Neural Networks and Biological Modeling

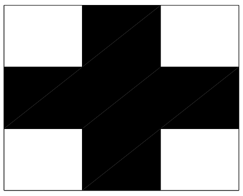
Professor Wulfram Gerstner
Laboratory of Computational Neuroscience

Question set 5

Exercise 1

Consider a neural network composed of 9 neurons.

1.1 This network uses the standard learning rule $w_{ij} = p_i p_j$ to learn the following pattern :



Now keeping the learned weights fixed, present a pattern $S_i(t = 0)$ and let it evolve according to:

$$S_i(t+1) = sgn\left(\sum_j w_{ij} S_j(t)\right) \tag{1}$$

Suppose the initial state is again the swiss cross above but with one bit flipped. Will the dynamics correct it?

1.2 Suppose that N bits are flipped. Will the dynamics correct them?

Exercise 2

Consider the evolution of synaptic weights given by the following equation:

$$\frac{d}{dt} w_{ij} = a_2^{\mathrm{corr}} (\nu_i^{\mathrm{post}}(t) - \vartheta)(\nu_j^{\mathrm{pre}}(t) - \vartheta)\,. \tag{2}$$

2.1 Imagine that you can control the value of ϑ, and a_2, for two layers of neurons connected with the weights w_{ij} and firing with a frequency defined by the vectors ν_j^{pre} and ν_j^{post} that you can also control. Design a protocol such that these dynamics yields the optimal weights for a Hopfield network:

$$w_{ij} = \sum_\mu p_i^\mu p_j^\mu\,, \tag{3}$$

where p is a vector describing one prototype, and the sum runs over all prototypes.

2.2 To bridge with the previous exercises compare this learning rule (eq.3) with the general formulation,

$$\frac{d}{dt} w_{ij} = a_0 + a_1^{\mathrm{pre}} \nu_j^{\mathrm{pre}} + a_1^{\mathrm{post}} \nu_i^{\mathrm{post}} + a_2^{\mathrm{corr}} \nu_j^{\mathrm{pre}} \nu_i^{\mathrm{post}}\,. \tag{4}$$

Exercise 3: Associative memory - Homework

Consider a Hopfield network having stored 4 patterns:

$$\begin{aligned} p^1 &= \{\xi_1^1, \cdots, \xi_N^1\} \\ &\cdot \\ &\cdot \\ p^4 &= \{\xi_1^4, \cdots, \xi_N^4\} \end{aligned} \tag{5}$$

Assume that the four patterns are orthogonal, i.e., $\frac{1}{N}\sum_{i=1}^N \xi_i^\mu \xi_i^\nu = \delta^{\mu\nu}$, where $\delta^{\mu\nu}$ is the Kronecker symbol

$$\delta^{\mu\nu} = \begin{cases} 1 & \text{if } \mu = \nu \\ 0 & \text{otherwise} \end{cases}$$

You present the network with an activity pattern that has overlap[1] with ξ^3 only (no overlap with other memories). The activity dynamics is given by

$$S_i(t+1) = g\left(\sum_j w_{ij} S_j(t)\right) \tag{6}$$

Discuss the temporal evolution of the system over the first 3 time steps

3.1 when g is the sign function

3.2 when g is an odd and monotonically increasing function mapping the real line onto $[-1;1]$.

Exercise 4: Probability of error in the Hopfield model

Consider a Hopfield network trained on a set of patterns ξ_i^μ. It was shown in the course that even if the network is initialized exactly in one of the learned patterns, i.e., in a state $S_i(t) = \xi_i^\nu$ for all i, there is a small probability that a pixel will flip.

This probability P_{error} is related to a random walk of $N(p-1)$ steps and step size $1/N$. The distance made by the random walk is called a.

