M2 Cogmaster 2025 | October 16th
TA: Yoan Fardella (yoan.fardella@ens.psl.eu)

THEORETICAL NEUROSCIENCE

TD6: SUPERVISED LEARNING

All TD materials will be made available at `https://github.com/yfardella/Th_Neuro_TD_2025`.

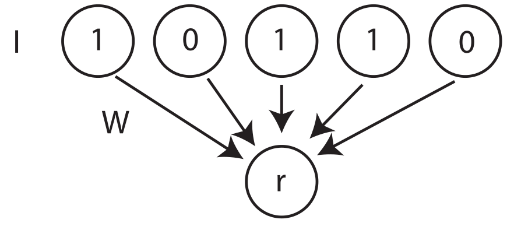
Several learning paradigms exist: mainly supervised, unsupervised and reinforcement learning. In these next three tutorials, we will develop standard models in each of these three paradigms, starting with the supervised perceptron. The perceptron model originates with Rosenblatt in 1958 and marks the beginning of what we know today as Artificial Neural Networks, which have in recent years attained unexpectedly high performance in various tasks, placing learning as a central question of Neuroscience.

1 Perceptron model

We consider a neuron receiving input from N other neurons. Each neuron can either be active or not. The input received by the neuron at a given time can therefore be represented by a vector $\vec{I}$ of zeros and ones where I_n is the activity of neuron n.

The synapses connecting the input neurons and the output neuron may have different strengths. They are represented by a vector of weights $\vec{W}$ where W_n is the strength of the synapse connecting neuron n to the output neuron.

The output neuron is active if the total input it receives is larger than a threshold θ.



The output neuron has the task of associating a set of P input patterns $\vec{I}_p$ with a desired output r_p. The parameters $\vec{W}$ and θ may be adjusted so as to solve this task.

1. Give an expression of the total input received by the output neuron when a given pattern $\vec{I}_p$ is presented.

The total input received by the neuron is the weighted sum over all input neurons, which can be written as a scalar product between two vectors

$$\sum_{j=1}^{N} I_{p,j} W_j = \vec{I}_p \cdot \vec{W}.$$

2. Write a condition on the input patterns $\vec{I}_p$, the desired outputs r_p and the parameters $\vec{W}$ and θ such that the task is solved.

The task is solved if the perceptron is able to categorize each input pattern correctly. Each output values r_p requires a relation between the input and the threshold as follows

$$\forall p, \quad \begin{cases} r_p = 1 \Rightarrow \vec{I}_p \cdot \vec{W} > \theta, \\ r_p = 0 \Rightarrow \vec{I}_p \cdot \vec{W} < \theta. \end{cases}$$

3. For convenience, consider an imaginary input $I_0 = 1$ which is always turned on and rewrite the condition as a function of $(I_0, \vec{I}_p)$ and $(W_0, \vec{W})$ where $W_0 = -\theta$.

In order to simplify the condition, the threshold can be modelled as a constant input. To do so, the vectors $\vec{I}_p$ and $\vec{W}$ are appended with a first entry $I_0 = 1$ and $W_0 = -\theta$ respectively, such that

$$\sum_{j=1}^{N} I_{p,j} W_j > \theta \Leftrightarrow \sum_{j=1}^{N} I_{p,j} W_j - \theta \times 1 > 0 \Leftrightarrow \sum_{j=0}^{N} I_{p,j} W_j > 0.$$

The condition becomes

$$\forall p, \quad \begin{cases} r_p = 1 \Rightarrow \vec{I}_p \cdot \vec{W} > 0, \\ r_p = 0 \Rightarrow \vec{I}_p \cdot \vec{W} < 0. \end{cases}$$

4. Find a set of input patterns $\vec{J}_p$ such that the condition is equivalent to "for all $p, \vec{J}_p \cdot \vec{W} > 0$".

To reduce the condition to a single equation whose result is always positive, the dot product $\vec{I}_p \cdot \vec{W}$ can be multiplied by a scalar of the same sign, expressed as a function of r_p . The goal is to get a variable s_p such that

$$\forall p, \quad \begin{cases} r_p = 1 \Rightarrow s_p = 1, \\ r_p = 0 \Rightarrow s_p = -1. \end{cases}$$

This can be obtained by introducing $s_p = 2r_p - 1$. Then

$$\forall p, \; s_p \vec{I}_p \cdot \vec{W} > 0.$$

Equivalently, this can be obtained by introducing the vectors $\vec{J}_p = (2r_p - 1)\vec{I}_p$.

