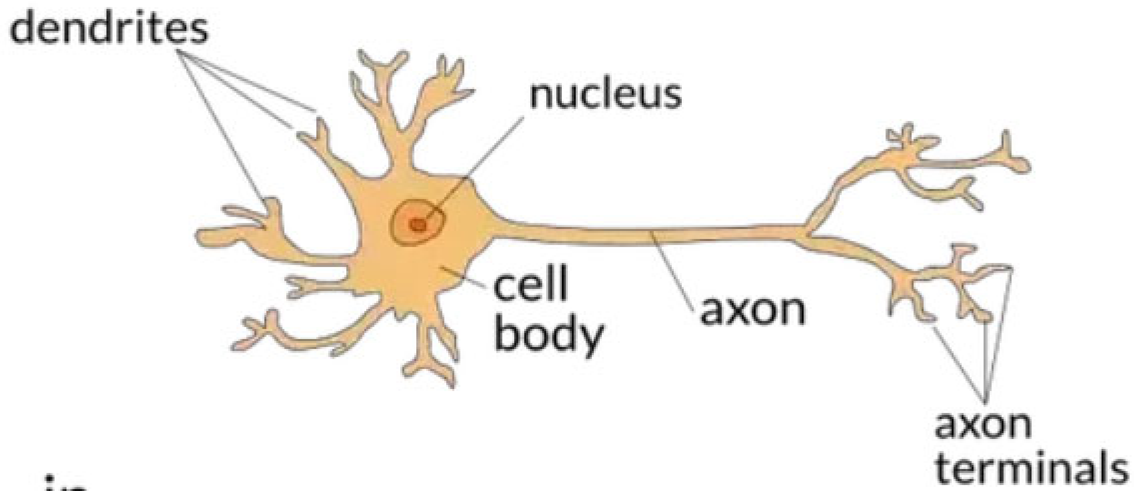
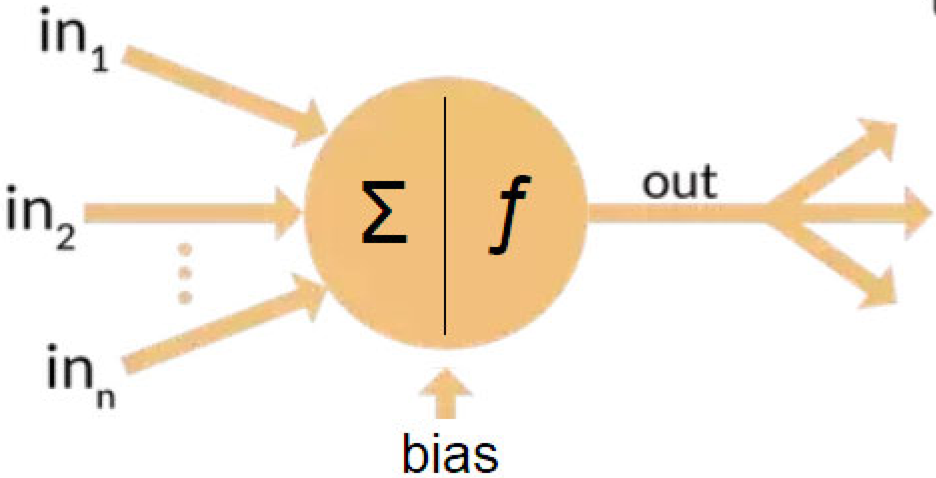
Machine Learning
Introduction to Neural Networks