4.1 Show that

$$P_{\text{error}} = \frac{1}{2}\left[1 - \text{erf}\left(\sqrt{\frac{N}{2(p-1)}}\right)\right] \tag{7}$$

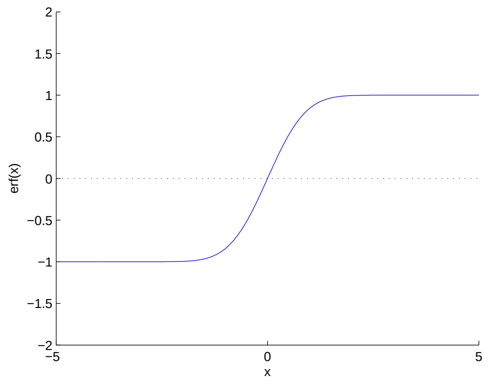
where erf is the error function, defined by

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-x'^2} dx' \,. \tag{8}$$

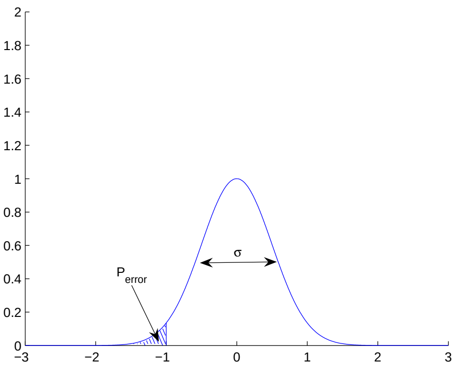
4.2 We require for perfect retrieval that at most one pixel is "wrong". What is then the maximum number of patterns that can be stored in a network of $N = 1000$ or $N = 10000$ neurons?

4.3 Same as 4.2 but the condition for perfect retrieval is that the *percentage* of wrong pixels is below 0.1%.

[1] by "having overlap with prototype μ" we mean with "having non-null scalar product with ξ^μ"



(a)



(b)

Exercise 5: Simulation of a Hopfield network

To test the theoretical results of the preceding exercises you will now simulate an associative memory network. You will use a model in which neurons are pixels and take the values of -1 (*off*) or +1 (*on*). The network can store a certain number of pixel patterns, which is to be investigated in this exercise. During a retrieval phase, the network is started with some initial configuration and the network dynamics evolves towards the stored pattern (attractor) which is closest to the initial configuration. The dynamics is that of equation (1).

In the Hopfield model each neuron is connected to every other neuron (full connectivity). The connection matrix is

$$w_{ij} = \frac{1}{N} \sum_{\mu} p_i^{\mu} p_j^{\mu}$$

where N is the number of neurons, p_i^{μ} is the value of neuron i in pattern number μ and the sum runs over all patterns from $\mu = 1$ to $\mu = P$. This is a simple correlation based learning rule (Hebbian learning). Since it is not a iterative rule it is sometimes called one-shot learning. The learning rule works best if the patterns that are to be stored are random patterns with equal probability for on (+1) and off (-1). In a large networks (N to infinity) the number of random patterns that can be stored is approximately 0.14 times N.

This session will also introduce you to the concept of classes or object oriented code. You can find a tutorial at

```
http://docs.python.org/tutorial/classes.html
```

Download `hopfield.py` and the folder `alphabet/` from moodle. Open ipython and type

```
>> import hopfield
```

The module contains two classes

- `hopfield_network` the network itself which contains *methods* to create patterns and run a retrieval phase
- `alphabet` which creates pixel patterns out of the gif files contained in `alphabet/`

Create an instance of your `hopfield_network` class (here of size 4×4 for instance)

```
>> hn = hopfield.hopfield_network(4)
```

Now you can create and store patterns using

```
>> hn.make_pattern()
```

(default is one random pattern with half of its pixels *on*) and test whether it is stored with

```
>> hn.run()
```

which, by defaults, runs the dynamics for the first pattern with no pixel flipped.

4×4 network

5.1 What is the experimental maximum number of random patterns the network is able to memorize? Store more and more random patterns and test retrieval of some of them. The first few patterns should be stored perfectly, but then the performance gets worse. Does it correspond to the theoretical maximum number of random patterns the network is able to memorize?

10×10 network

5.2 Repeat question 5.1

5.3 Store a finite number of random patterns, e.g. 8. How many wrong pixels can the network tolerate in the initial state so that it still settles into the correct pattern?

5.4 Try to store characters as the relevant patterns. How good is the retrieval? What is the reason? (to store characters see the doc string of the method `make_pattern`)

next level. . .

5.5 Try one of the preceding points in bigger networks. Try adding a smooth transfer function g to the neurons.