580.422 System Bioengineering II: Neurosciences (Spring 2015)

Homework #10: Ring Networks and Self-Organizing Maps

Dr. Kechen Zhang

1. Computer experiment on a ring network

The attached MATLAB program `ringnet.m` simulates a ring network with N neurons, indexed from 1 to N, representing the values of a circular variable $(\theta_1, \theta_2, \cdots, \theta_N)$ spaced uniformly from 0 to 2π. The dynamical equation of the network is given by

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} W_{ij} g\left(u_j\right),$$

where u_i is the state of neuron i, and g is a sigmoidal gain function. This equation can also be written equivalently in the vector-matrix form:

$$\frac{d\mathbf{u}}{dt} = -\mathbf{u} + \mathbf{W} g(\mathbf{u}).$$

In `ringnet.m` , the time evolution is simulated in discrete steps by Euler's method:

$$\mathbf{u}(t+\Delta t) = \mathbf{u}(t) + \Delta t\left(-\mathbf{u}(t) + \mathbf{W} g(\mathbf{u}(t))\right),$$

where Δt is the time step. The weight matrix is given by $W_{ij} = w\left(\theta_i - \theta_j\right)$, where function $w(\theta)$ describes a center-surround weight structure with short-distance excitatory and long-distance inhibitory connections. We use a circular normal (Von Mises) function:

$$w(\theta) = \exp\left(\cos(\theta + S)\right) + B$$

The rotation-invariance of the network is reflected by the circulant weight matrix, i.e., its rows are circularly shifted version of one another.

Questions:

1. *Stationary activity bump:* In `ringnet.m`, the initial states are assigned randomly. Run the program multiple times from different initial states. Does the system always reach the same final equilibrium state?
2. *Traveling activity bump:* By default, the weight shift parameter $S = 0$. Now add a small shift of the weight by setting $S = 0.2$, and then run the program multiple times, starting from different initial states. Now repeat with different values, e.g., $S = -0.2$, and $S = \pm 0.5$. How is the moving activity bump related to the value of this shift parameter?
3. *Uniform state vs symmetry breaking:* Now reduce the inhibitory level of the weights from $B = -1.5$ to $B = -1$. Repeat 2 and describe what you find.

4. *Noisy weights:* By adding noise to the weight matrix, the network is no longer strictly rotation-invariant. In `ringnet.m`, add the following line after the weight matrix is obtained: `W=W+randn(size(W))`. Repeat 2 and describe what happens.

2. Computer experiment on self-organizing map

The MATLAB program `SOM.m` simulates a self-organizing map with 100 neurons on a 10×10 grid. These neurons respond to the direction of visual motion (like MT neurons). The stimulus direction changes randomly from trial to trial with uniform distribution from 0 to 2π. The preferred direction of each neuron at grid location $\mathbf{r} = (i, j)$ is described by a weight vector $\mathbf{w}_{\mathbf{r}}$. The smaller the angle between the actual movement direction $\mathbf{v}$ and the preferred direction $\mathbf{w}_{\mathbf{r}}$, the larger the response. The actual response is not computed. What is important is the location $\mathbf{r}^*$ of the neuron with the largest response. The preferred direction vectors are updated according to $\Delta \mathbf{w}_{\mathbf{r}} = H\left(|\mathbf{r} - \mathbf{r}^*|\right)\left(\mathbf{v} - \mathbf{w}_{\mathbf{r}}\right)$ where $H\left(|\mathbf{r} - \mathbf{r}^*|\right) = \exp\left(-|\mathbf{r} - \mathbf{r}^*|^2 / 2\sigma^2\right)$ is a Gaussian neighborhood function. In self-organizing map simulations, the width of the neighborhood function is often decreasing in time. In this experiment, we simply use fixed width.

By default, the program `SOM.m` pauses for 1 second in each trial. In the animated display, the arrows indicate the preferred directions of all the neurons. The red circle indicates the winner neuron with the highest response. The red line segment indicates the direction of the actual movement. Note that only the weights of the neurons whose physical locations are close to the winner are updated.

Questions:

Please perform the following experiments in sequence. In each of the 4 conditions, run the program several times and describe briefly what happens to the final map of the preferred directions. Explain briefly and intuitively why your results make sense.

1. *Self-organization*. Set parameter `Ntrials=1000`, and comment out the `pause` command at the end of the loop, then run the program.
2. *Narrow neighborhood*. Use the same parameters as in question 1 but make the Gaussian neighborhood function very narrow by setting the width `s=0.1`.
3. *Wide neighborhood*. Make the Gaussian neighborhood function very wide by setting the width `s=1000`.
4. *Restricted inputs*. Now set the width back to its original value `s=1.5` as in question 1. Change the range of input angle from $[0, 2\pi]$ to $[0, \pi]$. That is, the line `a=rand*2*pi` in the beginning of the loop should be changed to: `a=rand*pi.`