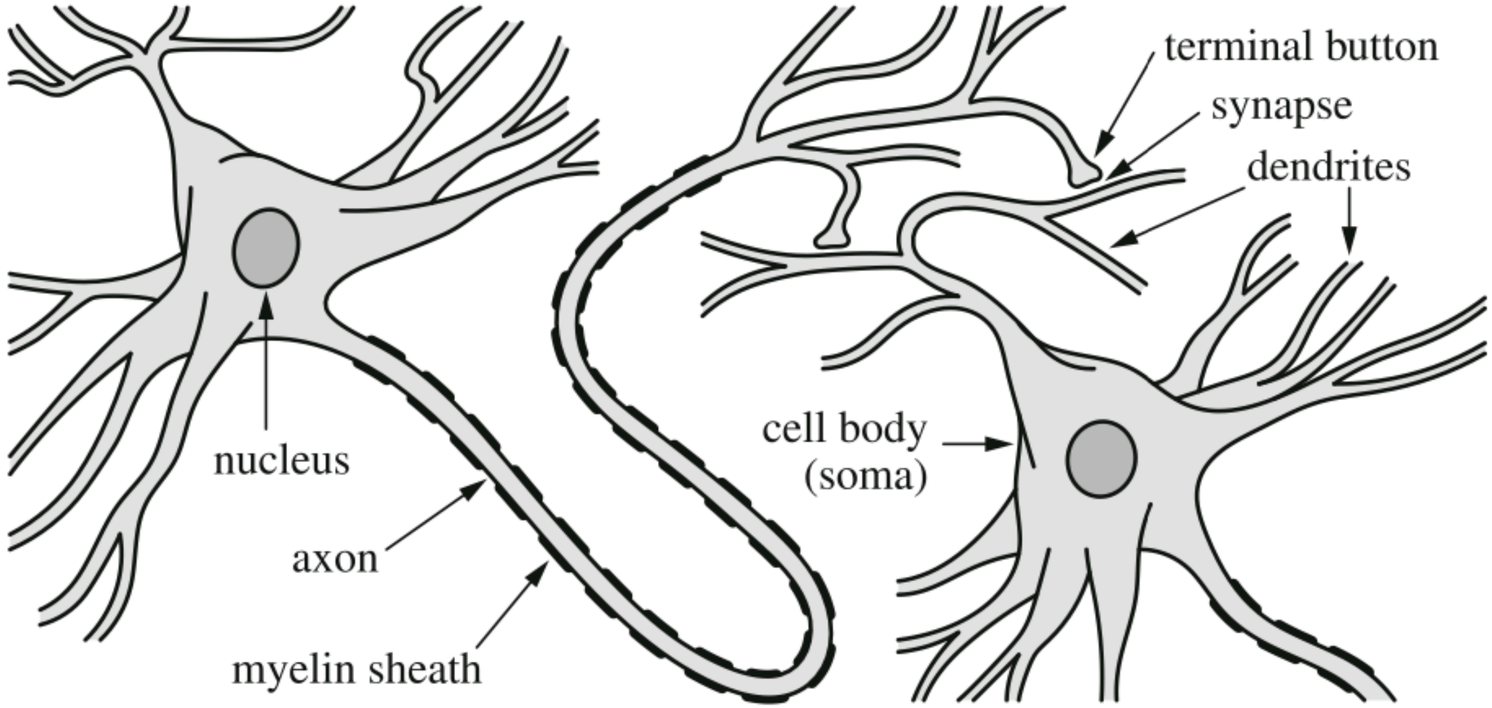
FACULTY OF PHYSICS
UNIVERSITY OF WARSAW

Neural Networks and Memory

Daniel Matoz

What is a neuron?

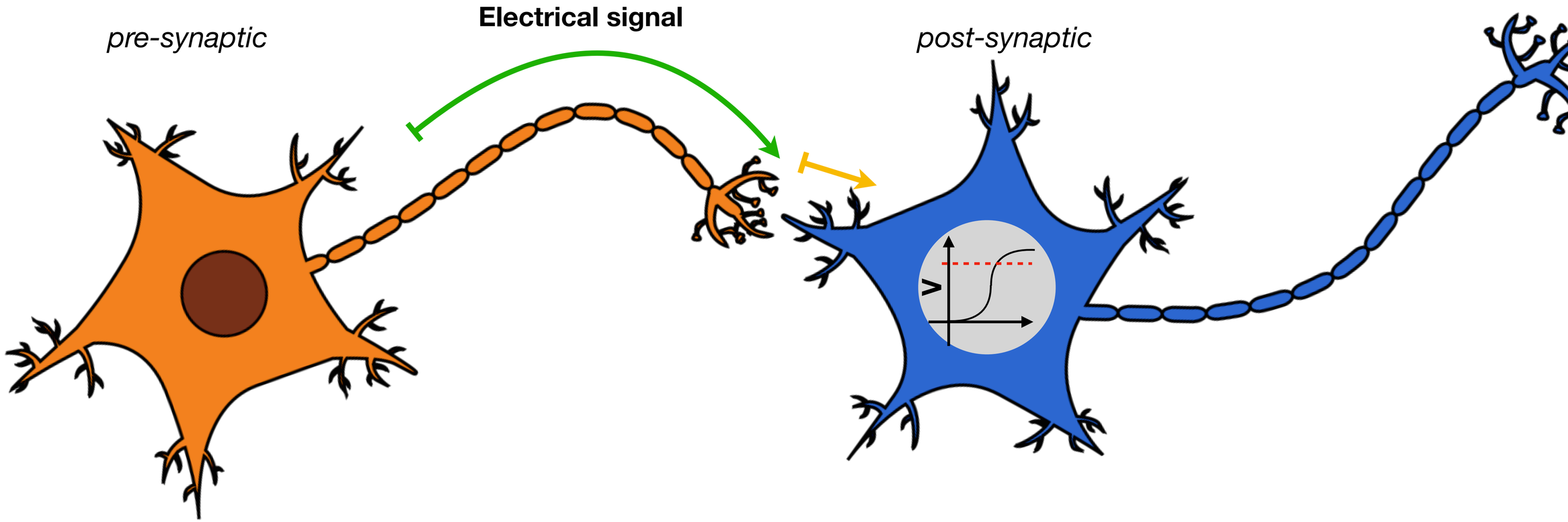
A neuron is a cell that collects and transmits electrical activity. Neurons exist in many different shapes and sizes.



An average (adult) neuron possesses between 1000 and 10,000 connections to other neurons

A simple model of neurons

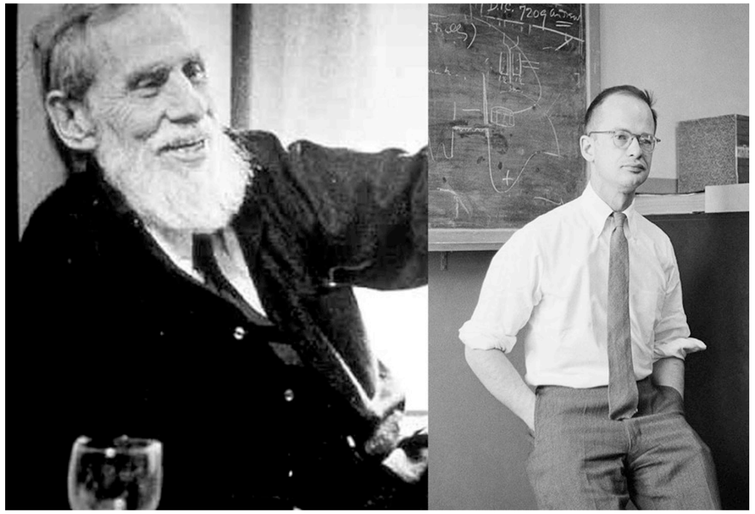
Neurons as logical units



Neurons communicate with each other by passing electrochemical signals from the axon terminals in the pre-synaptic neuron to the dendrites in the post-synaptic neuron. For a neuron to “fire”, a certain voltage threshold must be passed.

Neurons as logical units

McCulloch–Pitts neurons (1943)



if a neuron receives enough excitatory input that is not compensated by equally strong inhibitory input, it becomes active and sends a signal to other neurons

BULLETIN OF
MATHEMATICAL BIOPHYSICS
VOLUME 5, 1943

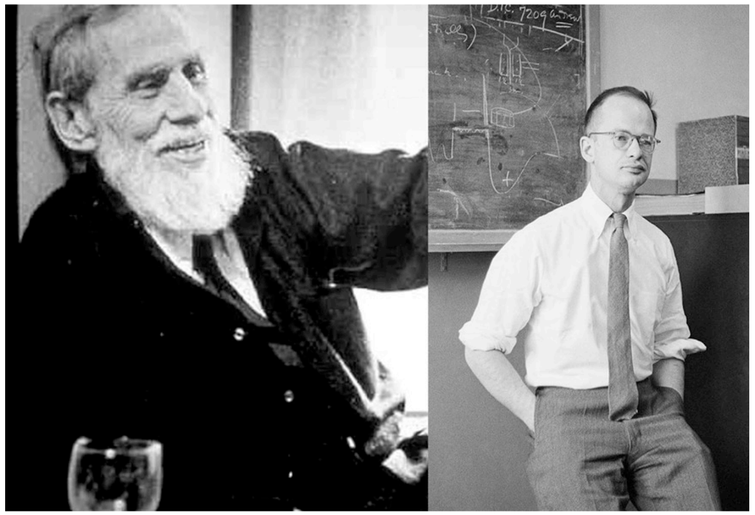
A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

