

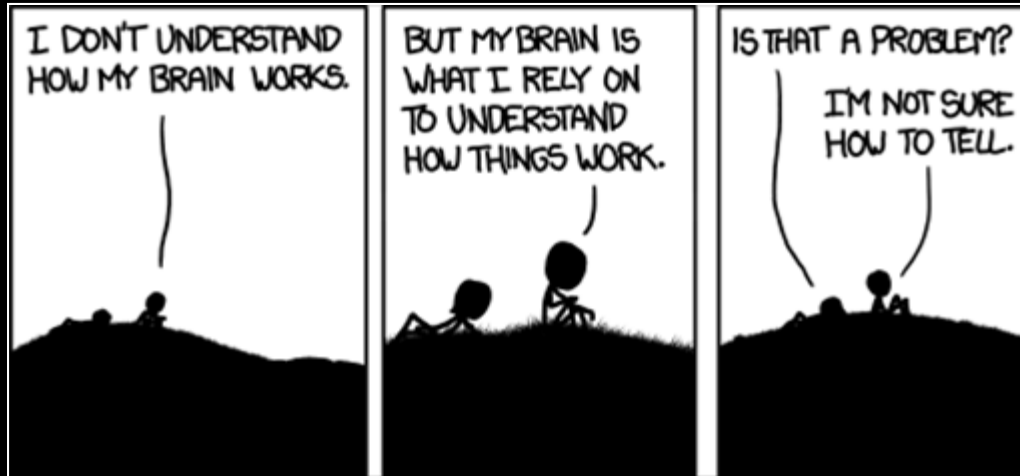
# Computer modeling of complex systems



Lecture 6 Neural Networks

Lab 6 Hopfield Model

# Brain?



# Consciousness?



Marvin Minsky

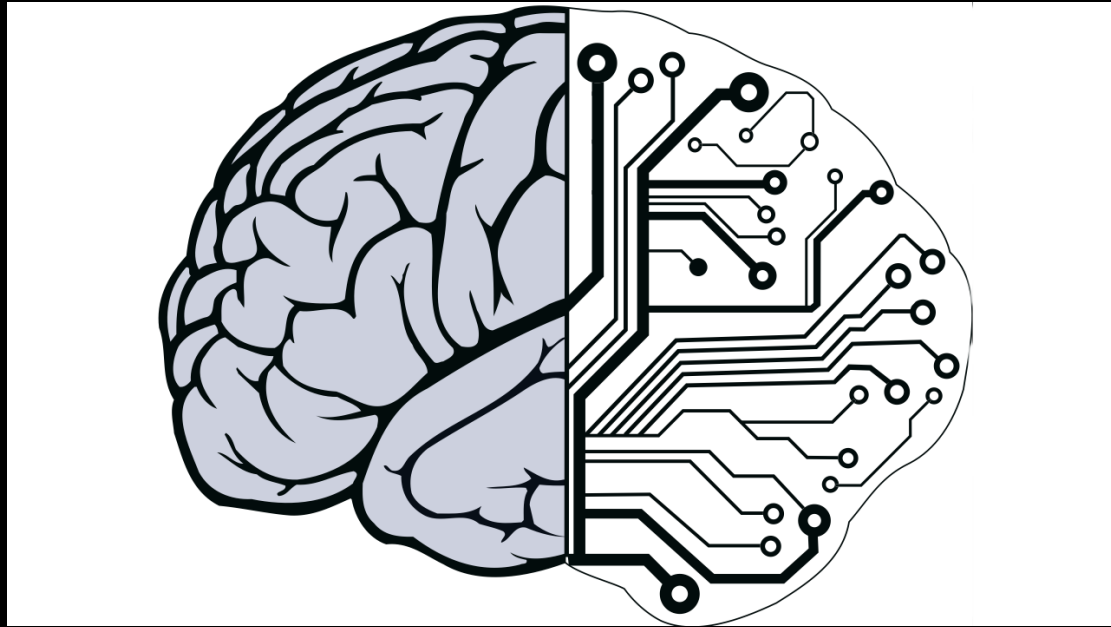
„What is commonly attributed to consciousness is mythical. In fact, some machines are potentially more conscious than are people”



