Coursework 1

This coursework relates to the Hopfield network. This coursework may appear before we have covered Hopfield networks in the lectures, but you can still get started!

In a Hopfield network you have a network of neurons; these neurons can be in an up state:

$$x = 1 \tag{1}$$

or a down state

$$x = -1 \tag{2}$$

In the synchronous version we will use here all the neurons are updated at the same time; I will put a hat on the new value, obviously in the nex iteration the new value becomes the current value and loses the hat. Here there is no threshold and the update rule is:

$$\hat{x}_i = \begin{cases} 1 & \sum_j w_{ij} x_j \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

You can store patterns in a Hopfield network by setting the synapse strengths, that is the w_{ij}'s. The more brain-like version has a learning rule for changing w_{ij}; here we will anticipate the end result of that learning and stipulate that

$$w_{ij} = \frac{1}{N} \sum_a x_i^a x_j^a \tag{4}$$

where for each of N values of the index a the x_i^a's represent the patterns we wish the network to store. The idea is that after the synapses have been set, that is after the network has learned, if a partial patter is presented, so some but not all of the values of x_i^a for one of the a's, the other x_i^a will adjust to complete the pattern.

In the `coursework1` folder you will find programmes `seven_segment.jl` and `seven_segment.py`. These contain a function for converting an 11-component vector of ones and -1's into an old-fashioned seven-segment digit[1] and a number: the first seven digits correspond to the seven-segment display and the remaining four code for the number in a sort of binary where the zeros have been replaced with -1. It also contains three patterns: one, three and six.

[1] `https://en.wikipedia.org/wiki/Seven-segment_display`

The goal of this coursework is to extend one or other of these programmes to include a Hopfield network to store these three patterns using the formula for w_{ij} when $i \neq j$

$$w_{ij} = \frac{1}{N} \sum_a x_i^a x_j^a \tag{5}$$

where a labels patterns, x_i^a is the activiation of the ith node in the ath pattern and N is the number of patterns. $w_{ii} = 0$.

In the code there are also two test patterns, your programme should update these synchronously until convergence and print out the patterns at each iteration.

This coursework is intended to check you understand Hopfield networks, you are not being asked to do anything over elaborate beyond that. In summary you should:

- Create a weight matrix of w_{ij}s.
- Fix the weight values to store the three patterns.
- Write a function to evolve the network according to the McCulloch-Pitts formula; this should be done synchronously so all the nodes are updated at each timestep. Assume the threshold $\theta = 0$.
- For each of the two test patterns, evolve the patterns until they stop changing, printing the pattern at each step.

Add to the above a function to calculate the energy of a configuration

$$E = -\frac{1}{2} \sum_{ij} x_i w_{ij} x_j \tag{6}$$

and print out the energy of the three learned patterns, of the test patterns and any patterns formed as the patterns are updated.

Submission instructions

This coursework used to be for credit and so there is a submission script, if you are curious you can still look at it, it generated an automatic report which was then uploaded, instructions are in `submission.README`. While this coursework is no longer marked, I do, as usual, offer a small prize for anyone who does the project in a genuinely odd choice of programming language.

`coms30127.github.io`