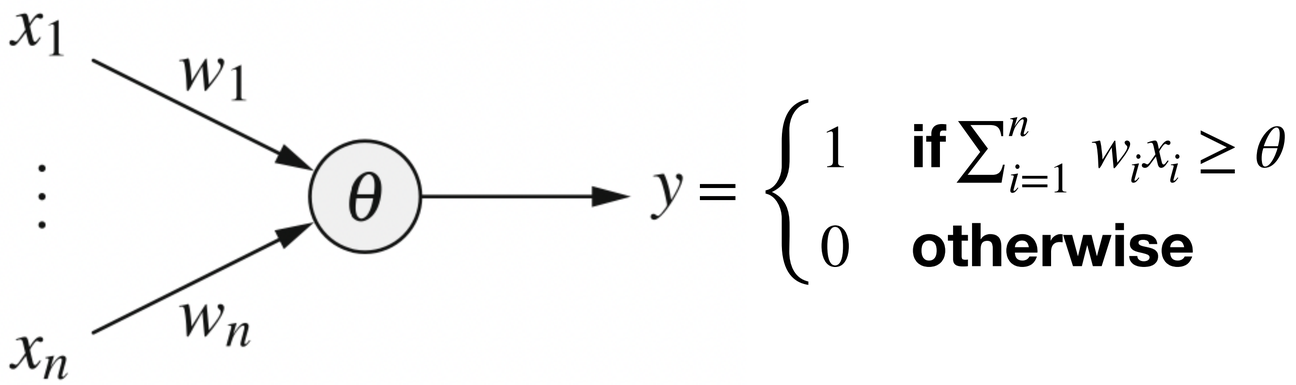
WARREN S. MCCULLOCH AND WALTER PITTS

FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,
AND THE UNIVERSITY OF CHICAGO

Neurons as logical units

McCulloch–Pitts neurons (1943)

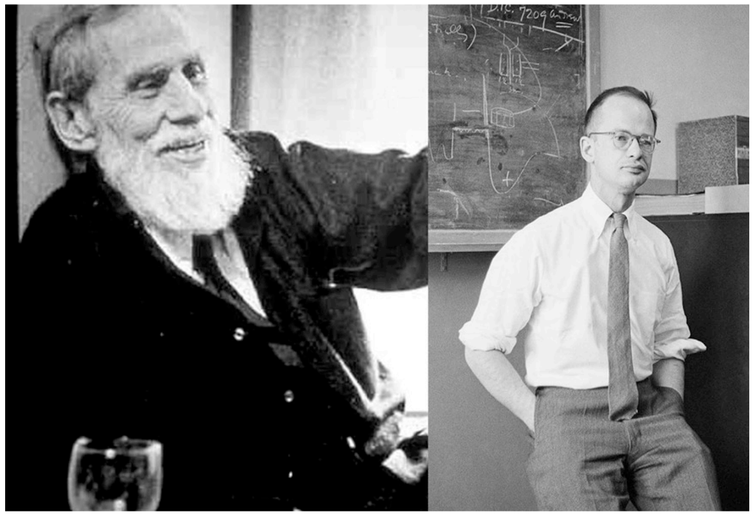


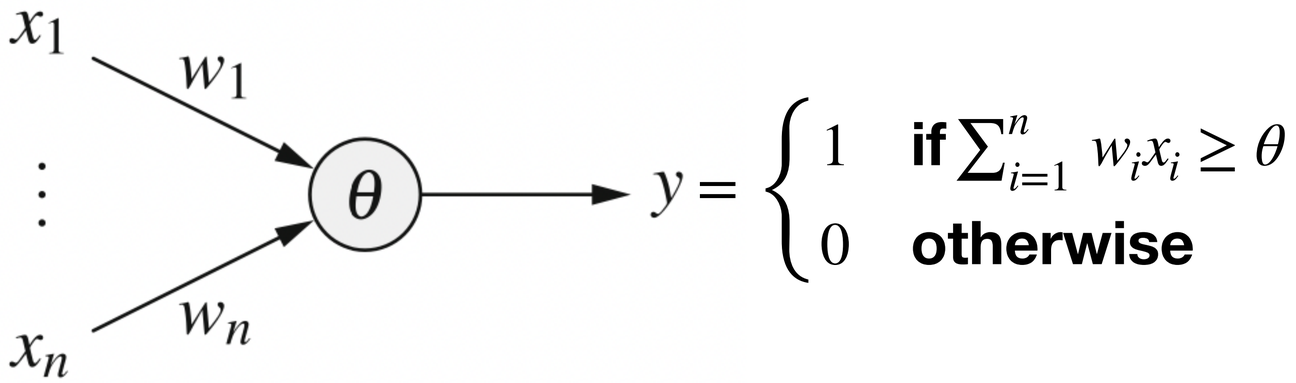


if a neuron receives enough excitatory input that is not compensated by equally strong inhibitory input, it becomes active and sends a signal to other neurons

Neurons as logical units

McCulloch–Pitts neurons (1943)



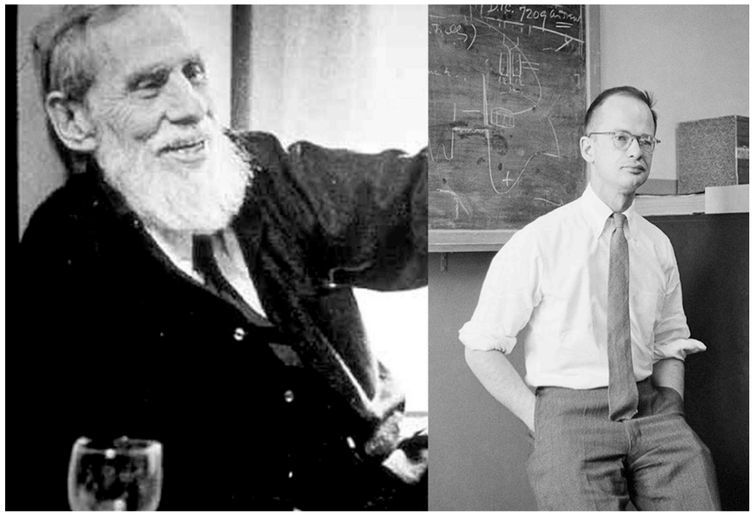


if a neuron receives enough excitatory input that is not compensated by equally strong inhibitory input, it becomes active and sends a signal to other neurons

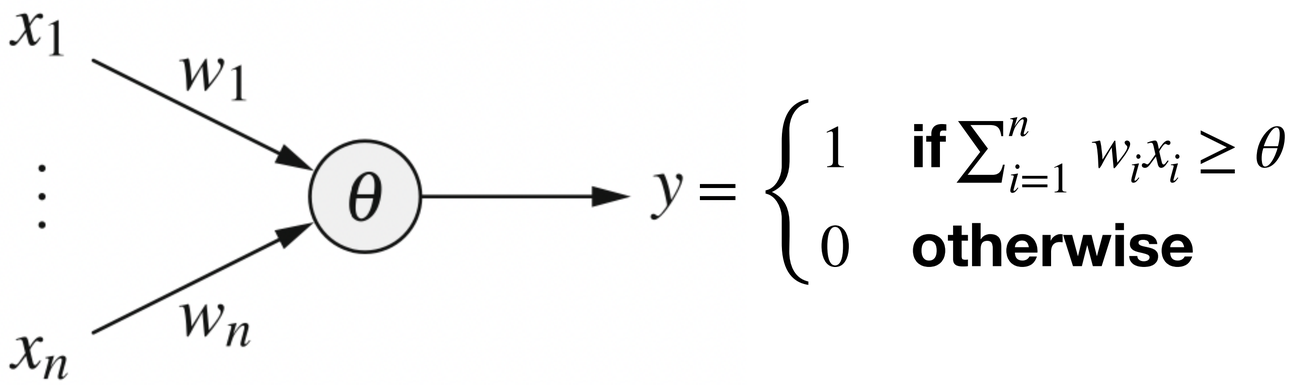
1. Neuron activation is binary. A neuron either *active* or *inactive*
2. For a neuron to fire, the weighted sum of inputs has to be equal to or larger than threshold
3. If one or more inputs are inhibitory the neuron will not fire
4. It takes a single “time step” for the signal to pass-through
5. The structure and the weights do not change over time

Neurons as logical units

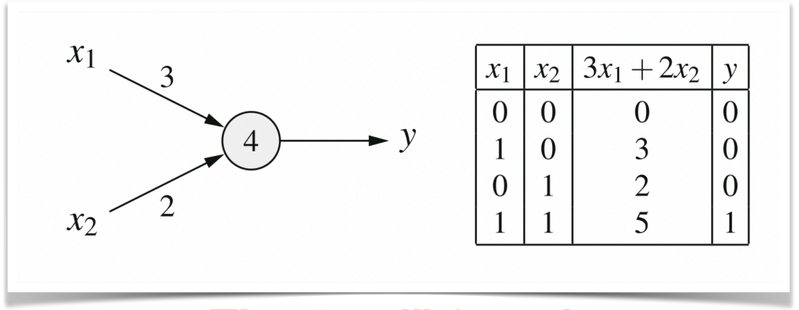
McCulloch–Pitts neurons (1943)



if a neuron receives enough excitatory input that is not compensated by equally strong inhibitory input, it becomes active and sends a signal to other neurons