Roger Penrose

„Conscious mind cannot work like a computer, even though much of what is actual involved in mental activity can do so”

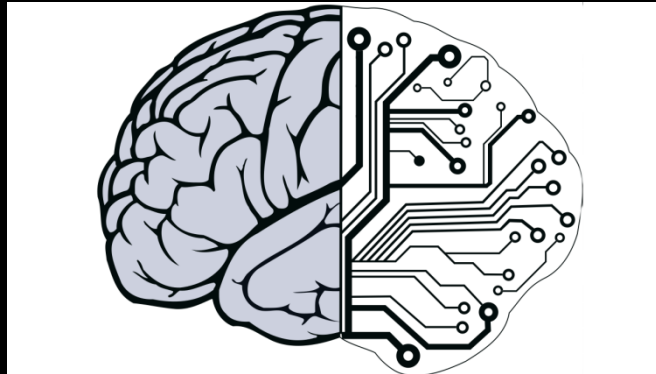
# Brain vs computer



*“ If the human brain were so simple that we could understand it, we would be so simple that we couldn’t.”*

*- Emerson M. Pugh*

# Brain vs computer



200 billion ( $200 \times 10^9$ ) neurons,  
32 trillion ( $32 \times 10^{12}$ ) synapses

Element size:  $10^{-6}$  m

Energy use: 25W

Processing speed: 100 Hz

Parallel, Distributed

Fault Tolerant

Learns: Yes

Intelligent/Conscious: Usually

1 billion bytes RAM, trillions of bytes  
on disk

Element size:  $10^{-9}$  m

Energy watt: 30-90W (CPU)

Processing speed:  $10^9$  Hz

Serial, Centralized

Generally not Fault Tolerant

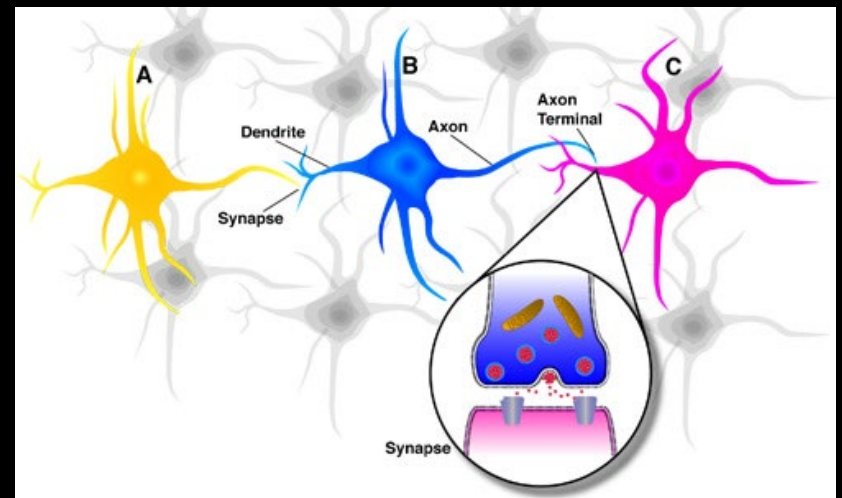
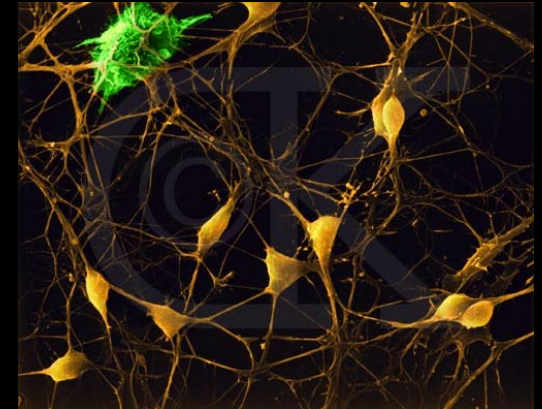
Learns: Some

Intelligent/Conscious: Generally No

# Neurons in the Brain

Although heterogeneous, at a low level the brain is composed of neurons

- A neuron receives input from other neurons (generally thousands) from its synapses
- Inputs are approximately summed
- When the input exceeds a threshold the neuron sends an electrical spike that travels that travels from the body, down the axon, to the next neuron(s)



# Neural networks - history

- 1933 - Edward Thorndike: human learning consists in the strengthening of some (then unknown) property of neurons
- 1949 - Donald Hebb: it is specifically a strengthening of *connections* between neurons in the brain that accounts for learning

„When an axon of cell A is near enough to excite 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.”

