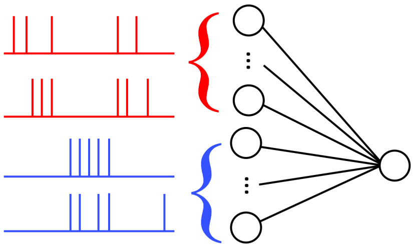
Neural Networks and Biological Modeling

Professor Wulfram Gerstner
Laboratory of Computational Neuroscience

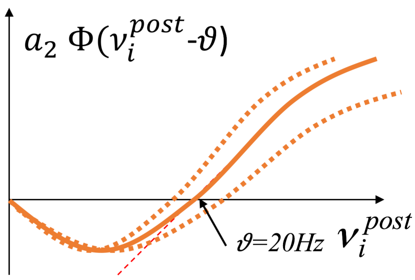
Question set 6

Exercise 1: Synaptic Plasticity: the BCM rule

A neuron receives 20 inputs that are organized in two groups of 10 inputs. The two groups fire in alternation: when group 1 is active, group 2 is silent; when group 2 is active, group 1 is silent. The input switches between the two groups every second (see figure 1(a)). All initial weights are $w_{ij} = 1$, but weights can change according to the BCM rule (eq. 1 with $\vartheta = 20Hz$). The firing rate of the postsynaptic neuron ν_i^{post} is given by eq. 2. The shape of Φ is shown in figure 1(b).



(a) One postsynaptic neuron receives input from 20 presynaptic neurons.



(b) weight-change as a function of ν_i^{post}. The solid line shows it for $\vartheta = 20Hz$.

Figure 1: Network and weight-change

$$\frac{d}{dt} w_{ij} = a_2^{corr} \Phi(\nu_i^{post} - \vartheta)\nu_j^{pre} \tag{1}$$

$$\nu_i^{post} = g(I_i) = \sum_j^N w_{ij}\nu_j^{pre} \tag{2}$$

a) Assume that group 1 fires at 3Hz, then group 2 at 1 Hz, then again group 1 etc. How do the weights of both groups evolve?

b) Assume that group 1 fires at 3Hz, then group 2 at 2.5 Hz, then again group 1 etc. How do the weights of both groups evolve?

c) The inputs are as in part b, but now you are free to choose theta. Suppose that the synapse can measure the time-average postsynaptic rate $\overline{\nu}$. What would you propose as model of ϑ so that the weight-pattern becomes non-trivial?

Exercise 2: Hopfield networks and Hebbian learning

Here we explore how we may obtain a Hopfield network with M stored prototypes through Hebbian plasticity instead of fixing the weights explicitly.

This is achieved by presenting the patterns to a fully connected network and apply a plasticity rule:

$$\frac{d}{dt}w_{ij} = a_2^{\text{corr}}(\nu_i^{\text{post}}(t) - \vartheta)(\nu_j^{\text{pre}}(t) - \vartheta)\,, \tag{3}$$

where a_2 and ϑ are parameters of the plasticity model; $\nu_i^{\text{post}}(t)$ and $\nu_j^{\text{pre}}(t)$ are the activities of neurons i and j at time t.

We present a pattern μ to the network in the following way: Each pixel j of pattern μ, $p_j^\mu \in \{-1, +1\}$, stimulates exactly one neuron j in the network. That neuron's firing rate ν_j depends on the pattern: $\nu_j = 0\,Hz$ if $p_j^\mu = -1$; $\nu_j = 20\,Hz$ if $p_j^\mu = +1$.

During that presentation, the network learns the pattern by adjusting its weights according to the plasticity rule given in equation 3. We assume initial weights $w_{ij} = 0$. For this exercise, we use a constant threshold $\vartheta = 10\,Hz$.

2.1 We now have the network learn M patterns. Each one is presented once for 0.5 seconds. Show that, for an appropriate choice of a_2, the final weights are given by

$$w_{ij} = \sum_\mu p_i^\mu p_j^\mu\,. \tag{4}$$

Hint: Begin by calculating the weight change induced by presenting a single pattern for $0.5s$.

2.2 How does this learning rule map to the general formulation

$$\frac{d}{dt}w_{ij} = a_0 + a_1^{\text{pre}}\nu_j^{\text{pre}} + a_1^{\text{post}}\nu_i^{\text{post}} + a_2^{\text{corr}}\nu_j^{\text{pre}}\nu_i^{\text{post}} + \ldots ? \tag{5}$$

2.3 Would you describe this learning procedure as reinforcement or unsupervised learning?

Exercise 3: Hopfield network with probabilistic update

So far we have studied Hopfield networks with deterministic activity dynamics. That is, for the same input potential h a neuron always takes the same state:

$$S_i(t+1) = sign(h_i(t)) \tag{6}$$

In this exercise we model stochastic neurons by replacing that equation with a probabilistic state update:

$$P\{S_i(t+1) = 1 | h_i(t)\} = g(h_i(t)) \tag{7}$$

Let's say we have stored M patterns p^μ in a network of N neurons. We then set the network to an initial state $S(t_0)$ that has significant overlap with the third pattern and no overlap with other patterns: $m^{\mu\neq 3}(t_0) = 0$. For the deterministic update (eq. 6) we know (either from the textbook or from the proof done last week) we would retrieve pattern p^3 in a single update: $m^3(t_0+1) = g(m^3(t_0)) = 1$.

We now study how that result changes in the presence of noisy neurons (eq. 7). Look at figure 2 to get an intuition about the stochastic update.

