AT2 – Atelier Neuromodélisation

Problem set #4
NETWORKS

PROBLEM 1 *Neuron with Autapse*

Consider a neuron whose output feeds back onto itself via a synapse. (Such a synapse is often called an "autapse".) We assume that the neuron's firing rate x, i.e., its average number of action potentials per second, is given by the differential equation

$$\dot{x}(t) = -x(t) + f\big(wx(t) + I\big)$$

where $w = 0.04$ is the strength of the synaptic connection and $I = -2$ is some constant, external (and inhibitory) background input. The input-output (or activation) function of the neuron is given by a sigmoidal function which we assume to be

$$f(s) = 50(1 + \tanh(s))$$

where s is the total input to the neuron.

Programming Help: *The hyperbolic tangent is defined as* $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$. *In Numpy you can use the function* `np.tanh()`.

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

(b) Plot the derivatives $\dot{x}$ as a function of the neuron's firing rate x. What do the zero-crossings of this plot indicate?

(c) Simulate the system for different initial conditions, $x(0) = 49$, $x(0) = 50$, and $x(0) = 51$, using a time step $dt = 0.1$. Choose a reasonable value of T, What are the outcomes of these simulations? Why?

(d) Redo the simulation of (c), but add a noise component to the system, so that

$$\dot{x}(t) = -x(t) + f\big(wx(t) + I\big) + \sigma\eta(t)$$

where $\eta(t)$ is Gaussian white noise. What happens for a noise value of $\sigma = 5$? What happens if you increase the noise (up to $\sigma = 80$)? Why?

PROBLEM 2 *Circuit with mutual inhibition*

We now consider a circuit of two neurons that are coupled by mutual inhibition. For the firing rates of these two neurons, x_1 and x_2, we assume the differential equations

$$\dot{x}_1(t) = -x_1(t) + f\big(wx_2(t) + I\big)$$
$$\dot{x}_2(t) = -x_2(t) + f\big(wx_1(t) + I\big)$$

where $f(\cdot)$ is defined as before and the inhibitory synaptic weights are given by $w = -0.1$. The external inputs are assumed to be excitatory, $I = 5$.

(a) Plot the null-isoclines ("nullclines") of the system, i.e., the line for which $\dot{x}_1(t) = 0$ and the line for which $\dot{x}_2(t) = 0$. What do the crossing points of these lines indicate?

(b) Simulate the system (time step $dt = 0.1$)! Choose an initial condition for $x_1(0)$ and $x_2(0)$ and plot the evolution of the firing rates of the two neurons into the same plot as the nullclines from (a). What happens for different initial conditions?

(c) So far we have treated each neuron separately, i.e., each neuron got its own equation. When we use matrix-vector notation, we can summarize these two equations into one:

$$\mathbf{x}(t) = -\mathbf{x}(t) + f\big(W\mathbf{x}(t) + \mathbf{I}\big)$$

where

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \qquad W = \begin{pmatrix} 0 & w \\ w & 0 \end{pmatrix} \qquad I = \begin{pmatrix} I_1 \\ I_2 \end{pmatrix}$$

We here follow the convention in the literature by which vectors are denoted by bold-faced letters, and matrices by capitalized letters. Reprogram your program from (b) using Scipy/MATLAB/Numpy's intrinsic ability to compute with matrices and vectors! (Note that **x** and **I** are *column*-vectors!)

(d) (*Advanced*) Plotting the vector field of derivatives In addition to the nullclines, use the Scipy/MATLAB/matplotlib function `quiver` to plot the vector field of derivatives!

PROBLEM 3 *Optional: Hopfield Model*

In this exercise, we seek to construct a Hopfield network with $N = 64$ neurons, whose dynamics are given by the differential equation

$$\mathbf{x} = -\mathbf{x} + f(W\mathbf{x}) + \sigma\eta(t) \tag{1}$$

where **x** is the vector of firing rates, W is the $N \times N$ weight matrix, and $\sigma = 0.1$ is an external noise level. For mathematical simplicity, we will set the activation function to $f(x) = \text{sign}(x)$, so that neural activity can run from -1 to 1. (This step is simply for mathematical convenience, i.e., it makes the math simpler. If necessary, however, one can also simulate the Hopfield network with a more realistic input-output functions such as the one we used above.) To store a pattern **p** in the Hopfield network, we set the weight matrix to $W_{ij} = p_i p_j$, i.e., the weight matrix is the outer product of two vectors,

$$W = \frac{1}{N}\mathbf{p}\mathbf{p}^T \tag{2}$$

where by convention, **p** is assumed to be a column vector, and the superscript ^{T} denotes the matrix transpose; here, it makes a column vector into a row vector.

Programming Help: *The matrix transpose is very simple: if your matrix is called* `A`*, then its transpose is* `A'` *in MATLAB and* `A.T` *in numpy.*

(a) Invent a pattern **p**! Each component p_i of this pattern should take either the value -1 or 1. One way to visualize the "pattern" **p** is to map this vector onto a matrix – for a network of $N = 64$ neurons, you can for instance map the 64-dimensional vector **p** onto a matrix of size $[8 \times 8]$.

To map a vector onto a matrix (and vice versa), you can use the function `reshape()` in MATLAB. In Scipy, you must use the method `reshape` of the `ndarray` object.

You can plot matrices using `imshow` (or `pcolor` for fancier plots) in matplotlib, and `imagesc()` in MATLAB.

(b) Create the weight matrix W as described above and simulate the network with stepwidth $dt = 0.1$, using random initial conditions. To visualize what is going on, try to plot the network activity at each time step using the false-color-plot from above. Note that you will have to remap the 64-dimensional network activity $\mathbf{x}(t)$ at each time step onto the 8×8 matrix! Repeat the simulation several times. What fixed point(s) does the network relax to?

(c) Try to create a second pattern $\mathbf{q}$ and design the weight matrix according to

$$W = \frac{1}{N}\left(\mathbf{p}\mathbf{p}^T + \mathbf{q}\mathbf{q}^T\right) \tag{3}$$

Repeat the experiment from (c). Verify that the network retrieves the right pattern by starting from initial conditions that are close to that pattern.

(d) The general rule is that for a set of M patterns $\mathbf{p}_i$, the weight matrix is constructed as

$$W = \frac{1}{N}\sum_{i=1}^{M} \mathbf{p}_i \mathbf{p}_i^T \tag{4}$$

If you construct more and more patterns, how many patterns can you store in your network with $N = 64$ neurons? (More specifically, if you set the initial conditions close to one of the patterns, does your network relax back to that pattern or not?) Investigate what happens if you replace the sign-function with a $\tanh$-function.

Background Literature: Hopfield (1982). Proc. Natl. Acad. Sci. 79:2554–8.
Dayan & Abbott, Chapter 7.4 ("Associative Memory")