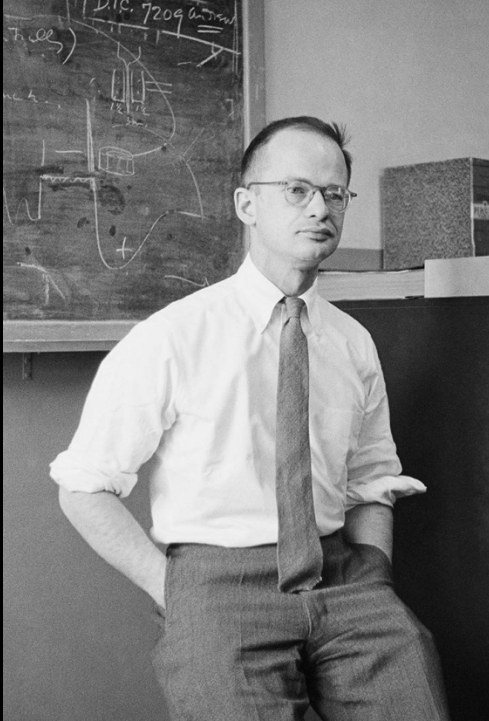
D. Hebb, The Organization of Behavior, 1949

or "Neurons that fire together wire together."

(Hebb's Law)



# Walter Pitts and Warren McCulloch



1923-1969

„A logical calculus of the ideas  
immanent in nervous activity”  
*Bulletin of Mathematical  
Biophysics*, 5:115-133.



1898-1969

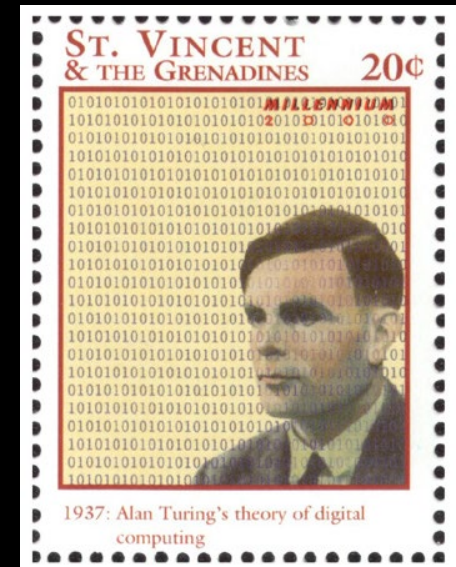
- each neuron in the brain is a simple digital processor and the brain as a whole is a form of computing machine



# Training networks

- network of initially randomly connected artificial neurons could be "trained" to perform a given task by means of a process that renders certain neural pathways effective and others ineffective.
- neurons and their interconnections can be simulated on a computer
- for the training of his networks, Turing suggested a sort of genetic algorithm:

