Methods in Computational Neuroscience

Firing rate models and dynamical systems

arianna.di.bernardo@ens.psl.eu

22.03.2024

Updated file: this document contains 4 exercises. All exercises must be included in the third report. Max pages: 15

1. Neuron with self-connection.

In this exercise, we will explore the behavior of a neuron with an autapse - a chemical or electrical synapse from the neuron to itself.

This simple neural network is depicted in Figure 1, and the dynamics of its firing rate is described by the following differential equation:

$$\frac{dr(t)}{dt} = -r(t) + f(wr(t) + I), \tag{1.1}$$

where r is the neuron's firing rate, i.e. the average number of spikes per second, $w = 0.05$ is the synaptic strength and $I = -3$ is some constant, external (and inhibitory) background input. The activation function of the neuron is given by a sigmoidal function:

$$f(s) = 60(1 + tanh(s)) \tag{1.2}$$

where s is the total input to the neuron.

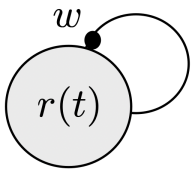

Figure 1: Simple neural network: a neuron whose output feeds back onto itself via a synapse.

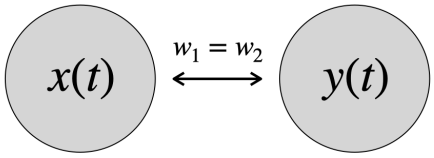

Figure 2: Mutual excitation.

1.1 Plot the activation function. Plot the neuron's activation function $f(s)$ for a suitable range of inputs s.

1.2 Study the dynamics. Simulate the system, using $dt = 0.1$. Focus on the cases in which the initial conditions are $r(0) = 59, r(0) = 60, r(0) = 61$. What is happening, and why?

1.3 Add noise. Add noise to the system and simulate it with the 3 initial conditions of the previous exercise:

$$\frac{dr(t)}{dt} = -r(t) + f(wr(t) + I) + \sigma\eta(t), \tag{1.3}$$

where η is gaussian white noise and σ is the noise magnitude. Start with a low-level of noise, $\sigma = 0.1$. What happens and why? Increase σ more and observe the consequences.

1.4 Study the flux. Plot the derivatives $\frac{dr}{dt}$ as a function of the neuron's firing rate r. Find the zero-crossings of this plot and explain their meaning.

1.5 Bifurcation diagram. Create a plot to visualize how the number of solutions (the zero-crossings of the previous exercise) changes with different values of synaptic strength (w) and external input (I). This plot is called *bifurcation diagram*. Set w to vary from 0 to 1 with increments of 0.01, and I to vary from -5 to 0 with increments of 0.05. Generate a 100x100 empty matrix. For each combination of w and I, calculate the number of solutions and store this value in the matrix. Finally, visualize the matrix and explain what you see.
[• w and I are two vectors containing 100 elements each: then you have 100x100 possible combinations of values. Each one of this combination will result in a different number of solution of the system. • To plot a matrix, you can use the command "plt.imshow(matrix)"].

2. 2D network: mutual excitation

In this exercise, we will address the study of a circuit of two neurons that excite each other (Figure 2). The firing rate model is described by the following system of coupled differential equations:

$$\begin{aligned} \frac{dx(t)}{dt} &= -x(t) + f\left(w_2 y(t) + I\right) \\ \frac{dy(t)}{dt} &= -y(t) + f\left(w_1 x(t) + I\right) \end{aligned} \tag{2.4}$$

where x, y are the firing rates of the two neurons. The synaptic strengths are positive (i.e. excitatory neurons): $w_1 = w_2 = 0.4$, and $I = -10$ is the external input. The activation function is the sigmoid function:

$$f(s) = 50\sigma(s) = 50\left(\frac{1}{1+e^{-s}}\right) \tag{2.5}$$

with s representing the total input to the neuron.

2.1 Plot the dynamics. Plot the evolution in time of the firing rate of the two neurons $x(t)$ and $y(t)$, for different initial conditions (on the same plot).

2.2 State space. Plot the trajectories in the *state space*, for different initial conditions (i.e. plot $x(t)$ vs $y(t)$ for different values of $[x(0), y(0)]$).

2.3 Nullclines. Compute the nullclines of the system and plot them on the same figure of point 2.2. What do you observe?

2.4 Fixed points. What do the crossing points of the nullclines indicate? Find the fixed points of the system (numerically) and use the plot of point 2.3 to infer qualitative conclusions about their stability. [Where the trajectories go?]

2.5 [Advanced] Study the stability of the fixed points analitically. Compute the Jacobian matrix of the system (3.7), and study its trace and determinant to obtain more accurate stability results. [The derivative of the sigmoid function $\sigma(s)$ is $\sigma'(s) = \sigma(s)(1-\sigma(s))$].

2.6 Bifurcation diagram Decrease the input current from -10 to -20 and observe how the nullclines plot varies. Compute the number of solutions as I varies, and plot the bifurcation diagram as function of I. How many regimes exist? [Advanced] Which kind of bifurcation is represented?