McCulloc-Pitts computations



x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

The “and” function

The Perceptron

Frank Rosenblatt (1958)

Rosenblatt, with the image sensor of the Mark I Perceptron

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

McCulloc-Pitts computations only allow to perform a wide variety of boolean functions but higher-order organisms can perform more complex cognitive skills such as:

1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

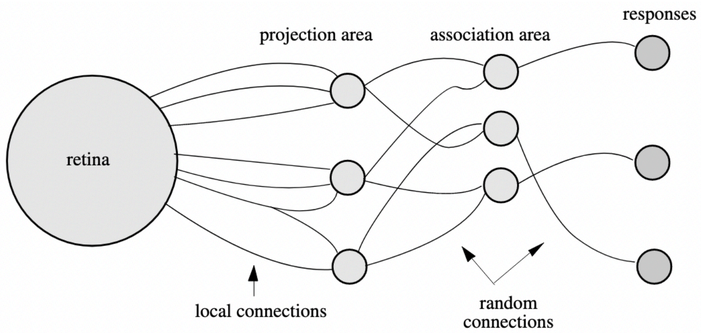
The Perceptron

Frank Rosenblatt (1958)

Rosenblatt, with the image sensor of the Mark I Perceptron

McCulloc-Pitts computations only allow to perform a wide variety of boolean functions but higher-order organisms can perform more complex cognitive skills such as:

1. detection
2. **storage**
3. **the use of the stored information in further behavioral or perceptual processes**

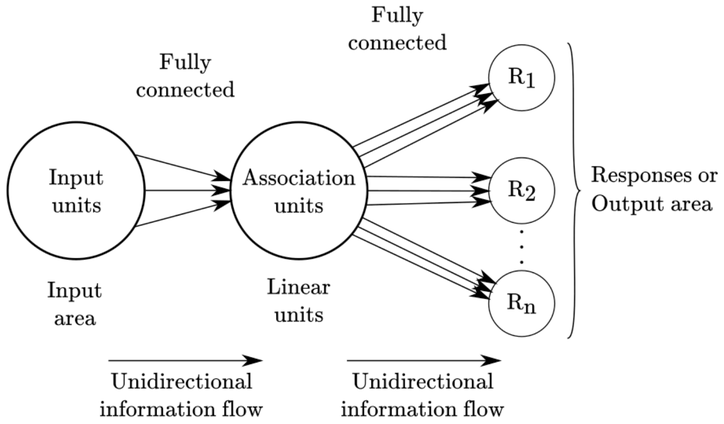


Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.

The Modern Perceptron

Frank Rosenblatt (1958)

Rosenblatt, with the image sensor of the Mark I Perceptron



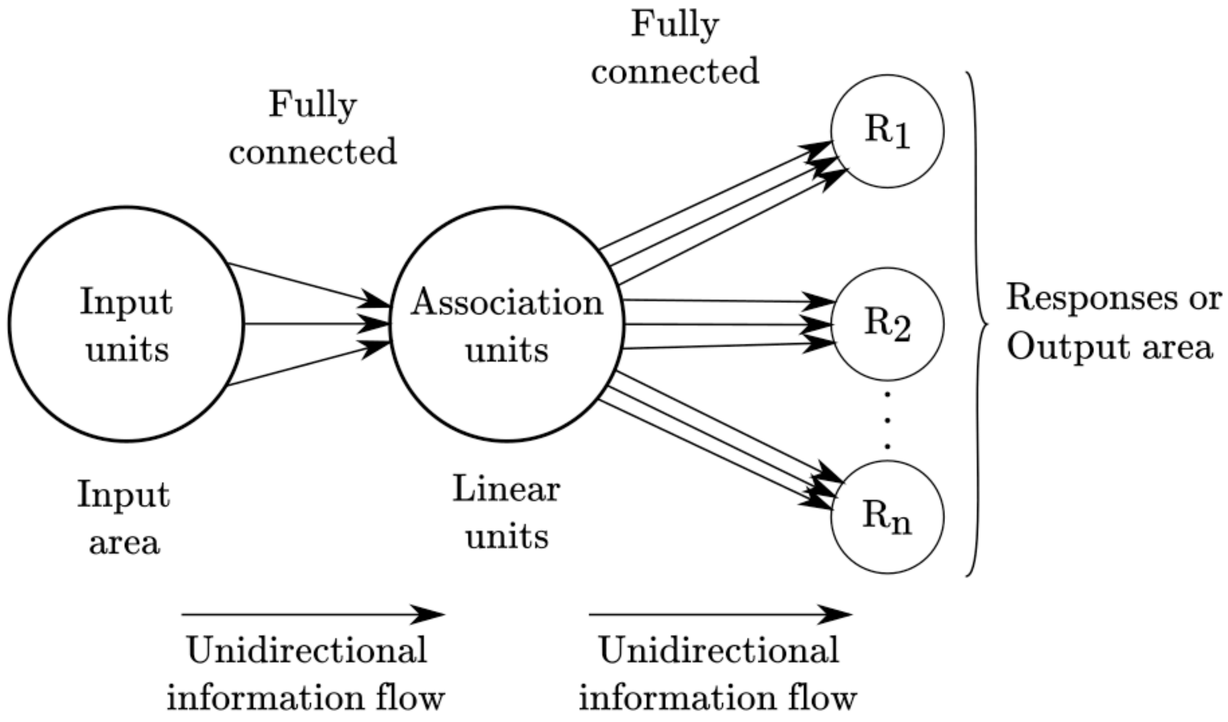
Mathematically, the perceptron can be described by:

- a *linear function* that aggregates the input signals
- a *threshold function* that determines if the response neuron activate or not
- a *learning procedure* to adjust connection weights

The Modern Perceptron

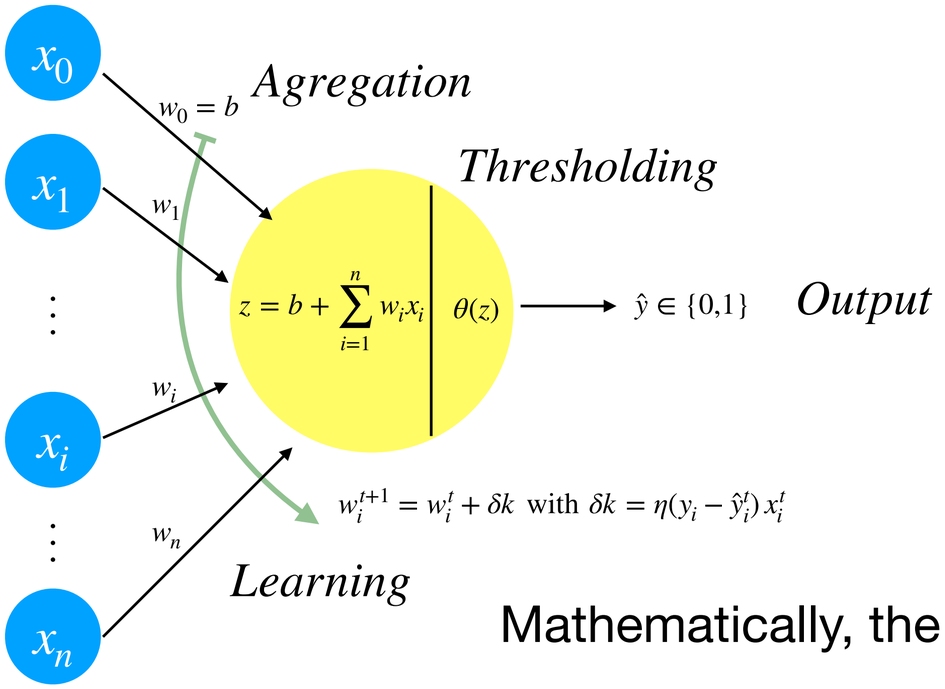
Frank Rosenblatt (1958)

Rosenblatt, with the image sensor of the Mark I Perceptron



1. the layers are fully connected, instead of partially connected at random.
2. the inputs and outputs can be real-valued numbers, instead of only binary values.
3. linear functions are used for the units in the intermediate layers (if any) rather than threshold functions.
4. it implements an error-based learning algorithm instead of the pattern association mechanism of the photo-perceptron.