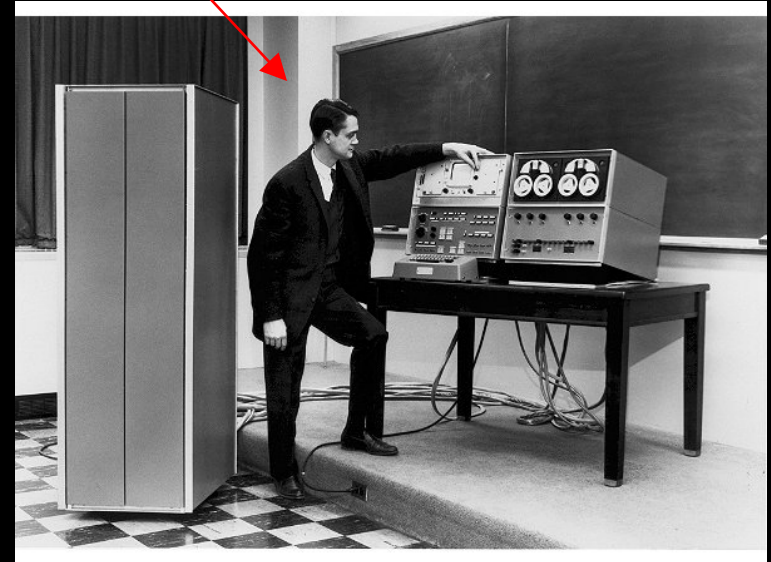
„There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value. The remarkable success of this search confirms to some extent the idea that intellectual activity consists mainly of various kinds of search”



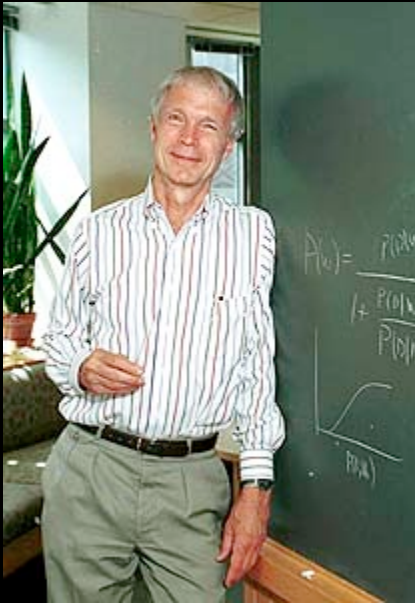
A. Turing: Intelligent Machinery, 1948

# Belmont Farley and Wesley Clark

- the first computer simulations of small neural networks, training the networks containing 128 neurons to recognise simple patterns
- they discovered that the random destruction of up to 10% of the neurons in a trained network does not affect the network's performance



# Associative memory model: Hopfield network



John Hopfield

- inspired by the Ising model.

$$x_i = \pm 1$$



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

all spins can be coupled

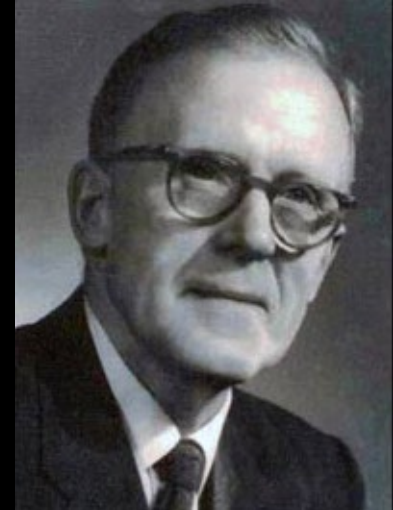
J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *PNAS*, **79**, 2554, 1982

# Training the network: Hebb's rule

"Neurons that fire together, wire together. Neurons that fire out of sync, fail to link".

When learning  $n$  binary (spin) patterns  $\mathbf{x}^\mu$ :

$$\mathbf{w} = \frac{1}{n} \sum_{\mu=1}^n \mathbf{x}^\mu \otimes \mathbf{x}^\mu - \mathbf{1}$$



If the bits corresponding to neurons  $i$  and  $j$  are equal in pattern, then the product will be positive. This would, in turn, have a positive effect on the weight  $w_{ij}$  and the values of  $i$  and  $j$  will tend to become equal. The opposite happens if the bits corresponding to neurons  $i$  and  $j$  are different.

