Computational Neuroscience: Problem set 5

Neural Networks

Ring network

Step 1: Modelling the inputs

We first model the inputs which depend on the orientation of visual stimuli. As described by Hubel and Wiesel, a neuron can respond preferentially to a visual stimulus of a certain orientation θ_0, where θ_0 is the orientation of a bar expressed in radian. It is thought that neurons from the visual cortex receive tuned inputs from the thalamus (i.e. the external inputs to the network). The magnitude of these inputs h_i^{ext} is proportional, for a given neuron, to the difference between its preferred orientation θ_i and the orientation of the stimuli θ_0:

$$h_i^{ext}(\theta_0) = c[(1-\epsilon) + \epsilon cos(2(\theta_i - \theta_0))], \tag{1}$$

where c represents the contrast of the stimuli and ϵ controls the selectivity of the input.

1.1 Write a MATLAB function computing h_i^{ext}. Note that MATLAB can operate with vectors, and use this property to compute the inputs for $N = 50$ neurons with a preferred direction varying from $-\pi/2$ to $\pi/2$ and distributed at equal interval. We give the first and last line of this function you have to write the middle line.

```
function out = h_input(theta0, theta, c, epsilon)

%Write your own code here

end
```

1.2 Plot the result of this function for all neurons with an input $\theta_0 = 0$, $c = 3$, and $\epsilon = 0.1$

1.3 The inputs to the neurons are non-linearly filtered given an activation function g. The function is defined by:

$$g(h) = \begin{cases} 0 & \text{if } h \leq T \\ \beta(h-T) & \text{if } T < h \leq T + 1/\beta \\ 1 & \text{if } h > T + 1/\beta \end{cases} \tag{2}$$

Write a MATLAB function implementing $g(h)$. Note that it should be able to take as inputs a vector, and that you can use statements like $a = b < t$ on vectors too.

1.4 Set $T = 0$ and $\beta = 0.1$. Plot g as a function of the input h when h is a scalar varying between -15 and 15.

Step 2: Modelling the neurons

2.1 The rate-based neuron models are described by:

$$\tau \frac{dm_i}{dt} = -m_i + g(h_i), \tag{3}$$

where m_i stands for the activity of neuron i, τ is a time-constant, g is the non-linear function and h_i is the input to neuron i.

Write a MATLAB function that implements this neuron model. We fix $\tau = 5$ms for all the following simulations. Therefore, the only argument of this function is the inputs h_i and m. Use the Euler method to simulate the neurons with a time step of 1ms.

2.2 We are working first with the simplest scenario for orientation tuning. This scenario corresponds to the case where the only source of input is the thalamus, i.e. there is no connection between the neurons.

Write a MATLAB function simulating $N = 50$ neurons for 30 iterations. The arguments of this function are: the initial activity of the network, θ_0, N (number of neurons), Ni (number of iterations), ϵ, c. Test the function with a zero initial activity, $\theta_0 = 0$, for 50 neurons and 30 iterations, $\epsilon = 0.9$, $c = 1.5$. Plot the activity of the network over time using the MATLAB function image (you might need to multiply the activity by 100 to see something).

2.3 Run this function with $c = 1.2$ and $c = 4$ and plot the neurons' activity in this case. What do you observe?

Step 3: Modelling the network

We now add connections within the neurons. And we set $\epsilon = 0.1$. The connection between two neurons depends on their tuning in the following way:

$$J_{ij} = -J_0 + J_2 cos(2(\theta_i - \theta_j)) \tag{4}$$

3.1 Write a function to generate the 2D matrix of connections containing the connections weight J_{ij} between neurons. Pick $J_0 = 86$ and $J_2 = 112$. Use *image* to plot this matrix and check that it has the desired form.

3.2 With the recurrent connections, the input to the neurons becomes:

$$h_i(\theta_0) = \sum_{j=1}^{j=N} J_{ij} m_j + h_i^{ext}(\theta_0) \tag{5}$$

Modify the function you just coded to include the connections. Test it with $c = 1.2$ and all the other arguments unchanged. Run the model again for $c = 1.5$ and $c = 4$. Plot the activity of the network over time and comment.

Step 4: Apply different stimuli to the ring network

4.1 Changing the stimulus: Run the network for 30 iterations with $\theta_0 = 0$. Use the last vector of activity as the initial activity for another simulation and change θ to $\theta_0 = 2\pi/3$. Run this simulation for 500 iterations. Use $c = 100$ and $\epsilon = 0.8$. Comment.

4.2 Removing the stimulus: Set $c = 1.2$ and $\epsilon = 0.1$. Run the model for 60 iterations, while removing the stimulus h_i^{ext} after 30 iterations. Comment.