Jian Liu



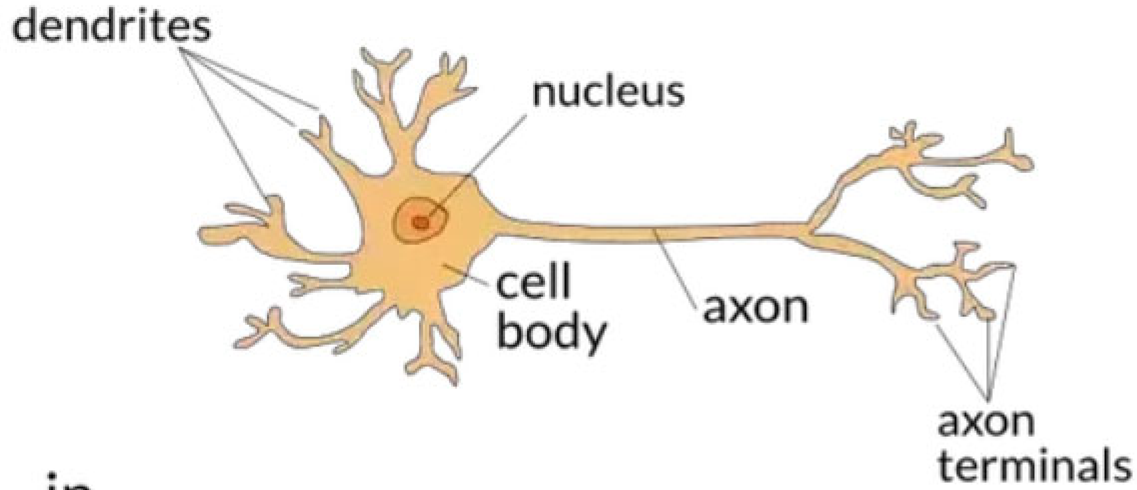


from towardsdatascience

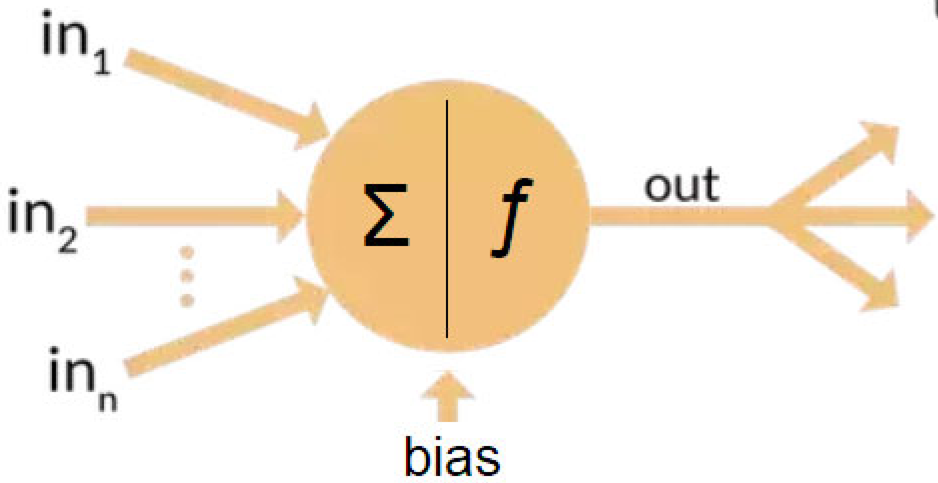
Machine Learning
Introduction to Neural Networks

