University of BRISTOL

McCulloch-Pitts Neurons and Hopfield Networks

Dr Amelia Burroughs

bristol.ac.uk

Learning Objectives

Understand simple threshold logic units and what they are useful for

To gain an understanding of unsupervised learning and Hebbian learning rules

To understand the Hopfield network architecture and how it can be used for pattern completion.

McCulloch-Pitts neuron model

The McCulloch Pitts neuron model, or Threshold Logic Unit, was introduced in 1943 by Warren McCulloch and Walter Pitts as a computational model of a neuronal network.

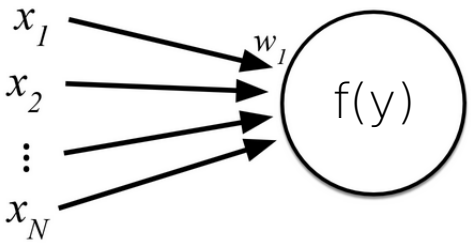
It was 1943 – what did they think to include?

They assumed that:

- neurons are joined to each other with connections of variable strength
- the inputs from other neurons are added up in the soma
- the summation of inputs at the soma determines the activity of the neuron in a non-linear way
- neurons tend to ignore input up to some threshold value before responding strongly

They tried to include these properties in their model neurons

The McCulloch-Pitts Neuron



x_i is the i^{th} input neuron's activity

w_i is the strength of the synaptic weight from neuron i

The 'net input' to the neuron is:

$y = \sum i\ w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$

This is the 'weighted sum' of input activation

The output of the neuron is a function of y: f(y)

The McCulloch-Pitts Neuron

Artificial neurons, of the sort used in artificial intelligence, are described by a single dynamical variable. The value for the neuron in the last slide is determined by the weighted input from the other neurons:

$$\text{output} = \phi\left(\sum_i w_i\, x_i - \theta\right)$$

φ is an activation (or transfer) function, θ is a threshold and the w ij are the connection strengths weighting the inputs from the other neurons. The McCulloch-Pitts neuron was the first example of an artificial neuron and had a step function for φ:

$$f(y) = \begin{cases} 1 & \sum_i w_i\, x_j > \theta \\ -1 & \text{otherwise} \end{cases}$$

The McCulloch-Pitts Neuron

$$f(y) = \begin{cases} 1 & \sum_i w_i \; x_j > \theta \\ -1 & \text{otherwise} \end{cases}$$

This means that the neuron has two states. It is in the on state, x i = 1 if the weighted input exceeds a threshold θ and in the off state, x i = −1 if it doesn't.

'on' corresponds to rapid spiking and 'off' to spiking at a much lower rate.

The w_i connection strengths are like the synapse strengths, a positive w ij is an excitatory synapse and negative, an inhibitory synapse.

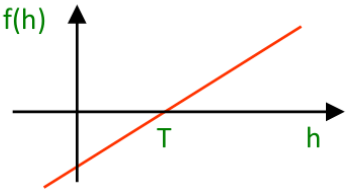
Each input neuron in this model can be either negative and positive: there is no restriction that says that w_i must always have the same sign.

This is different from real neurons where all the outgoing synapses from a given neuron are either excitatory or inhibitory

Common Activation Functions

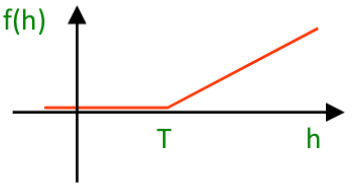
linear, e.g.

$f(h) = h - T$



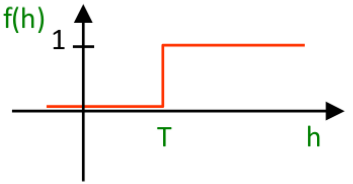
threshold-linear, e.g.

