) = \frac{2}{\pi}\arctan(g(r_i^+(t) - \mu))\Theta(r_i^+(t) - \mu) \tag{8}$$

The factor $\frac{2}{\pi}$ serves to normalize the output to a value between 0 and 1 while the Heaviside step-function Θ makes sure that we only recieve positive output-values.

2.4 Weights

The weights connecting input and output layer are initially being drawn from a uniform distribution in $[0, 1]$ and normalized to unity i.e. for every output unit i we impose $\sum_j w_{ij} = 1$. At each timestep the weights are then updated using a Hebbian learning rule:

$$w_{ij}(t + \Delta t) = w_{ij}(t) + \epsilon(r_i^{out}(t) r_j^{in}(t) - \overline{r}_i^{out}(t) \overline{r}_j^{in}(t)) \tag{9}$$

here $\epsilon = 0.005$ is a moderate learning rate and $\overline{r}_i^{out}(t)$ and $\overline{r}_j^{in}(t)$ are the running averages of input and output firing rates. Using a time averaging factor $\eta = 0.05$ we calcualte these averages via:

$$\overline{r}_i^{out}(t) = \overline{r}_i^{out}(t-1) + \eta(r_i^{out}(t) - \overline{r}_i^{out}(t-1)) \tag{10}$$

$$\overline{r}_j^{in}(t) = \overline{r}_j^{in}(t-1) + \eta(r_j^{in}(t) - \overline{r}_j^{in}(t-1)) \tag{11}$$

After each update the weights are again normalized to unit norm.

3 Results

We ran the simulation for 2.5 hours in rat time. The results show that this duration is sufficient for grid cells to develop. Figure 7 shows the time evolution of weights of a grid cell. The cell we picked has one of the most apparent grid structure among the other cells.

When all grid cells are inspected, it is seen that most grid cells have similar spatial frequencies, however, their orientations and off-sets are different. Figure 8 illustrates this difference.

4 References

neurons/neuron_w_41_t_0.png

neurons/neuron_w_41_t_50.png

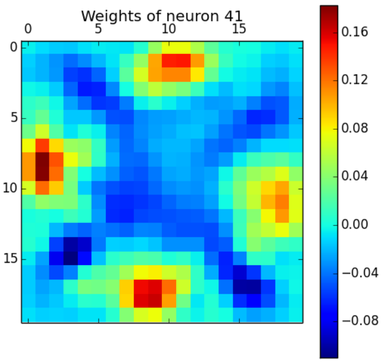
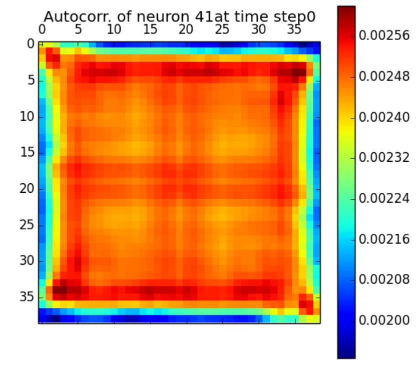
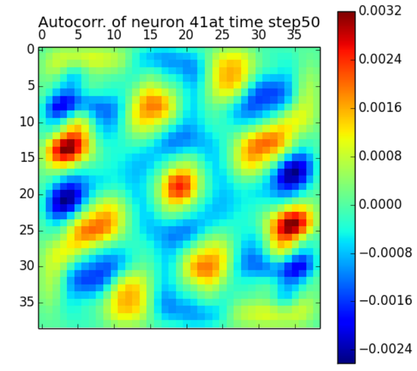


Figure 5: weight development





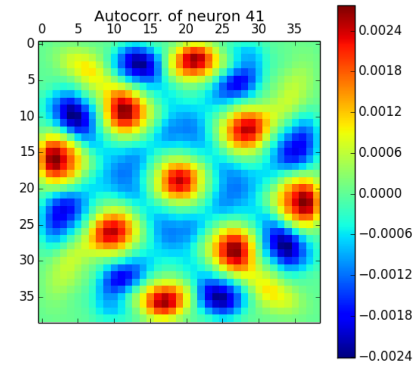
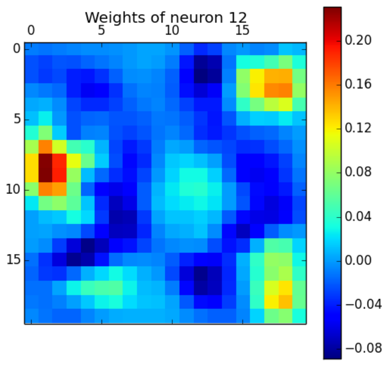


Figure 6: autocorrellation development

Figure 7: The weights and the autocorrelogram of a neuron at the beginning, middle and the end of the simulation



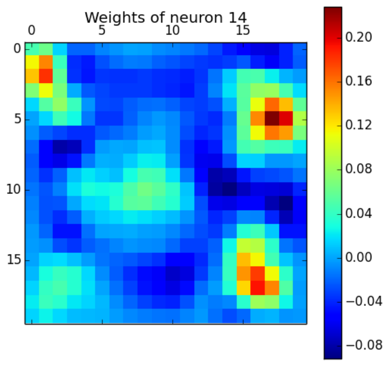


Figure 8: Different grid cells have different orientations and off-sets but similar spatial frequencies.