Jian Liu

Part 1:
Biological neurons



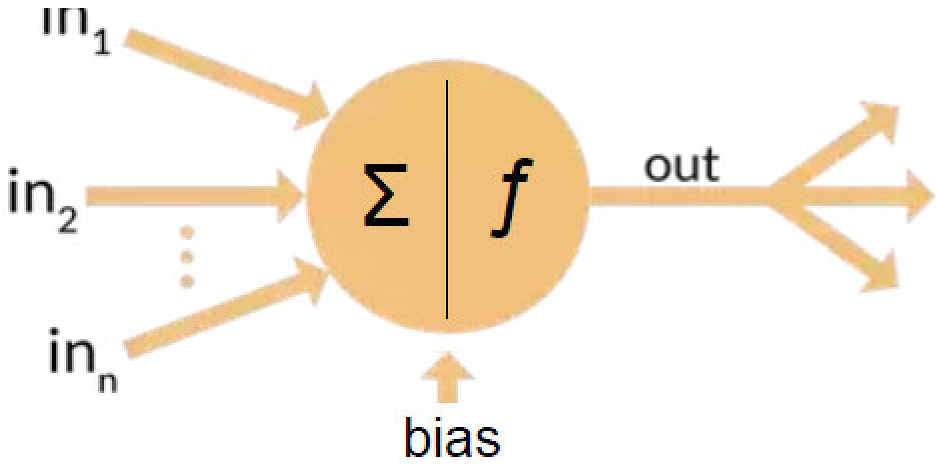
Part 2:
Artificial neurons



Machine Learning
Introduction to Neural Networks

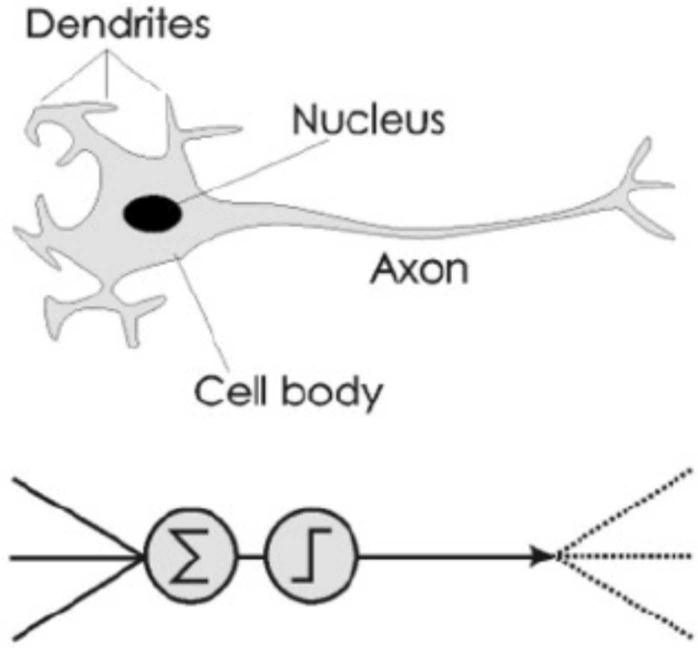
Jian Liu

Part 2:
Artificial neurons



McCulloch-Pitts (idea)

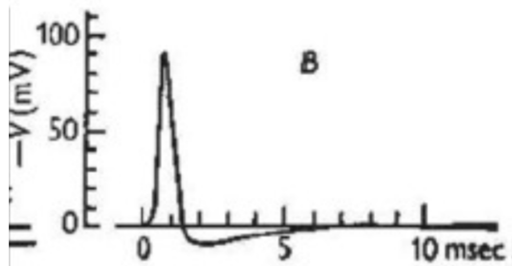
McCulloch and Pitts (1943): pioneers to formally define neurons as computational elements



The idea: explore simplified neural models to get the essence of neural processing by ignoring irrelevant detail and focusing in what is needed to do a computational task

McCulloch-Pitts (idea)

McCulloch and Pitts knew that spikes (action potential) somehow carry information through the brain:

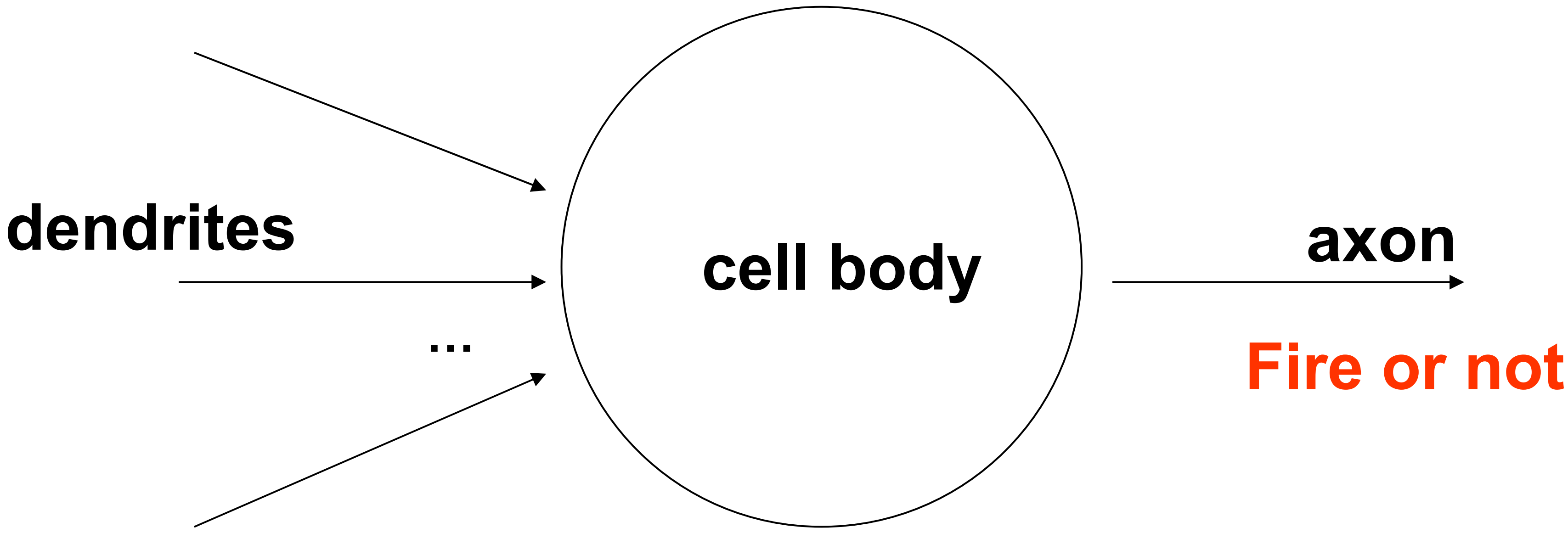


each spike would represent a binary 1

each lack of spike would represent a binary

They showed **how spikes could be combined to do logical and arithmetical operations**