$f(h) = \begin{cases} h-T & \text{if } h>T \\ 0 & \text{if } h<T \end{cases}$



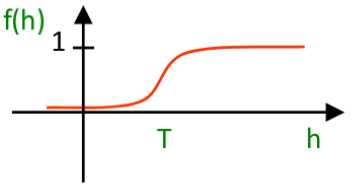
threshold logic function

$f(h) = \begin{cases} 1 & \text{if } h>T \\ 0 & \text{if } h<T \end{cases}$



sigmoidal

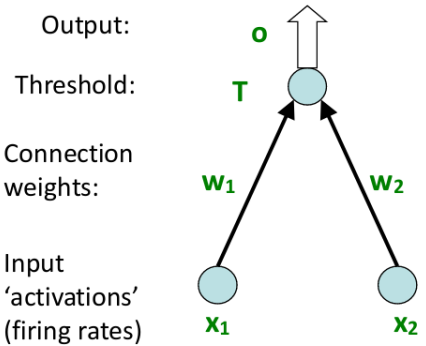
$f(h) = \dfrac{1}{1+\exp(-(h-T))}$



We saw something similar to the logic threshold function or 'step' function in the previous slide with 1 or -1, instead of 1 or 0.
T = threshold in this case.

Implications of using the ‘weighted sum’

The weighted sum of input activations is used to calculate the net input to our artificial neuron. Take this example:

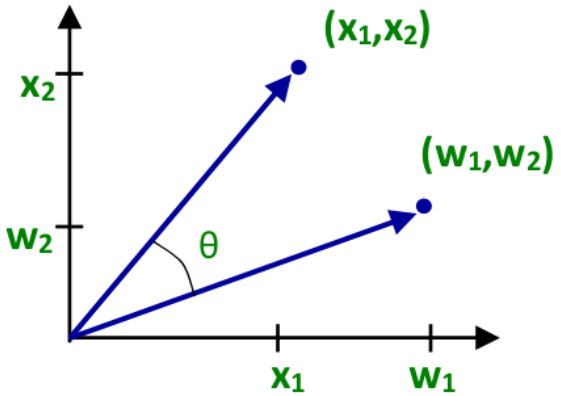


Output = f(y), $y = w_1x_1 + w_2x_2$

Here, the input activations and the connection weights can be thought of as vectors $\underline{x}$ and $\underline{w}$.

And the ‘weighted sum’, $y = w_1x_1 + w_2x_2$, is also known as the ‘dot product’ of $\underline{x}$ and $\underline{w}$ ($\underline{w}.\underline{x}$), which depends on the angle θ between them.

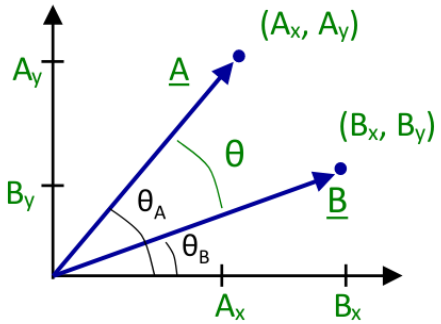
Implications of using the ‘weighted sum’



$w.x = |w|\,|x| \cos(\theta)$

This means that if the total amount of input activation and connection weights are limited, e.g. |w|= 1, |x|=1, the maximum net input y (and therefore output firing rate f(y)) occurs when the patterns of input activations and connection weights **match**.