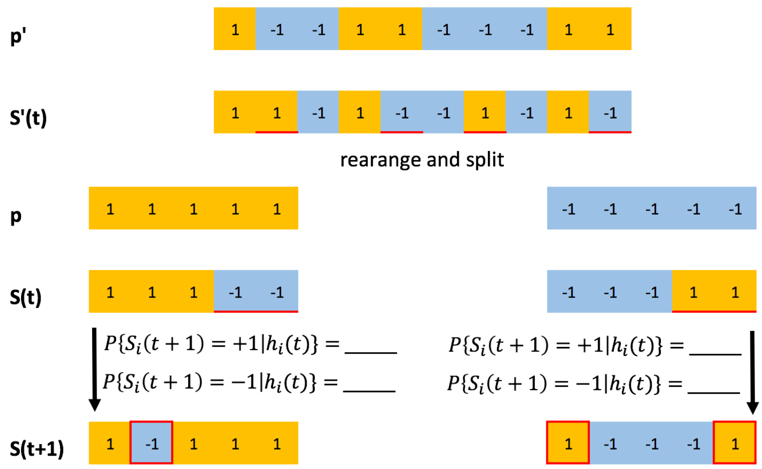


Figure 2: For the analysis of the overlap $m^3(t+1)$ it helps to rearrange pattern p and state S such that we can identify four sub-populations in the last row. We first split the neurons $S_i(t)$ into those that *should* be active and those that *should not* be active. All neurons in the same sub-population share the same probabilistic activity dynamics. In the last row, we see four groups of neurons which we label $\{p_i/S_i(t+1)\}$: {on/on}, {on/off}, {off/on}, {off/off}.

3.1 Derive the overlap $m^3(t_0+1)$ (eq. 8) under the state dynamics of eq. 7. Assume that there's only overlap with pattern p^3, and that for each pixel of the pattern 3, the probability to be on is $P\{p_i^3 = 1\} = 0.5$

$$m^3(t_0+1) = g(m^3(t_0)) - g(-m^3(t_0)) \tag{8}$$

Hints:

1. Use a result we derived earlier: $h_i(t_0) = p_i^3 m^3(t_0)$.
2. For each of the four groups (see figure 2) find the probabilities for $P\{S_i(t+1)|h_i(t_0)\}$
3. Recall the definition of the overlap m: $m^3(t_0+1) = \frac{1}{N}\sum_{i=1}^{N} p_i^3 S_i(t_0+1)$
4. For large N we can use the expected number of neurons in each of the four sub populations to express (the expected) overlap $m^3(t_0+1)$.

3.2

(a) In equation 7, what properties should the transfer function g have?

(b) Use $g(h) = \frac{1}{2}(tanh(h)+1)$ in equation 8. Simplify it, plot the function graph and discuss it.

Exercise 4: Hopfield, asynchronous update and the energy picture

Consider a Hopfield network of N neurons with an **asynchronous** update regime. That is, only *one* randomly selected neuron k is updated at each step according to equation 9:

$$\begin{cases} S_k(t+1) = g(h_k(t)) = sign\left(\sum_j^N w_{kj} S_j(t)\right) & \text{for exactly one randomly chosen neuron k} \\ S_i(t+1) = S_i(t) & \text{for all other neurons,} \quad i \neq k \end{cases} \tag{9}$$

For each state S of a Hopfield network, we can compute a scalar value, known as the **energy E** of the network:

$$E := -\sum_i^N \sum_j^N w_{ij} S_i S_j. \tag{10}$$

The evolution of the network state and the change of energy are related in an interesting way:

> When a network is updated asynchronously then the energy function $E(S(t))$ does either decrease or stays at a (local) minimum.

We will now proof this property:

In the trivial case of $S_k(t+1) = S_k(t)\ \forall k$ the network has reached a stable state and therefore the energy function is stable too: $\Delta E = E(t+1) - E(t) = 0$.

Now consider the case of one neuron k changing its state and proof, in steps 4.1 to 4.3, that the energy decreases:

4.1 The energy $E(t)$ in eq. 10 is summed over all pre- and post- synaptic neurons i and j. Rewrite that sum such that the contribution of neuron k to the total energy E appears explicitly.

Hint: To simplify the resulting expression, remember that in a Hopfield network, the weight are symmetric: $w_{ij} = w_{ji}$ and there are no self recurrent connections: $w_{kk} = 0$

4.2 Write the change in energy $\Delta E = E(t+1) - E(t)$ when exactly one neuron k does changes its state.

4.3 Proof that $\Delta E < 0$ when exactly one neuron k does changes its state under the dynamics of eq. 9

Exercise 5: Binary codes and spikes

A Hopfield model is specified by a binary variable $S_i \in \{-1, +1\}$, the weights (eq. 11) and the update dynamics (eq. 12).

$$w_{ij} = c \sum_{\mu=1}^{M} p_i^\mu p_j^\mu \quad \text{with } c = \frac{1}{N} \tag{11}$$

$$S_i(t+1) = sign\left(\sum_{j=1}^{N} w_{ij} S_j(t) \right) \tag{12}$$

For an interpretation in terms of spikes it is, however, more appealing to work with a binary variable σ_i which is zero or 1.

5.1 Rewrite the Hopfield model in terms of $\sigma_i \in \{0, 1\}$, $S_i = 2\sigma_i - 1$.

5.2 Assume that the patterns have the property $\sum_{i=1}^{N} p_i^\mu = 0 \quad \forall \mu$.
Discuss that condition and use it to simplify the update dynamics found in the previous question.