Homework 8
Systems & Theoretical Neuroscience [SWC]

Due: Mon, 5th March

1 Motivation

a) Read http://neuronaldynamics.epfl.ch/online/Ch17.S1.html for some context.

2 Implementing your own Hopfield network

In this question you will implement a small Hopfield network to learn 4 variables, i.e. $\mathbf{x}$ is a vector of 4 numbers, where each value is either $+1$ or -1.

2.1 The dynamics of a random network

a) Write a function to generate a random 4x4 weight matrix, where the diagonal terms are constrained to be zero and the off-diagonals are symmetric (i.e. $W_{ij} = W_{ji}$).

b) Write a function that applies the hopfield network update equations for a given $\mathbf{W}$ and $\mathbf{x}$. You may choose to apply the updates synchronously or asynchronously.

The Hopfield update equation is given by:

$$x_i(t+1) = sgn\left(\sum_j W_{ij} x_j(t)\right)$$

where $sgn(x)$ is $+1$ if $x > 0$ and -1 if $x < 0$

c) Is this update equation biologically plausible? Justify your answer.

d) Generate a random weight matrix and a random starting value of $\mathbf{x}$. Apply the hopfield updates. Does the system converge to a fixed point? If so, how quickly? Try with different initial $\mathbf{x}$, do you always get the same fixed point? If not, how many are there?

e) Write a method that takes a given weight matrix, and finds all the fixed points in the system by trying every possible starting $\mathbf{x}$. Try different (random) weight matrices. How many fixed points do you typically find?

2.2 Learning input patterns

A network with a memory is only useful if it can learn specific memories. In this section we will use a Hebbian rule for learning a Hopfield network. Other learning rules have been proposed.

The Hebbian rule is given by:

$$W_{ij} = \frac{1}{M} \sum_\mu x_i^\mu x_j^\mu$$

where M is the number of patterns we are learning and $\mathbf{x}^\mu$ is the μ^{th} input pattern we are learning.

a) Write a function to learn weights using the Hebbian rule

b) Is this rule biologically plausible? Justify your answer.

c) Pick 2 patterns (specific values of $\mathbf{x}$) to learn, and run the hebbian rule to learn $\mathbf{W}$. Has the system learned your patterns? Are there any other fixed points? Does your answer change if you learn different patterns?

d) What happens if you try to learn more patterns?

3 Higher dimensional Hopfield networks

In this question we will play with slightly higher-dimensional networks in order to learn more interesting patterns.

a) Install the neuronal-dynamics python libraries following the instructions at https://neuronaldynamics-exercises.readthedocs.io/en/latest/exercises/setup.html

b) Complete the exercise on hopfield networks at https://neuronaldynamics-exercises.readthedocs.io/en/latest/exercises/hopfield-network.html