5. What does it mean to have a network which is able to generalise after learning?

Generalisation is the ability to correctly classify a new input (i.e. not previously seen) after the network has been trained.

2 Perceptron algorithm

Now that we have better characterized the inputs to the perceptron, we need to train it in order to correctly associate inputs to their respective output. To do so, we use the perceptron algorithm

- randomly pick an input pattern $\vec{J}_p$
- if $\vec{J}_p \cdot \vec{W} > 0$, pick a new input pattern
- if $\vec{J}_p \cdot \vec{W} < 0$, perform a learning step $\vec{W}(t+1) = \vec{W}(t) + \epsilon \vec{J}_p$.

6. Explain why this algorithm is an implementation of "supervised" learning.

The correct output is provided by an external "teaching" signal which drives learning in the right direction. The information about the correct output is contained in the vector $\vec{J}_p$ through the appearance of r_p in its expression.

We will show that if there exists a solution $\vec{W}^*$, then this algorithm necessarily finds a solution. For this, we consider

$$\cos(\alpha(t)) = \frac{\vec{W}(t) \cdot \vec{W}^*}{\|\vec{W}(t)\| \|\vec{W}^*\|},$$

with $\alpha(t)$ the angle between $\vec{W}(t)$ and $\vec{W}^*$.

7. Introduce $l = \min_p(\vec{J}_p \cdot \vec{W}^*) > 0$ and find a lower bound on $\vec{W}(t+1) \cdot \vec{W}^*$ given $\vec{W}(t) \cdot \vec{W}^*$. Considering $\vec{W}(0) = 0$, deduce a lower bound on $\vec{W}(t) \cdot \vec{W}^*$.

From the perceptron algorithm, we have that

$$\vec{W}(t+1) \cdot \vec{W}^* = (\vec{W}(t) + \epsilon \vec{J}_p) \cdot \vec{W}^* = \vec{W}(t) \cdot \vec{W}^* + \epsilon \vec{J}_p \cdot \vec{W}^* \geq \vec{W}(t) \cdot \vec{W}^* + \epsilon l.$$

Applying the same reasoning as above, by recurrence this leads to

$$\vec{W}(t) \cdot \vec{W}^* \geq \vec{W}(0) \cdot \vec{W}^* + \epsilon l t = \epsilon l t.$$

8. Introduce $L = \max_p(\|\vec{J}_p\|^2)$ and find an upper bound on $\|\vec{W}(t+1)\|^2$ given $\|\vec{W}(t)\|^2$. Deduce an upper bound on $\|\vec{W}(t)\|^2$.

Again, from the perceptron algorithm, we have that

$$\|\vec{W}(t+1)\|^2 = \|\vec{W}(t) + \epsilon \vec{J}_p\|^2 = \|\vec{W}(t)\|^2 + \epsilon^2 \|\vec{J}_p\|^2 + 2\epsilon \vec{J}_p \cdot \vec{W}(t) \leq \|\vec{W}(t)\|^2 + \epsilon^2 L.$$

Applying the same reasoning as above, by recurrence this leads to

$$\|\vec{W}(t)\|^2 \leq \|\vec{W}(0)\|^2 + t\epsilon^2 L = t\epsilon^2 L \Rightarrow \|\vec{W}(t)\| \leq \sqrt{t\epsilon^2 L}.$$

9. Find a lower bound on $\cos(\alpha(t))$.

Using the two previous questions, we have that

- $\vec{W}(t) \cdot \vec{W}^* \geq \epsilon l t$,
- $\|\vec{W}(t)\|\|\vec{W}^*\| \leq \sqrt{t\epsilon^2 L}\|\vec{W}^*\| \Rightarrow \dfrac{1}{\|\vec{W}(t)\|\|\vec{W}^*\|} \geq \dfrac{1}{\sqrt{t\epsilon^2 L}\|\vec{W}^*\|}$,

and therefore,

$$\cos(\alpha(t)) = \frac{\vec{W}(t) \cdot \vec{W}^*}{\|\vec{W}(t)\|\|\vec{W}^*\|} \geq \frac{t\epsilon l}{\sqrt{t\epsilon^2 L}} = \sqrt{t}\frac{l}{\sqrt{L}\|\vec{W}^*\|}.$$

10. Explain why the algorithm necessarily finds a solution.

The result obtained in the previous question hold each time a learning step is performed. In this case, as $\cos(\alpha(t)) \leq 1$, it imposes that

$$t \leq \frac{L\|\vec{W}^*\|^2}{l^2} = \text{constant}.$$

therefore, the algorithm necessarily stops after a finite number of updates. When the algorithm stops, all patterns are correctly classified.