The Modern Perceptron



Mathematically, the perceptron can be described by:

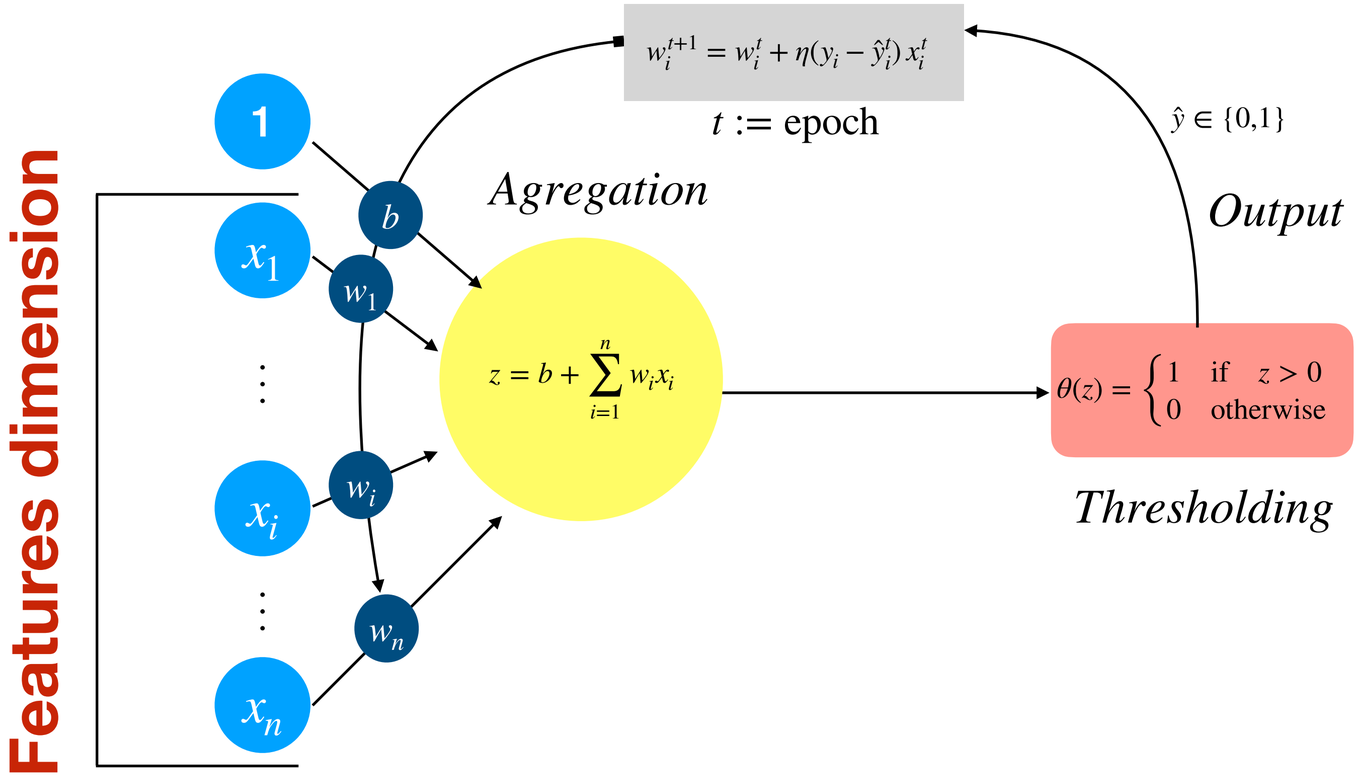
$$z = x_0\, b + \sum_{i=1}^{n} w_i x_i \qquad x_0 := \textit{bias}$$

$$\hat{y} := \theta(z) = \begin{cases} 1 & \text{if} \quad z > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_i^{t+1} = w_i^t + \eta(y_i - \hat{y}_i^t)\, x_i^t$$

The Modern Perceptron

Algorithm

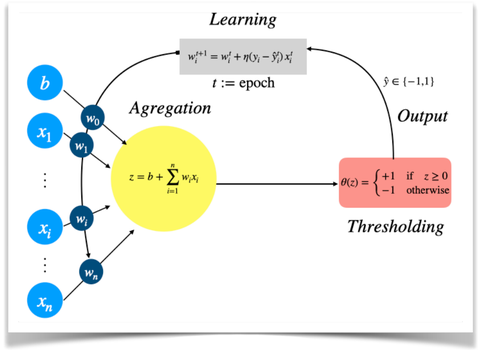


Important:

We repeat the algorithm until the error $\Delta error = error_{t+1} - error_t \sim 0$ *(or some small value). Where* $error$ *is equal to:*

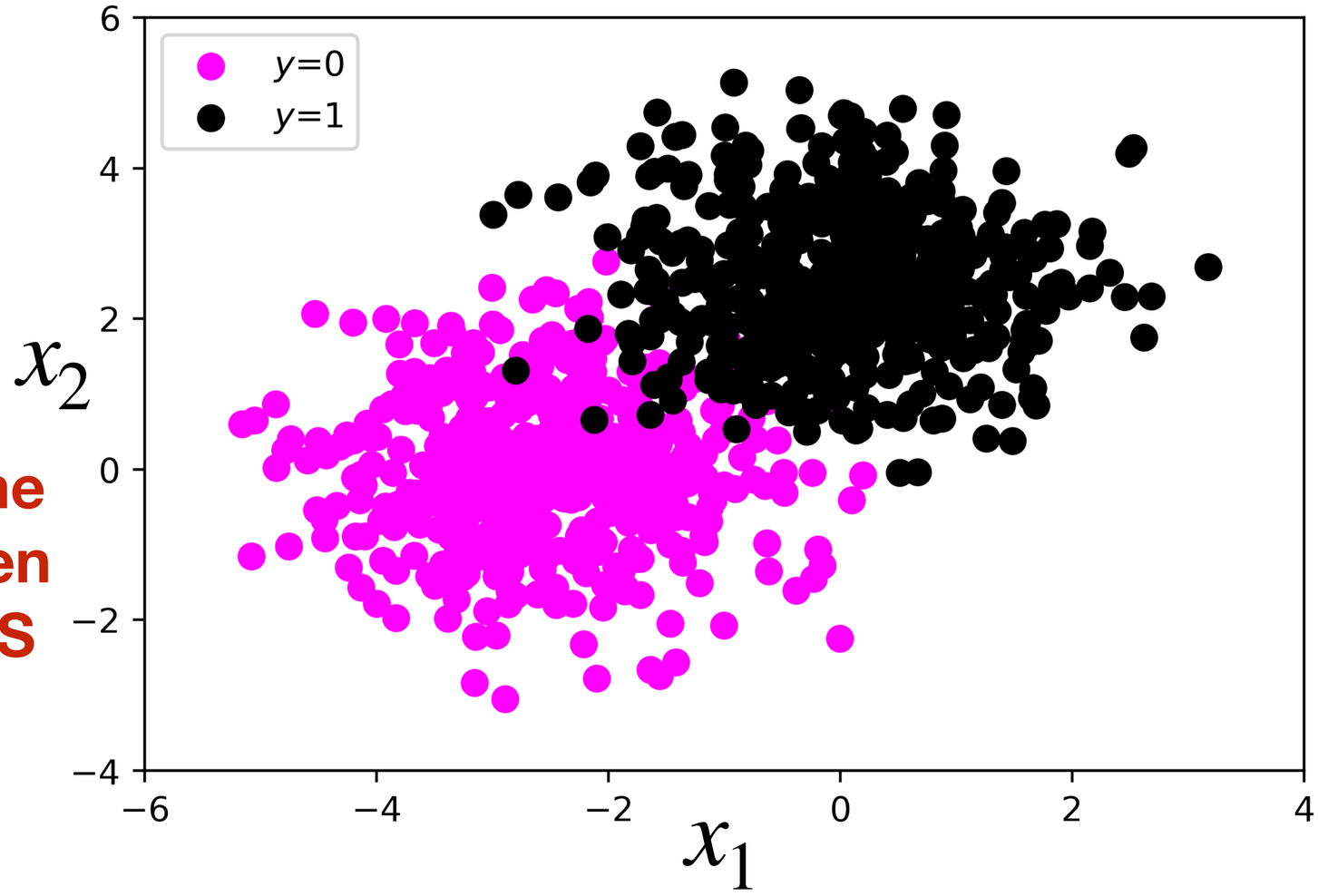
$$error = \sum_{i}^{n} (y_i - \hat{y}_i)^2$$

The Modern Perceptron



Example

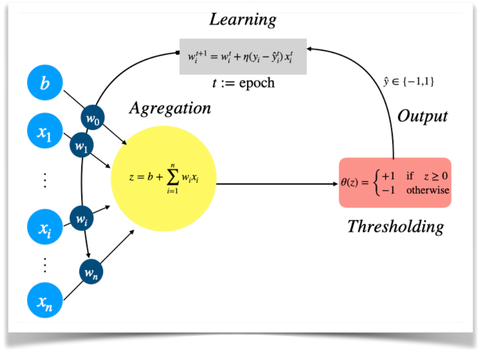
In this case there are 2 features



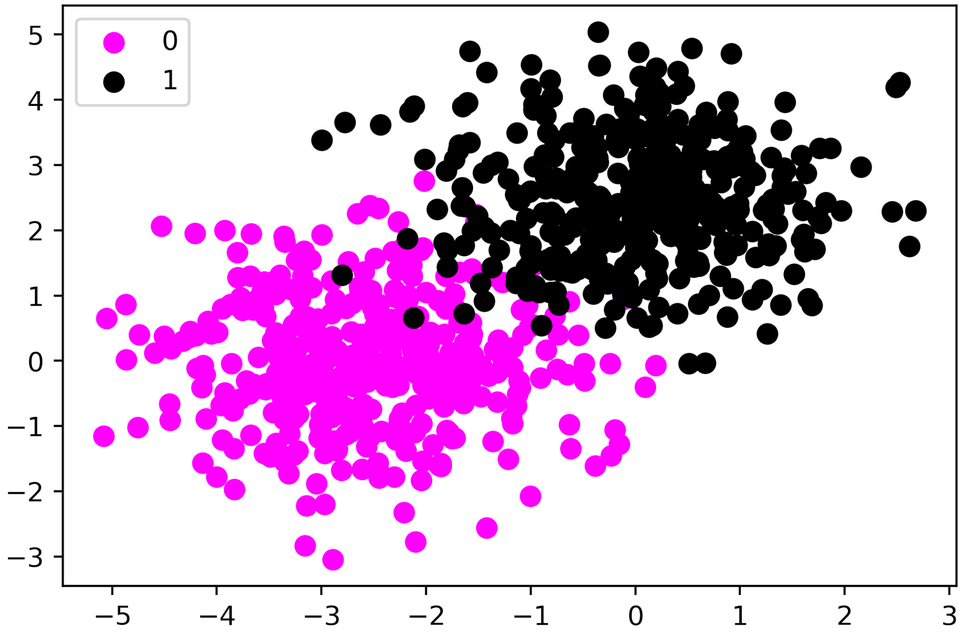
Clearly there is some pattern that has been provided as LABELS (i.e supervised learning)