The vector dot product



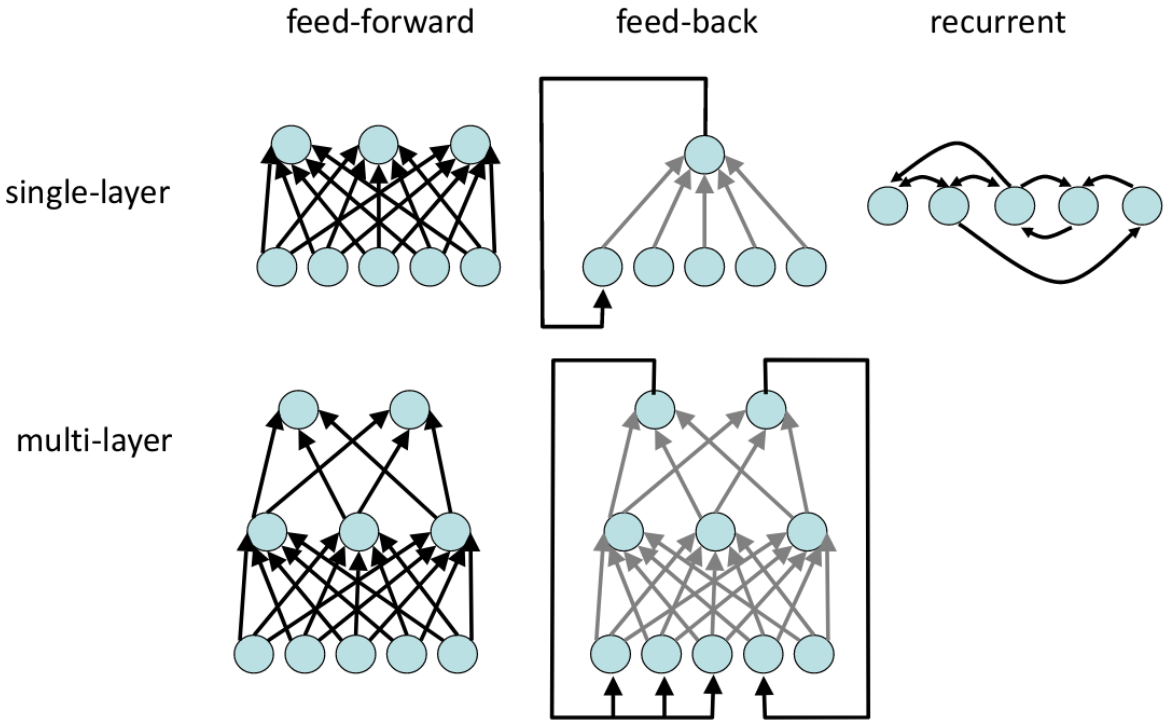
Definition: $\underline{A}.\underline{B} = |\underline{A}|\ |\underline{B}|\ \cos(\theta)$

So:
$$\begin{aligned}\underline{A}.\underline{B} &= |\underline{A}|\ |\underline{B}|\ \cos(\theta_A-\theta_B)\\ &= |\underline{A}|\ |\underline{B}|\ (\cos(\theta_A)\cos(\theta_B) + \sin(\theta_A)\sin(\theta_B))\\ &= |\underline{A}|\cos(\theta_A)\ |\underline{B}|\cos(\theta_B) + |\underline{A}|\sin(\theta_A)\ |\underline{B}|\sin(\theta_B)\\ &= A_x B_x + A_y B_y\end{aligned}$$

$A_x = |\underline{A}|\cos(\theta_A),\quad A_y = |\underline{A}|\sin(\theta_A)$
$B_x = |\underline{B}|\cos(\theta_B),\quad B_y = |B|\sin(\theta_B)$

More generally: $\underline{A}.\underline{B} = \sum_i A_i B_i$

Networks of McCulloch-Pitts neurons



But why are they useful?

When they were working, at the dawn of the age of electronic computers, McCulloch and Pitts believed that their neurons might form the natural unit in computer circuits

- they thought they might perform the role actually played by logical circuits.

It is still not clear if the artificial neuron is or isn't the natural unit of computation in
silicon since they are a component in, for example, deep learning networks.

In fact, there are two major applications of McCulloch-Pitts neurons: the perceptron and the Hopfield network.

These two applications add a rule for changing the connection strength to the original McCulloch-Pitts neuron.

- In this way, McCulloch-Pitts neurons can be used for learning!

McCulloch-Pitts neurons and learning

McCulloch-Pitts neurons can be used for learning. But what does that mean?