Abstraction

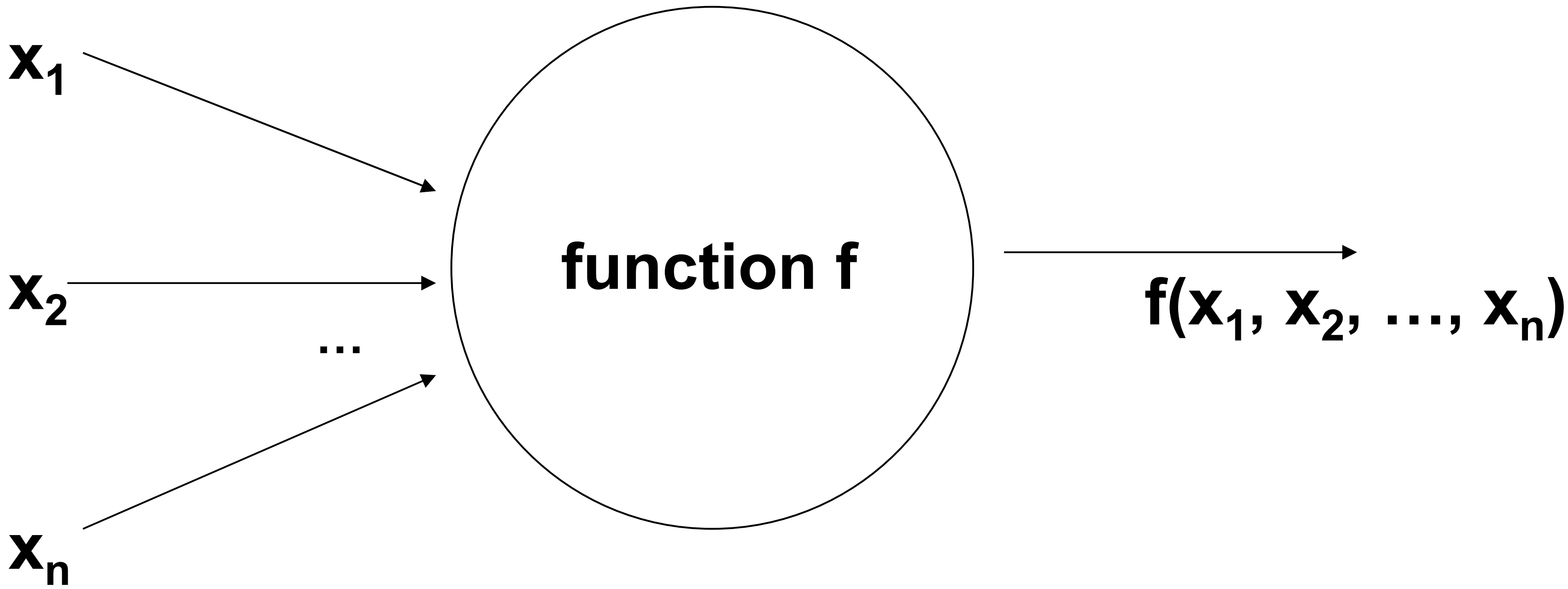


signal input

signal processing

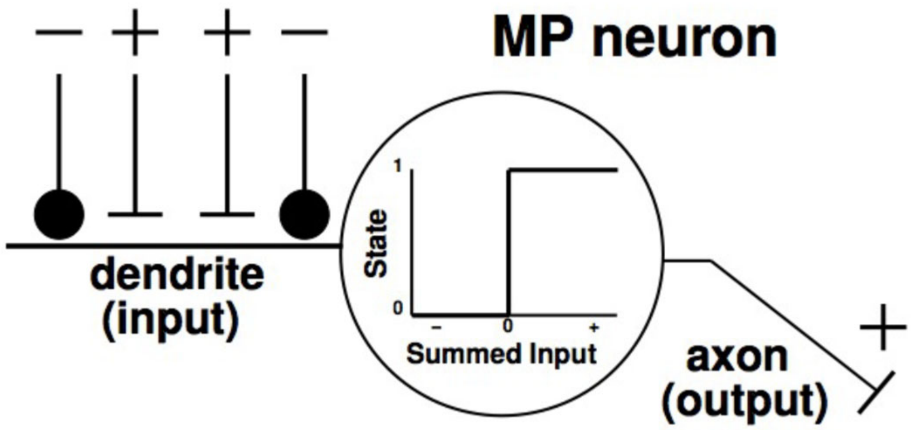
signal output

Model



McCulloch-Pitts (rule)

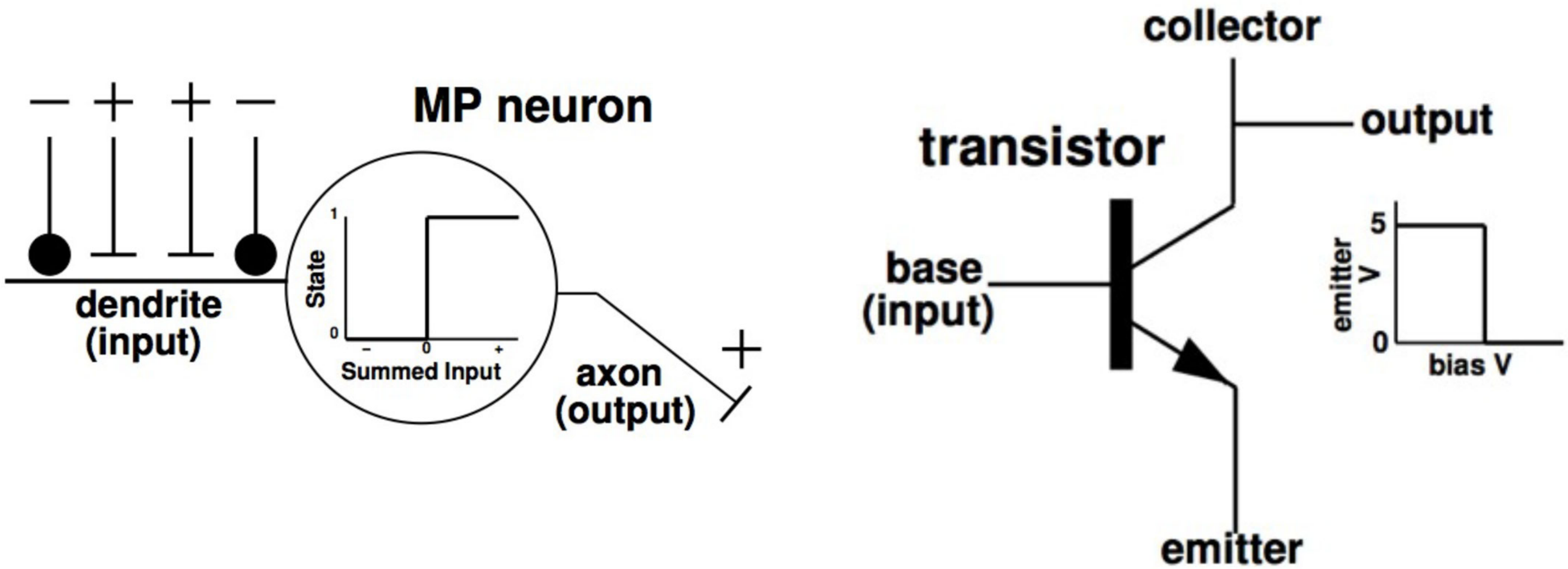
MP neurons are binary: **they take as input and produce as output only 0's or 1's**