The Modern Perceptron

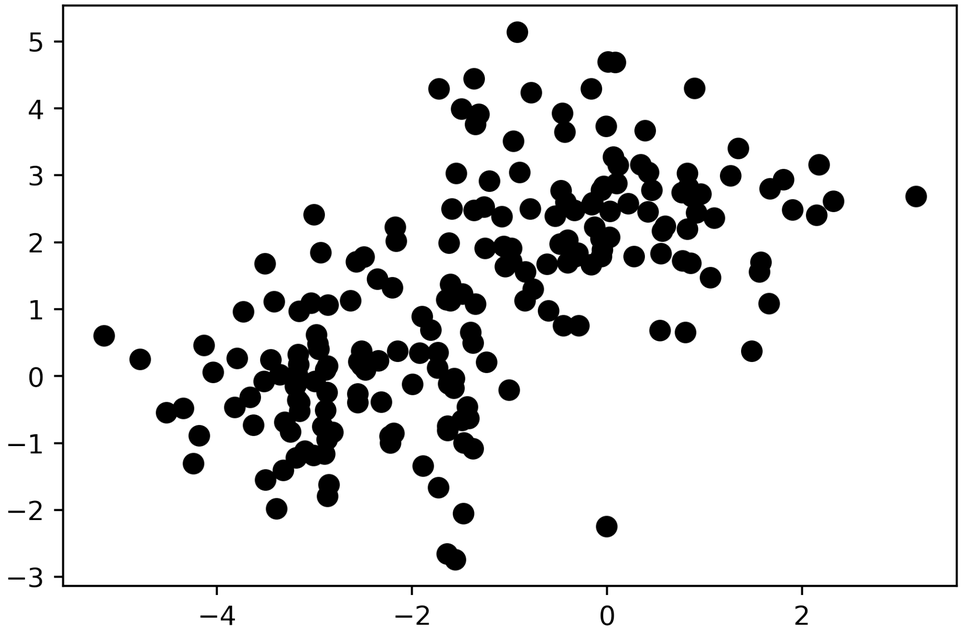
Example



(1) We can randomly pick up some LABEL data to train our neural network

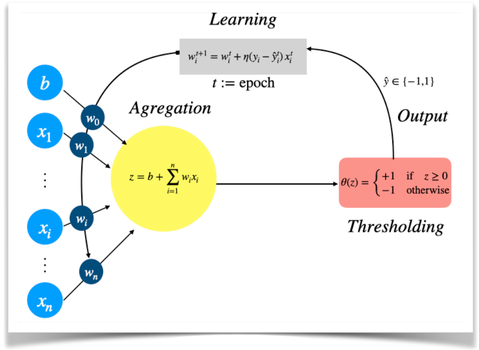


(2) We can randomly pick up some data that the NN never seen before and check if the label are recognized appropriately



The Modern Perceptron

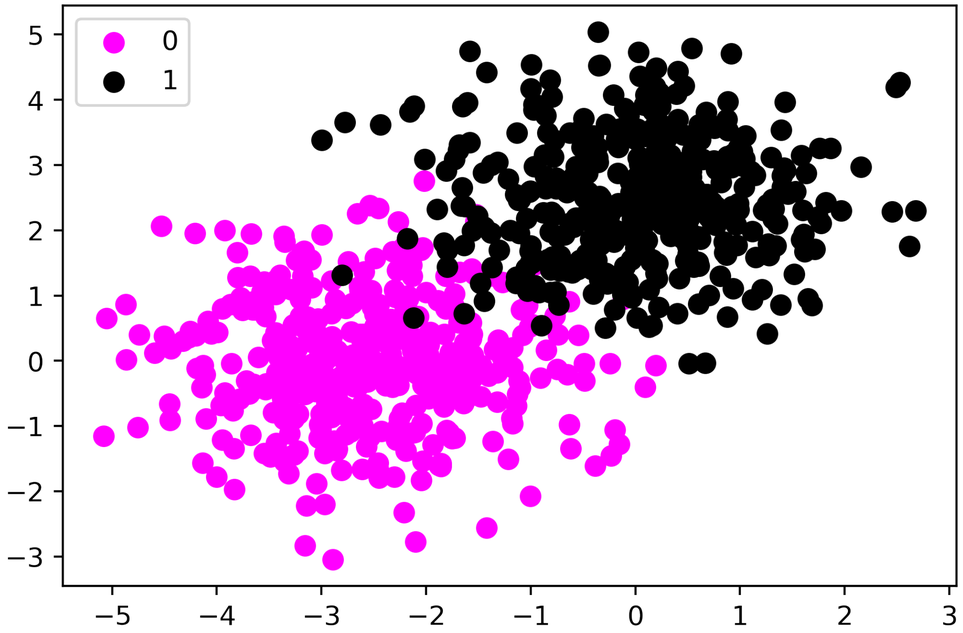
Example

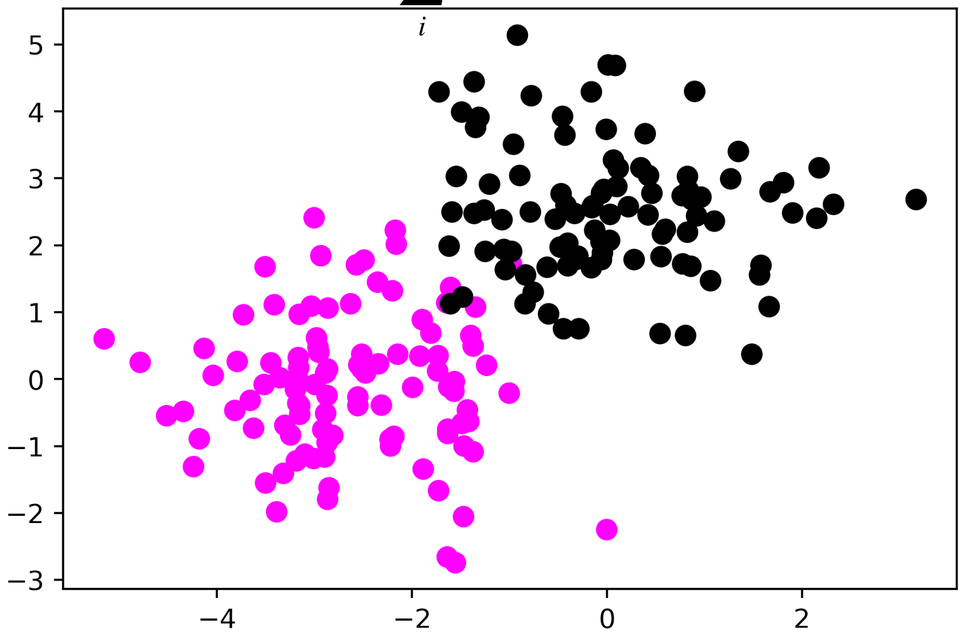


(1) We can randomly pick up some LABEL data to train our neural network

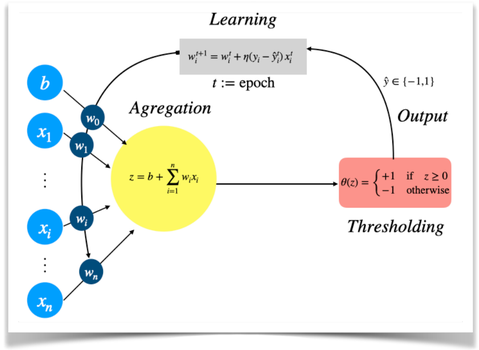
(2) We can randomly pick up some data that the NN never seen before and check if the label are recognized appropriately