The problem: Finding connection weights that mean that the network can do something useful = 'learning'

The solution: Include experience-dependent learning rules that modify connection weights and improve the output

There are two forms of learning generally applied to these networks:

1) Unsupervised learning

- there is no teacher and the network is not informed (via feedback connections) about right or wrong outputs

2) Supervised learning

- the network received varying degrees of supervision:
  - fully supervised (each example includes correct output)
  - occasional reward or punishment (reinforcement learning)
  - evolution / genetic algorithms

Unsupervised learning

Hopfield Networks

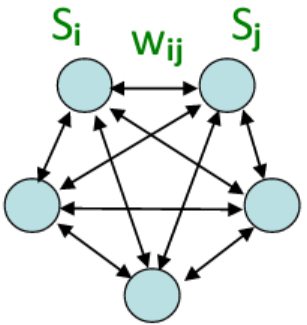
A Hopfield network is a fully connected recurrent network of McCulloch-Pitts neurons proposed by John Hopfield in 1982.

Connection weights between neurons are symmetric ($w_{ij}=w_{ji}$)

Connectivity is all-to-all

Units can be active (+1) or inactive (-1)

Forms a basic model of associative memory rec



Hopfield Networks

You can evolve the network state by synchronous or asynchronous updates of the connection weights between neurons:

Synchronous updates: All units in the network are updated at the same time

Asynchronous updates: Only one unit is updated at a time

Learning

During learning a pattern of activation is imposed (an attractor state) and a Hebbian learning rule is used to change connection weights

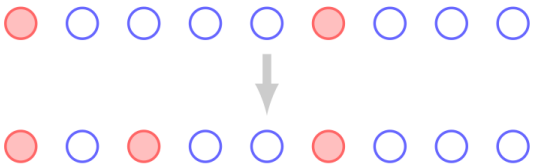
Recall

You present a similar pattern of activation as during the learning. Activation through the network is changed according to the sign of the input and the original pattern is recovered

Hopfield Networks

The idea is that this is a model for auto-associative memory. Auto- associative memories are patterns representing memories along with some dynamics that complete partial patters. Imagine a sequence of McCulloch-Pitts neurons:

where the filled circles correspond to on. Recall occurs when the network is presented with a partial pattern and evolves into the complete patterns.



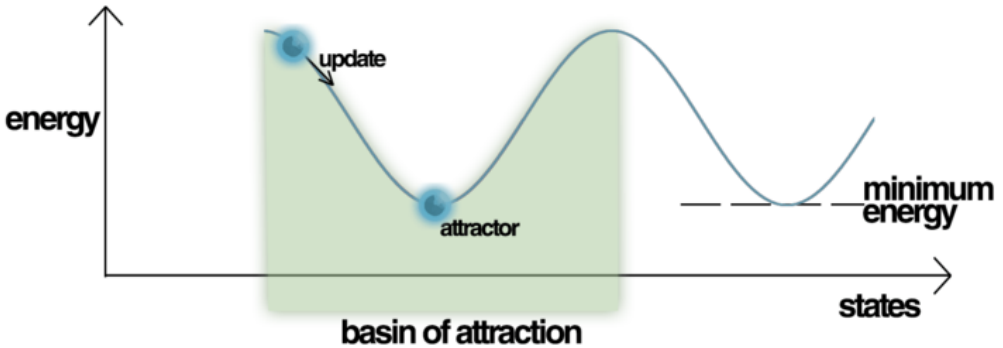
The Hopfield network is intended to model the recall, and learning, of memories in an auto-associative network.

‘Energy’

One way to think about this is to note that there is an ‘energy’ associated with a Hopfield network:

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

If you update a node, you will reduce the energy (or it stays the same).



https://en.wikipedia.org/wiki/Hopfield_network

Hopfield networks

To support a pattern of activity, connections should be positive between units in the same state (i.e. 1,1 or -1,-1) and negative between units in different states (1,-1 or -1,1)

The ‘energy’ of the system is how much this is not true.

The update rule changes each unit’s activity to reduce the overall energy until the network ends up in a stable state from which it cannot be reduced further.

The learning rule sets the weights so that to-be-remembered patterns of activity are stable states (aka ‘attractor states’).