Rule: activations from other neurons are summed at the neuron and outputs 1 if threshold is reached and 0 if not

McCulloch-Pitts (analogy)

MP neurons function much like a transistor:



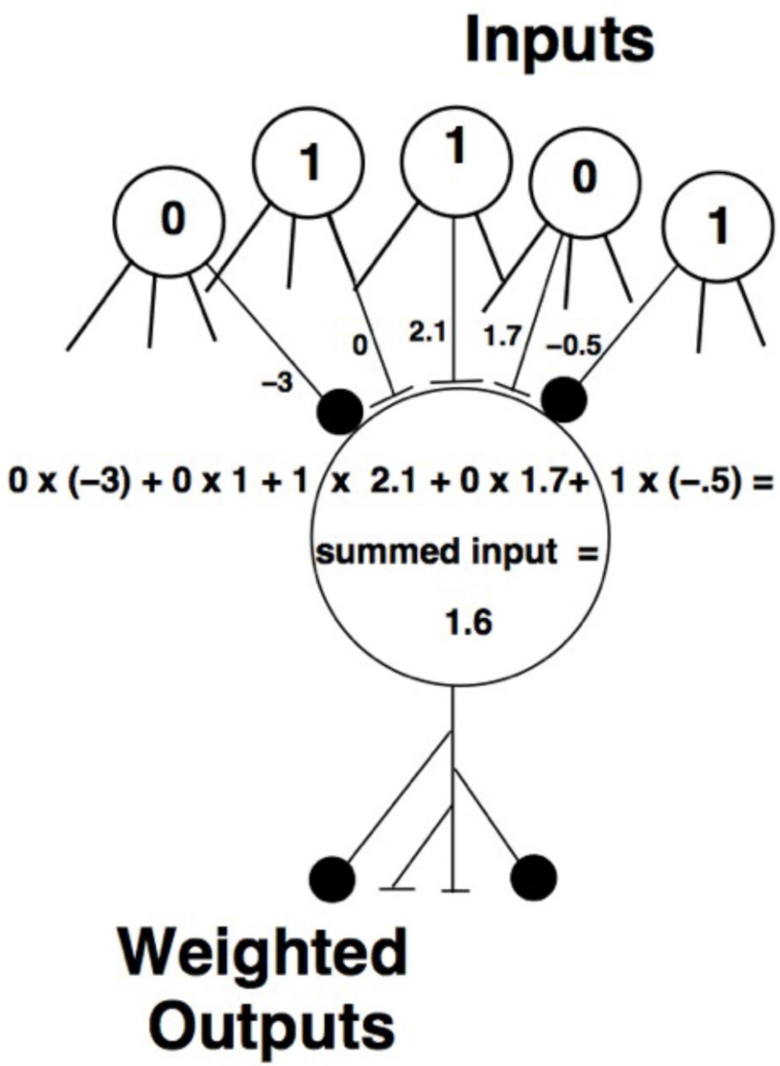
In artificial networks, inputs come from the outputs of other MP neurons (as transistors in a circuit)

McCulloch-Pitts (configuration)