$$error = \sum_{i}^{n} (y_i - \hat{y}_i)^2 = 0.105$$

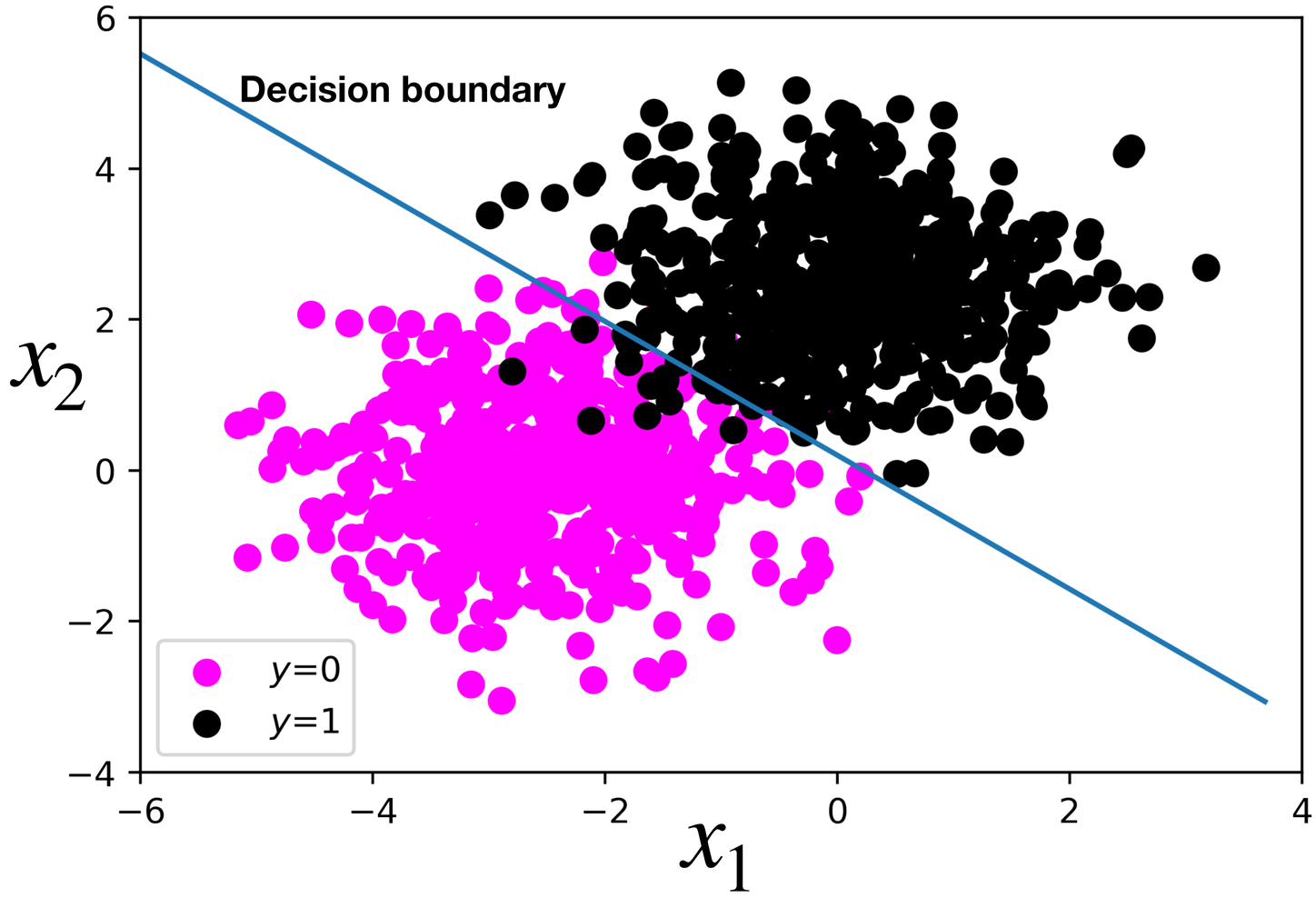




The Modern Perceptron

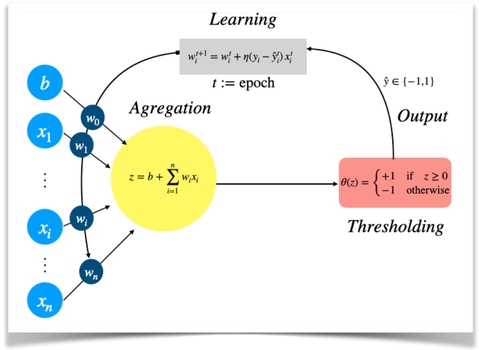


Example

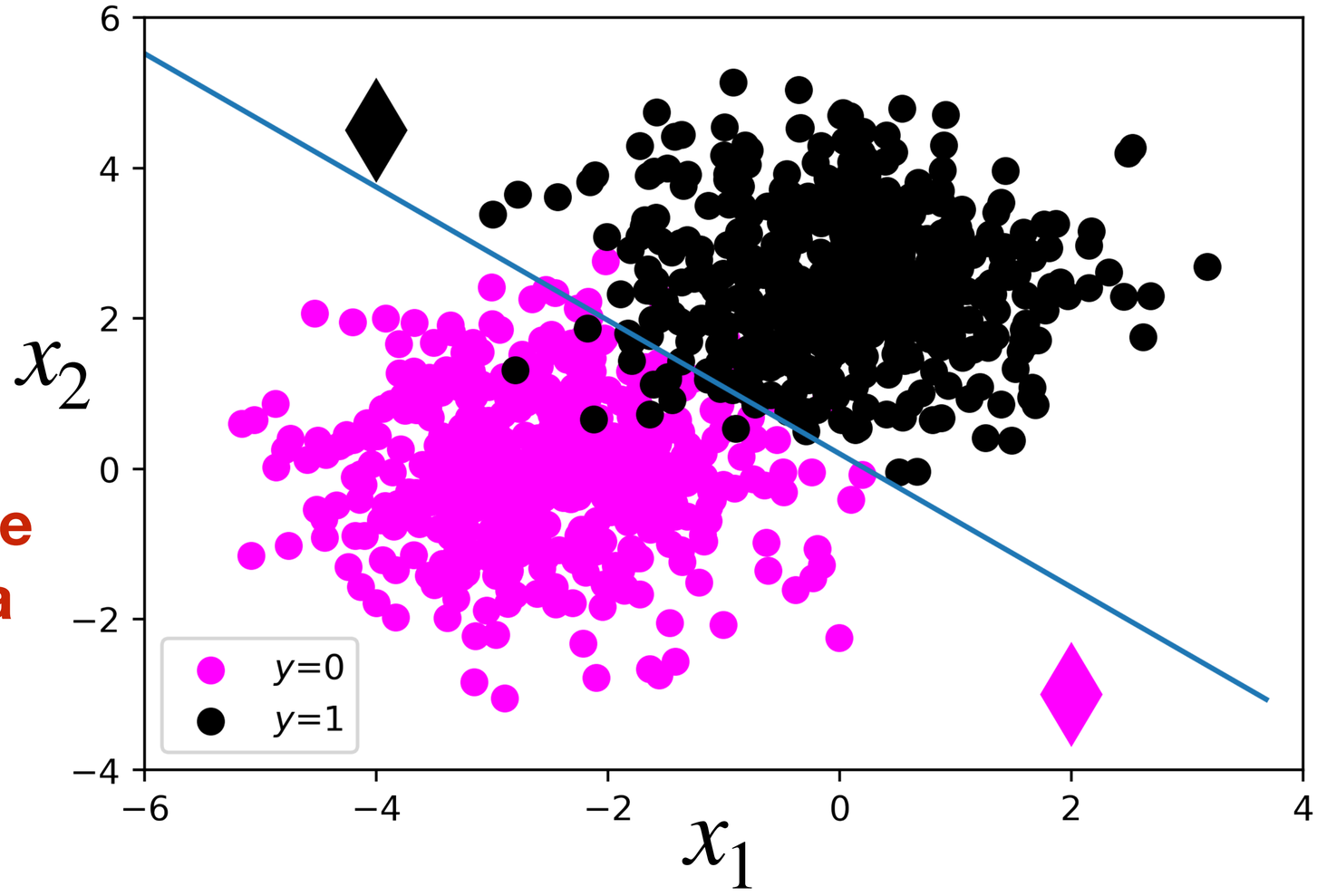


The Modern Perceptron

Example



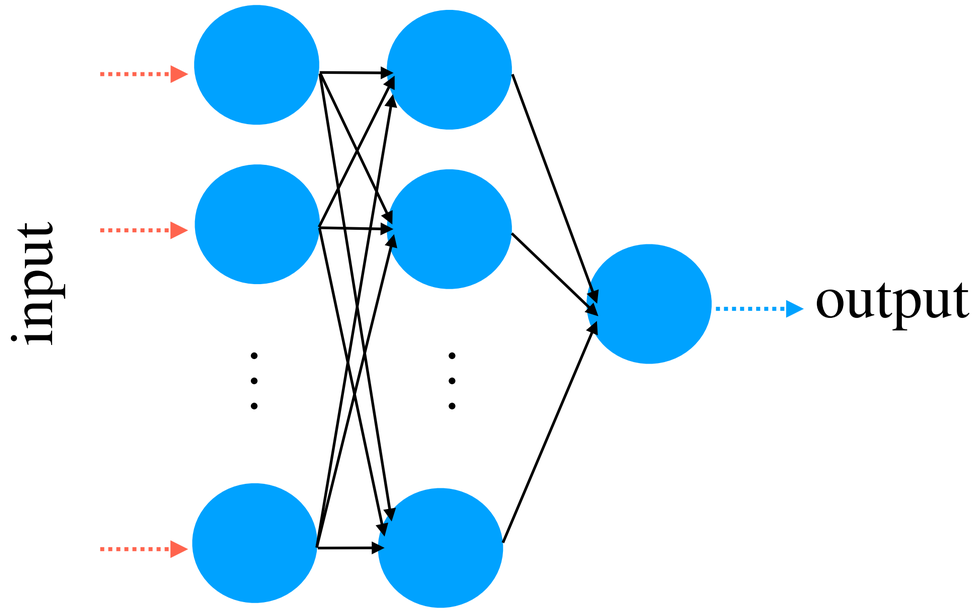
Now we can use the NN to classify data that never have seen!!