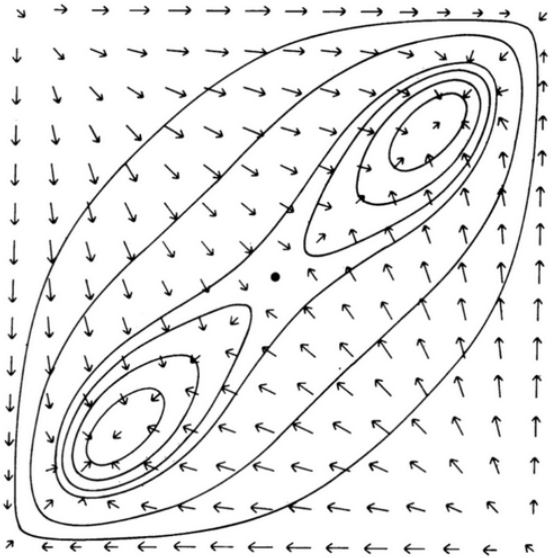


FIG. 3. An energy contour map for a two-neuron, two-stable-state system. The ordinate and abscissa are the outputs of the two neurons. Stable states are located near the lower left and upper right corners, and unstable extrema at the other two corners. The arrows show the motion of the state from Eq. 5. This motion is not in general perpendicular to the energy contours. The system parameters are $T_{12} = T_{21} = 1$, $\lambda = 1.4$, and $g(u) = (2/\pi)\tan^{-1}(\pi\lambda u/2)$. Energy contours are 0.449, 0.156, 0.017, −0.003, −0.023, and −0.041.

How to create local minima?

We can imprint desired attractor states into the synaptic weights using a learning rule:

$$w_{ij} = \frac{1}{N} \sum_{a} x_i^a x_j^a$$

where N is the number of patterns to be stored, and a indexes the patterns.

This learning rule is inspired by an experimental neuroscience finding: ‘Hebbian plasticity’: neurons that fire together wire together.

Synapses are plastic, meaning that they are able to change their strength, over both the short and long term.

Synaptic plasticity usually refers to the long-term changes in synapse strength, a long term increase in synaptic strength is called long term potentiation of LTP, a decrease is called long term depression or LTD.

Hebbian Plasticity

Synapses respond to their pre- and post-synaptic activity, so that the changes depend on the behavior of the pre- and post-synaptic neurons.

In 1949, Hebb said:

“Let us assume that the persistence or repetition of a reverberatory activity (or ‘trace’) tends to induce lasting cellular changes that add to its stability. [. . .] When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.”

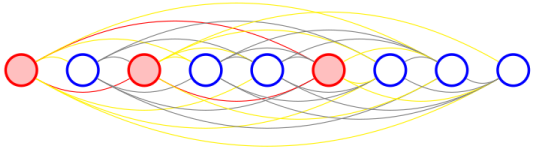
If one neurons tends to cause another to fire, the synapse from the first to the second will get stronger.

In artificial neurons that lack spiking dynamics and instead have a continuous state or rate variable Hebbian plasticity is often applied as a rule that strengthens synapses between neurons that are active at the same time.

Hopfield networks and Hebbian plasticity

During learning the patterns are presented.

Plastic changes are made to the synapse strength according to a simple correlation based Hebbian plasticity rule



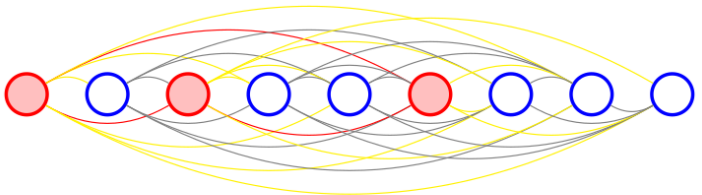
Typically there will be a large increase in the connection strength between two neurons that are active at the same time, a tiny increase for pairs neurons that are inactive at the same time and a medium size decrease for pairs of neurons where one is active and one inactive at the same time.