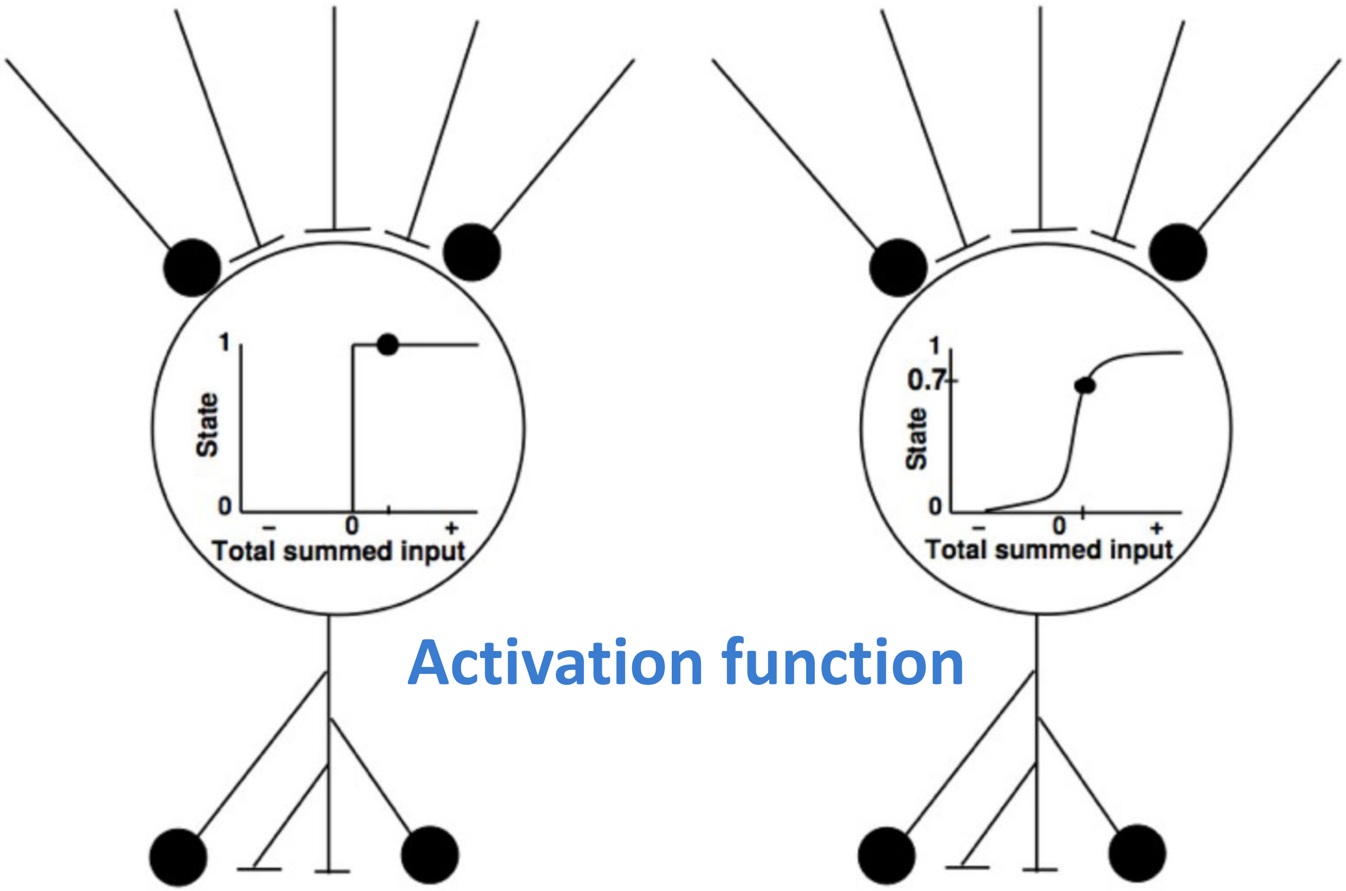
State: is the degree of activation of a single neuron

Weight: is the strength of the connection between two neurons

Update rule: determines how input to a neuron is translated into the state of that neuron



McCulloch-Pitts (states)