# Updating the network

The energy of  $i$ th spin can be written as:

$$E_i = -\frac{1}{2} x_i \left( \sum_j w_{ij} x_j \right) = -\frac{1}{2} x_i h(i)$$

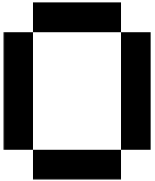

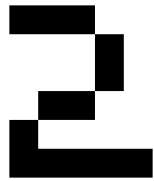
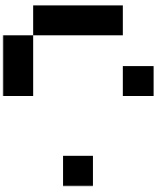
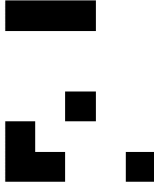

where  $h(i)$  is the field acting on the spin. If the product  $x_i h(i)$  is negative, then the field would try to flip the spin:

$$\mathbf{x}(t+1) = \text{sgn}(\mathbf{w} \cdot \mathbf{x}(t))$$

- patterns that the network uses for training become attractors of the system.
- repeated updates eventually lead to convergence to one of these states.
- sometimes the network will converge to spurious patterns, that are different from the training patterns, but also (local) energy minima
- the capacity of the network is  $\sim 0.138$  pattern per node

# Your task

**Task # 1** Try to model a simple 6x5 Hopfield network using the numbers as the stored patterns – use the three patterns below (zero, one and two) to train the network

<pre>zero = np.matrix([   0, 1, 1, 1, 0,   1, 0, 0, 0, 1,   1, 0, 0, 0, 1,   1, 0, 0, 0, 1,   1, 0, 0, 0, 1,   1, 0, 0, 0, 1,   0, 1, 1, 1, 0 ])</pre> 	<pre>one = np.matrix([   0, 1, 1, 0, 0,   0, 1, 1, 0, 0,   0, 0, 1, 0, 0,   0, 0, 1, 0, 0,   0, 0, 1, 0, 0,   0, 0, 1, 0, 0,   0, 0, 1, 0, 0,   0, 0, 1, 0, 0 ])</pre> 	<pre>two = np.matrix([   1, 1, 1, 0, 0,   0, 0, 0, 1, 0,   0, 0, 0, 1, 0,   0, 0, 0, 1, 0,   0, 1, 1, 0, 0,   1, 0, 0, 0, 0,   1, 1, 1, 1, 1 ])</pre> 
<pre>noisy0 = np.matrix([   0, 1, 1, 1, 0,   1, 0, 0, 0, 0,   1, 0, 0, 0, 1,   0, 0, 0, 0, 0,   0, 0, 0, 0, 0,   0, 0, 0, 0, 0,   0, 0, 1, 0, 0 ])</pre> 	<pre>noisy2 = np.matrix([   1, 1, 1, 0, 0,   0, 0, 0, 0, 0,   0, 0, 0, 0, 0,   0, 0, 1, 0, 0,   1, 0, 0, 0, 0,   1, 1, 0, 0, 1,   1, 1, 0, 0, 1 ])</pre> 	<pre>noisy2b = np.matrix([   1, 1, 1, 0, 0,   0, 0, 0, 1, 0,   0, 0, 0, 1, 0,   0, 0, 0, 0, 0,   0, 0, 0, 0, 0,   0, 0, 0, 0, 0,   0, 0, 0, 0, 0 ])</pre> 

Next, use the three other patterns (noisy zero, noisy 2 and another noisy 2) as starting configurations of the network (these patterns were obtained by taking the original patterns and erasing some bits). What are the outcomes of the network dynamics in each of these three cases? How is the energy of the network changing?

# Task # 2

While analysing pattern noisy2b in the previous task you probably noticed that it converges towards a spurious attractor. This is one of the problems of Hopfield network. The number of spurious attractors (or 'hallucinations', as they are sometimes called) can be significantly reduced if – contrary to task #1 – the network is updated asynchronously. In asynchronous update the spins are picked randomly, one by one, and they are flipped if this leads to decrease of their energy

- Modify your code to perform such an update and track the evolution of the pattern noisy2b. What is the final pattern to which it converges? Does it always converge to the same pattern?
- Plot the evolution of the energy vs time corresponding to such an evolution.

# Extra task

Let's come back to the network from Task #1 and the problem of correct recognition of characters: 0, 1 and 2. Can we increase the efficiency of recognition of these three characters by increasing the number of patterns in network's memory? If so, then what kind of extra patterns it is best to add? Please demonstrate this point with a working example.