3 Simple Hopfield network

We consider a simple Hopfield network with 3 neurons x_1, x_2 and x_3 and 2 stored patterns $\xi^1 = [+1, -1, +1]$ and $\xi^2 = [-1, +1, -1]$. The update of the Hopfield network is asynchronous and is given by:

$$h_i = \sum_j w_{ij} x_j \Rightarrow x_i = \begin{cases} +1 & \text{if } h_i \geq 0 \\ -1 & \text{if } h_i < 0 \end{cases}$$

11. Compute the connection matrix W.

The elements of the connection matrix are given by

$$w_{ij} = \frac{1}{2}\sum_{\mu=1}^{2} \xi_i^\mu \xi_j^\mu,$$

when $i \neq j$ and 0 for w_{ii}. The connection matrix is then

$$W = \begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}$$

12. What is the evolution of the state $x_1 = 1$, $x_2 = 1$ and $x_3 = 1$?

(a) if we update x_1: $h_1 = w_{12}x_2 + w_{13}x_3 = -1 * 1 + 1 * 1 = 0$, hence $x_1 \rightarrow 1$. This doesn't change the state of the network.

(b) if we update x_1: $h_3 = w_{32}x_2 + w_{31}x_1 = -1 * 1 + 1 * 1 = 0$, hence $x_3 \rightarrow 1$. This doesn't change the state of the network.

(c) if we update x_1: $h_2 = w_{21}x_1 + w_{23}x_3 = -1 * 1 + -1 * 1 = -2$, hence $x_2 \rightarrow -1$. This changes the network state.

The only possible different state is $[x_1, x_2, x_3] = [+1, -1, +1]$. Doing the same computation we find that this state is stable (i.e. no update changes the state of any of the neurons).

13. What are the stable states of the network?

To find those states we have to find the evolution of the network for any starting state:

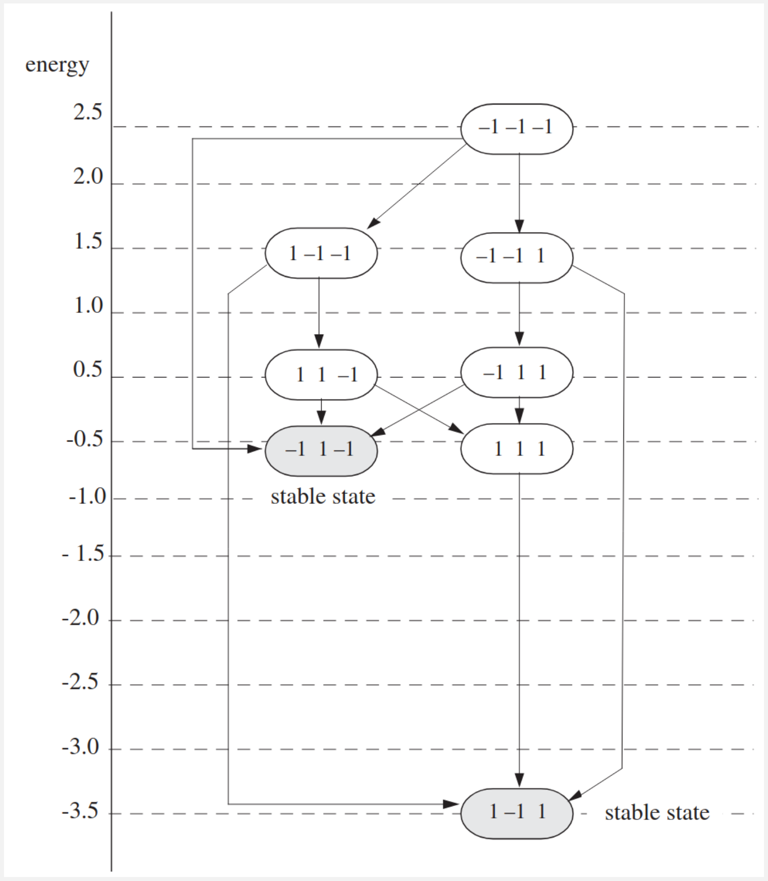


Figure 1: From Fig. 13.12. of "Neural Networks: A Systematic Introduction" by Raùl Rojas