3. ND network: Hopfield neural network

In this exercise, we try to construct a Hopfield network, consisting of 64 neurons. The dynamics of the network is given by the following system:

$$\frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + f(W\boldsymbol{x}(t)) + \sigma\boldsymbol{\eta}(t), \tag{3.6}$$

where $\boldsymbol{x}(t)$ is a 64-dimensional vector collecting the firing rate of the 64 neurons in the network, $\boldsymbol{x}(t) = [x_1(t), x_2(t), ..., x_{64}(t)]$, and W is the connectivity matrix (64x64). The activation function is the sign function:

$$\begin{aligned} f(s) &= +1, \ s > 0 \\ f(s) &= 0, \ \ s = 0 \\ f(s) &= -1, \ s < 0 \end{aligned} \tag{3.7}$$

and $\sigma = 0.1$ is the noise amplitude.

To store a pattern $\boldsymbol{p}$ (a 64-dimensional vector) in the Hopfield network, we set the weight matrix to $W_{ij} = \frac{1}{N} p_i p_j$, i.e. the weight matrix is the outer product of two vectors,

$$W = \frac{1}{N}\boldsymbol{p}\boldsymbol{p}^T, \tag{3.8}$$

where $\boldsymbol{p}$ is a column vector and $\boldsymbol{p}^T$ is its transpose, i.e. a row vector.

3.1 Invent a pattern. $\boldsymbol{p}$ is 64-dimensional vector. Each component p_i of this pattern should take either the value -1 or 1. One way to visualize $\boldsymbol{p}$ is to map it into a matrix (a 64-dimensional vector can be mapped into a 8x8 matrix). Plot the pattern. [To map a vector onto a matrix (and vice versa), you can use the function reshape()]

3.2 Plot W and simulate the dynamics. Compute and plot the matrix W, as described above. Simulate the network with stepwidth $dt = 0.1$, using random initial conditions. To see what happens, plot the network activity for each time step. Note that you will have to remap the 64-dimensional network activity $x(t)$ at each time step onto the 8 × 8 matrix (you have to plot a sequence of matrices).

3.3 Fixed points Repeat the simulation a few times. At what fixed points does the network converge?

3.4 Introduce a new pattern Try to create a second pattern $\boldsymbol{q}$ and design the weight matrix according to:

$$W = \frac{1}{N}(\boldsymbol{p}\boldsymbol{p}^T + \boldsymbol{q}\boldsymbol{q}^T), \tag{3.9}$$

Repeat the previous experiment (point 3.2 and 3.3) and see what happens.

3.5 Initial conditions Give the network as an initial condition a ”corrupt” pattern, and verify that the network recovers the original pattern. For example, you can add noise to the pattern $\boldsymbol{p}$ (or $\boldsymbol{q}$) and use it as initial condition.

3.6 Network capacity If you construct more and more patterns, how many patterns can you store in your network with N = 64 neurons? To design a weight matrix with M patterns $\boldsymbol{p}_i$, the general rule is

$$W = \frac{1}{N} \sum_{i}^{M} \boldsymbol{p}_i \boldsymbol{p}_i^T. \tag{3.10}$$

(More specifically, if you set the initial conditions close to one of the patterns, does your network relax back to that pattern or not?)

4. ND network and Principal component analysis (PCA): the 'ring' network model

In this last exercise, you will implement an N-dimensional network called *ring attractor model* (I will not explain why it has this name; you will find out by completing the exercise!). Consider a recurrent network of N=100 neurons.
The dynamics of the network is given by the following system:

$$\frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + J\phi(\boldsymbol{x}(t)), \tag{4.11}$$

where the activation function is $\phi(s) = tanh(s)$.

The connectivity matrix of this network, J, is such that the strength between neurons i and neuron j is given by

$$J_{ij} = 2cos(\theta_i - \theta_j) \tag{4.12}$$

where θ assumes values in $[0, 2\pi)$.

4.1 Connectivity matrix. Compute the matrix J and plot it. What do you see?

4.2 Compute the dynamics. Use the Euler method to implement the dynamics. Use $dt = 0.1$ and $T = 1000$ (=number of time steps). Plot the dynamics $x_i(t), i = 1, ..., N$.

4.3 Principal component analysis (PCA). Collect the dynamics in the matrix X. This matrix has shape $N \times T = 100 \times 1000$ (100 neurons, 1000 time steps). Perform the PCA on the dynamics X and plot the covariance matrix and the first principal component. What is its explained variance?

4.4 PCA on a set of attractors For different initial conditions, collect the network attractors. Simulate the dynamics 500 times (be careful to use different initial conditions each time), and collect the 500 attractors into a matrix Q. This matrix will have dimension $100x500$ (each attractor is a fixed point in a 100-dimensional space!). To visualize it, perform a PCA on the set of attractors Q. Plot the covariance matrix and the eigenvalues. What is the explained variance? How many components does it make sense to consider? Plot the principal components in the plane $PC_1 - PC_2$ and discuss the results. [Depending on the method used to collect the attractors you may have less than 500 attractors (some can be repeated in the set).]

4.5 PCA with scikitlearn Use the python command *sklearn.decomposition.PCA* to implement PCA on the matrix Q and verify that the result is equal to what you got in the previous step.

[For 4.3 and 4.4 the PCA must be calculated by hand (in python) and without using scikitlearn's pca function.]