Learning in a Hopfield Network

The idea is that after learning the pattern {0, 2, 5}

the connections between these nodes will be strong, so if the network gets presented with a partial pattern:

the value $r_2 = w_{02} + w_{52}$ will be larger than the threshold and the subsequent dynamics will switch neuron 2 on

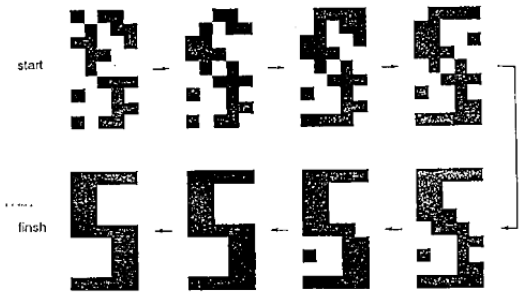


However, in this network, if a different initial set of neurons are activated, the activity will die away because the r_i will all be sub-threshold.

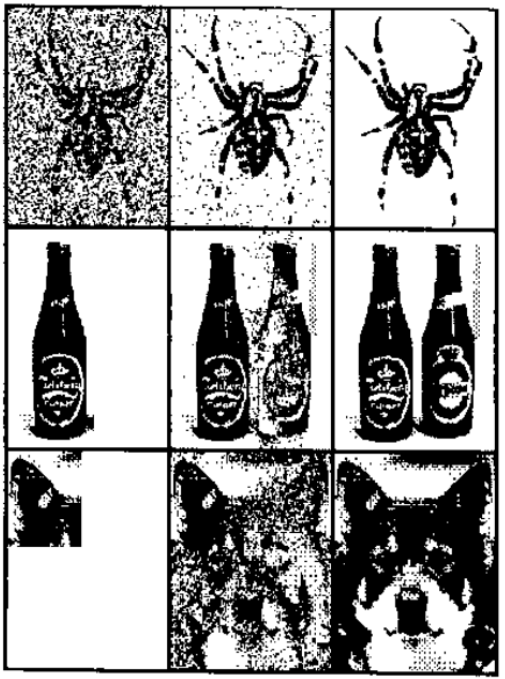
Examples of Hopfield networks

A 5x9 network storing 8 patterns

Figure 6.3 The training set for the Hopfield network. BeJ, 143

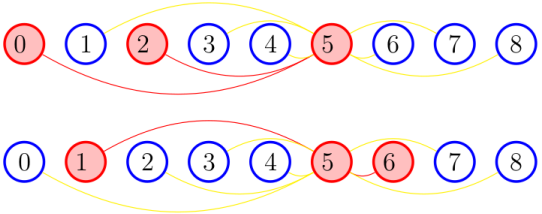


Retrieval in a 130x180 network



Interference

When many patterns are stored it is likely that there will be interference between them.



Neuron 5 is involved in two patterns and therefore some of its connections strengthening for one pattern and weaken for the other. If these strengthening and weakening effects are similar in size then it is unlikely either pattern will be retrieved.

For larger networks some overlap is possible, but too much overlap prevents retrieval.

The capacity is proportional to the number of neurons. Capacity is also larger if there is sparseness.

Hopfield networks summary

- Activation rule:
  - If net input is greater than zero, unit gets an activation of 1; otherwise activation is -1
  - There is random, asynchronous update of activations
- Architecure
  - Symmetrically connected recurrent network
- Hebbian learning
  - For each training pattern the states of the units are set to corresponding elements of pattern
  - Connection weights are changed in proportion to the product of pre- and post-synaptic states
- Desirable features
  - Attractor dynamics: guaranteed convergence to an attractor state
  - Pattern completion
- Undesirable features
  - Spurious attractors
  - Limited storage capacity

Conclusions

McCulloch-Pitts neurons are described by one dynamic variable:

- The weighted sum of their inputs

Very simple networks of McCulloch Pitts neurons can perform ‘learning’

Learning occurs by changes in connection weights

Changes in connection weights are determined by both the pre- and post-synaptic activity : this is likened to Hebbian plasticity

Hebbian plasticity: cells that fire together wire together

Hebbian plasticity can be incorporated into Hopfield networks so that they can learn and store patterns

- This is an example of auto-associative memory