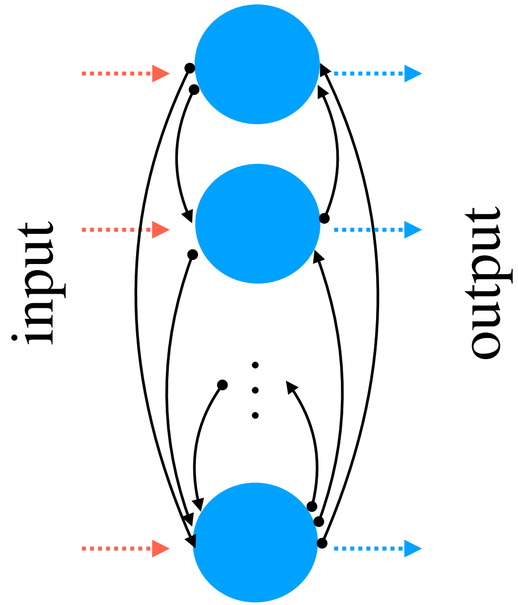
Memory and Neural Networks

Recap

Feed Forward

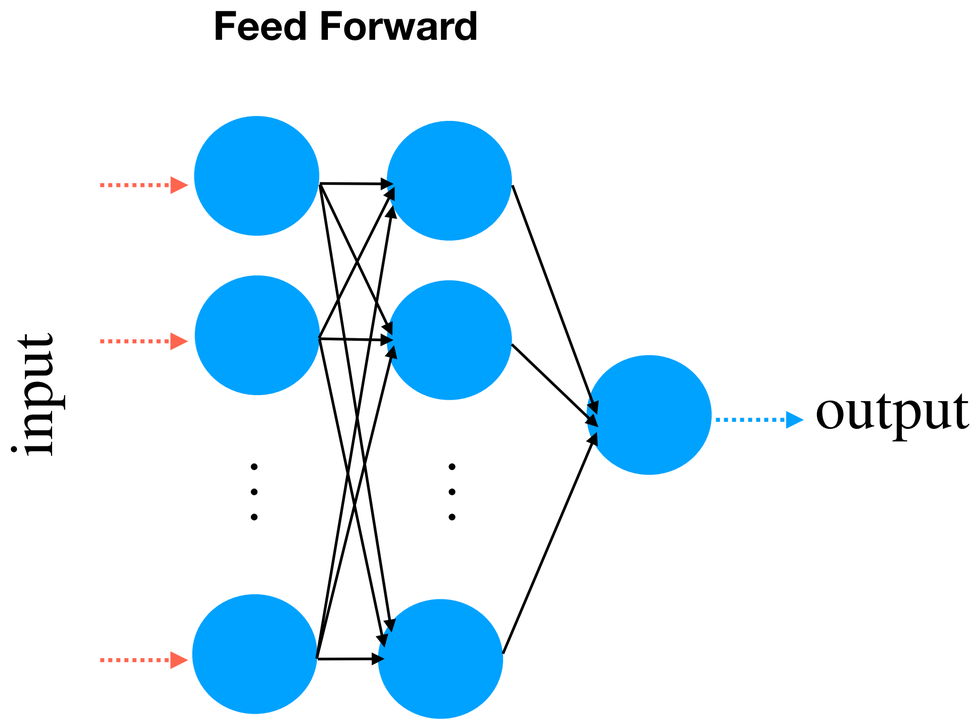


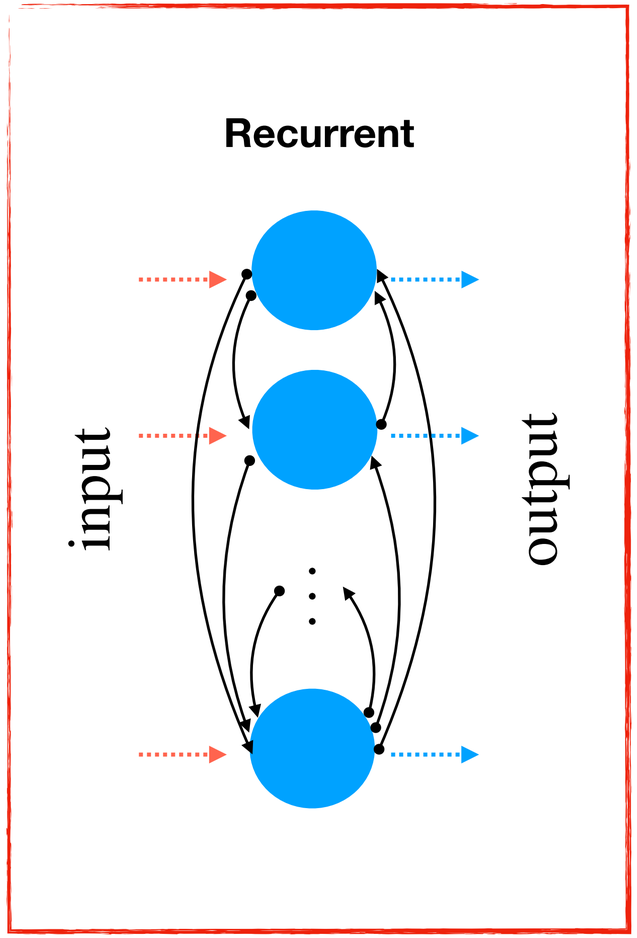
Recurrent



Memory and Neural Networks

Recap





Memory and Neural Networks

Hopefield Recurrent Neural Network (1982)

Proc. Natl. Acad. Sci. USA
Vol. 79, pp. 2554–2558, April 1982
Biophysics

Check for updates

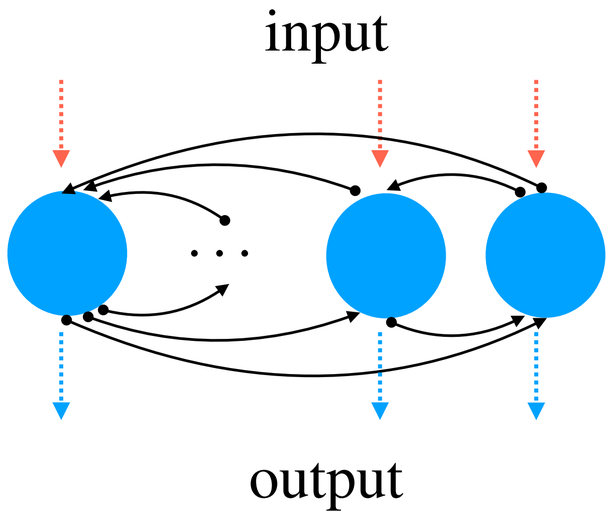
Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

J. J. HOPFIELD

Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125; and Bell Laboratories, Murray Hill, New Jersey 07974

Contributed by John J. Hopfield, January 15, 1982

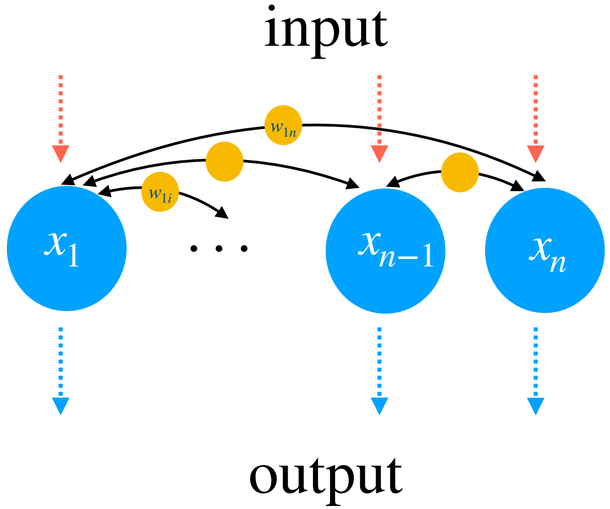


Neural networks modeled as a spin glass system

$$H = -\sum_{<ij>} J_{ij} s_i s_j - h \sum_i s_i$$

Memory and Neural Networks

Hopefield Recurrent Neural Network (1982)



Fully connected spin glass!

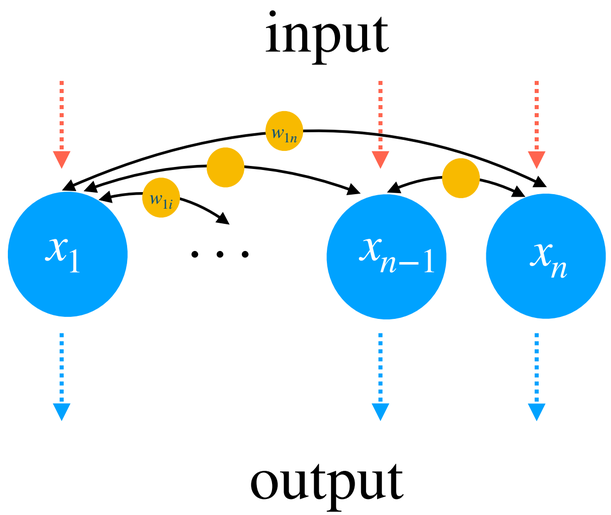
Neural networks modeled as a spin glass system

$$H = -\sum_{<ij>} J_{ij} s_i s_j - h \sum_i s_i$$

$$E_i = -\sum_{ij} w_{ij} x_i x_j \qquad x_i \in \{-1,1\} \text{ and } w_{ij} \in \mathbb{R}$$

Memory and Neural Networks

Hopefield Recurrent Neural Network (1982)



Fully connected spin glass!

$$E = -\sum_{ij} w_{ij} x_i x_j \qquad x_i \in \{-1,1\} \text{ and } w_{ij} \in \mathbb{R}$$

For training procedure, it doesn't require iterations. It includes just an outer product between the input vector and transposed input vector to fill the weighted matrix w_{ij} (or synaptic weights) and in the case of many patterns it is as follow:

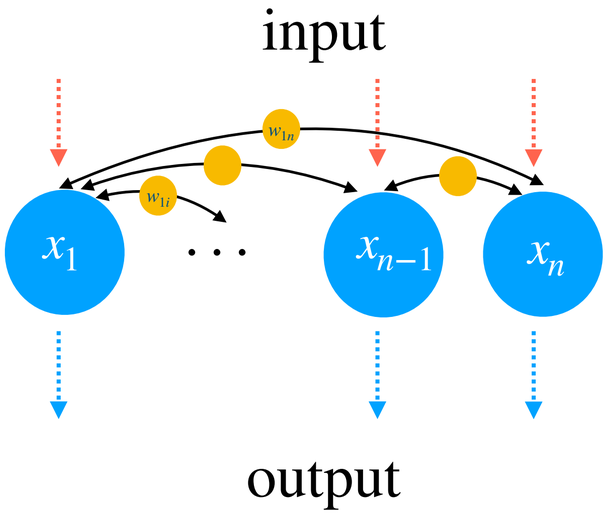
$$w_{ij} = \frac{1}{N} \sum_{\alpha=1}^{p} \epsilon_i^\alpha \epsilon_j^\alpha$$

where ϵ^α is a pattern and p is the total number of patterns.

The wights can be chosen in a way that we can store information or patterns if the x_i represent white (black) pixels in an image.

Memory and Neural Networks

Hopefield Recurrent Neural Network (1982)



Fully connected spin glass!

$$E = -\sum_{ij} w_{ij} x_i x_j \qquad x_i \in \{-1,1\} \text{ and } w_{ij} \in \mathbb{R}$$

For training procedure, it doesn't require iterations. It includes just an outer product between the input vector and transposed input vector to fill the weighted matrix w_{ij} (or synaptic weights) and in the case of many patterns it is as follow:

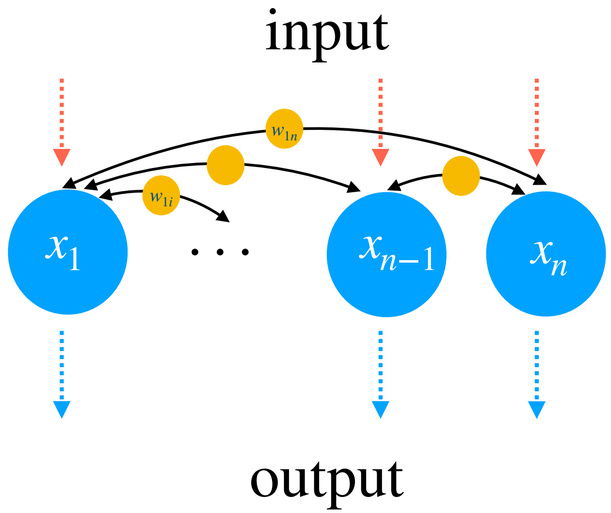
$$w_{ij} = \frac{1}{N} \sum_{\alpha=1}^{p} \epsilon_i^\alpha \epsilon_j^\alpha$$

where ϵ^α is a pattern and p is the total number of patterns.

If the bits correspond 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 i and j will tend to become equal. The opposite happens if the bits correspond to neurons i and j are different.

Hopefield Recurrent Neural Networks

Algorithm



> 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 ~ 0.138 pattern per node

The energy of ith spin can be written as:

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

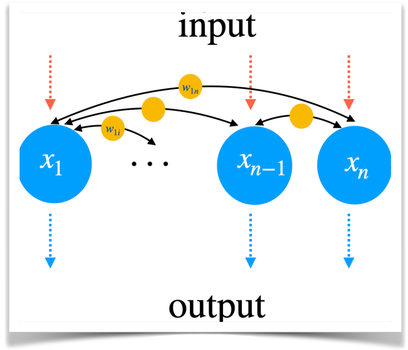
So then if the product then the field would try to flip the spin:

$$x_i^{t+1} = \operatorname{sgn}\left(w_{ij} x_i^t \right)$$

Which will guarantee to minimize the energy. Remember, here t referes to the epoch.

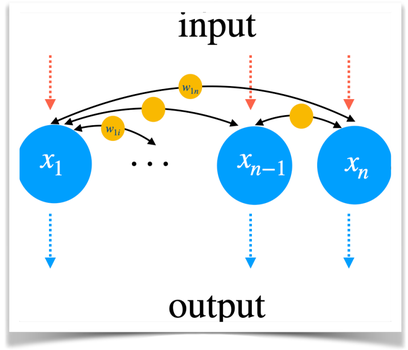
Hopefield Recurrent Neural Networks

Example

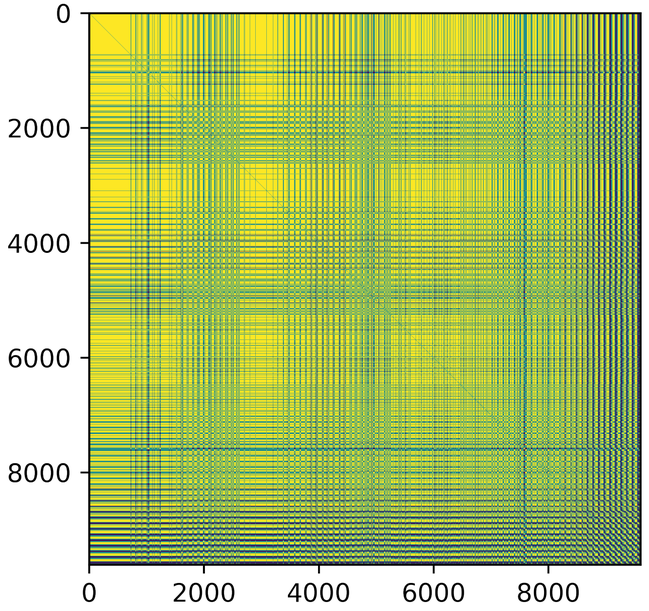


Hopefield Recurrent Neural Networks

Example Groundskeeper Willie and Bart



RGB->Binary $[-1,1]$

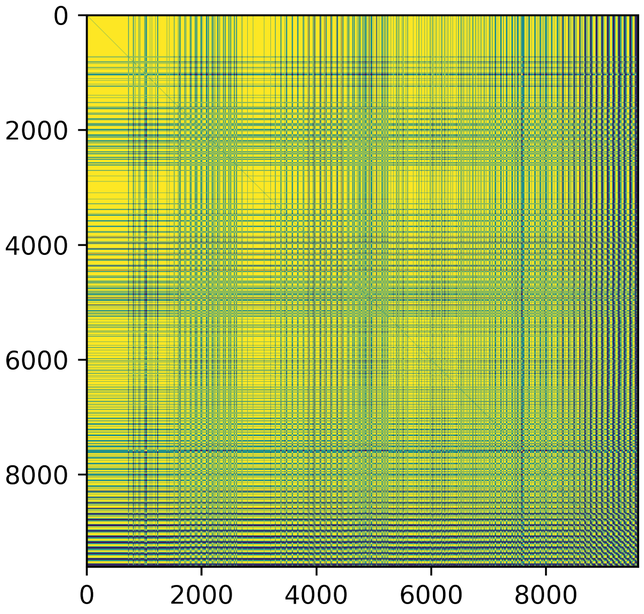


Train

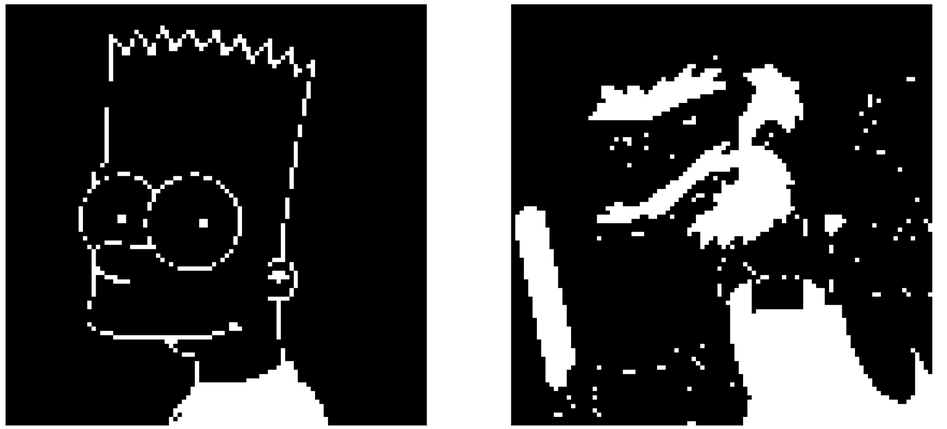
Hopefield Recurrent Neural Networks

Example Groundskeeper Willie and Bart

Memory



Reconstructing and remembering



Neural Networks and Memory LAB

LAB

- Code a simple linear classifier using the perceptron algorithm. To test your code use the following data: **train_data**, **validation_data.** To make the things simple use the bias $input_b = 1$

- If you need to generate more data use the following script: **here**.

Extra

- Use the Hopefield Recurrent Neural Network to “REMEMBER” and “RETRIEVE” the Simpson characters. You can download the images **here,** the folder contains numpy binary files that can be read by using **numpy.load**. Each file contains a binary matrix representation of each image.

- Remember that binary images have zeros and ones but the spins -1 and 1.