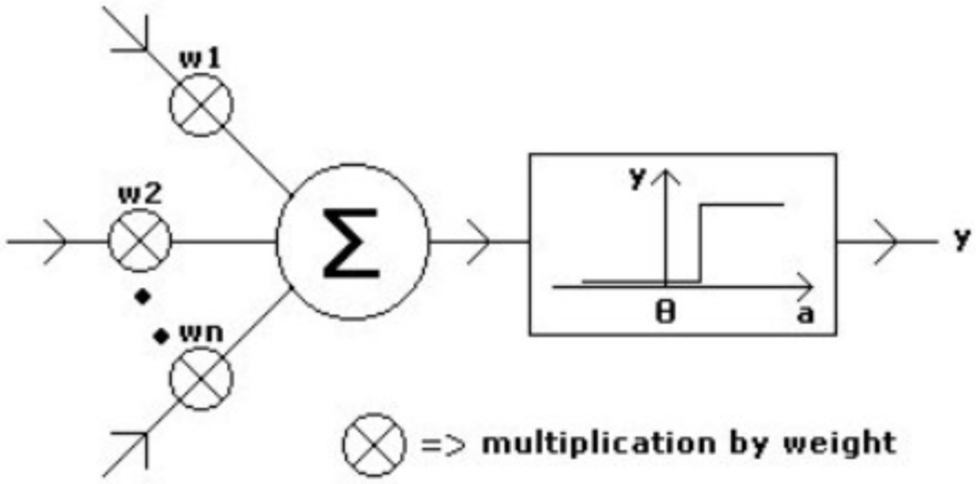


Sharp threshold (digital)

Sigmoid function (analog)

McCulloch-Pitts (states)

$$y = \phi\left(\sum_{i=1}^{n} \omega_i \cdot x_i\right)$$



Where *Φ* represents a threshold or a sigmoid function

```
total_input = sum(w.*x);
if total_input >= 0
    y = 1;
else
    y = 0;
end
```

McCulloch-Pitts-Neuron

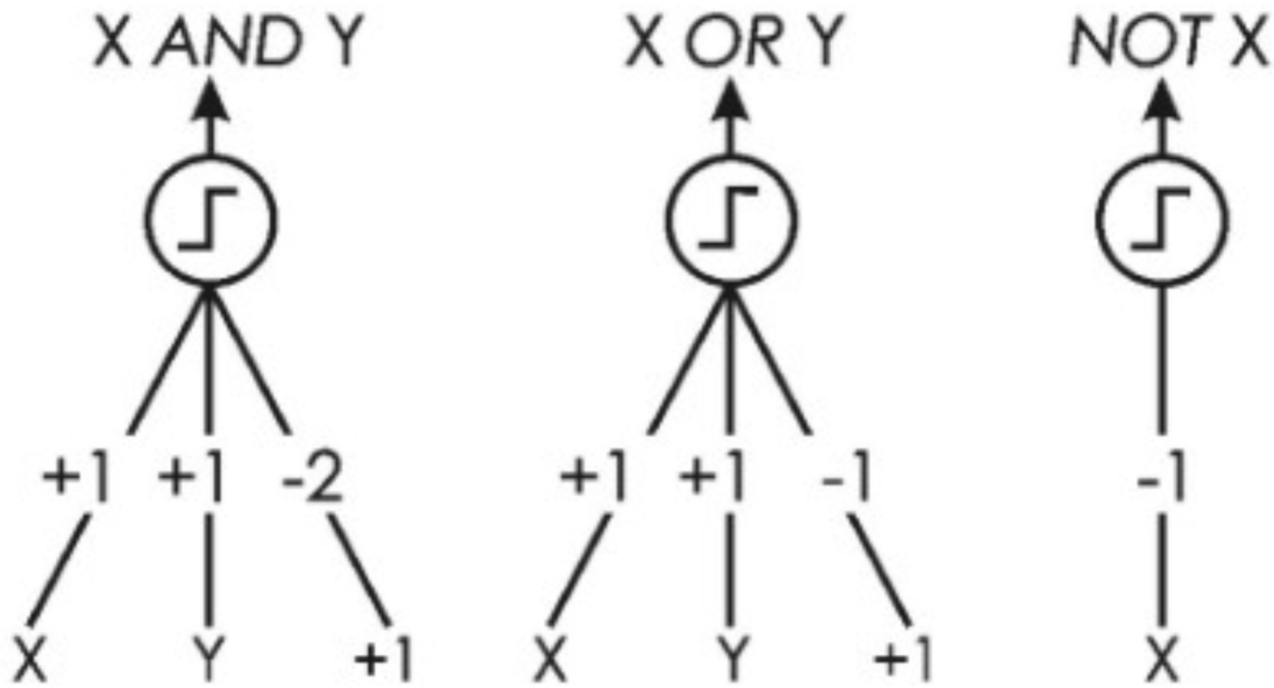
n binary input signals $x_1, \ldots, x_n$

threshold $\theta > 0$

$$f(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq \theta \\ 0 & \text{else} \end{cases}$$

McCulloch-Pitts (logic)

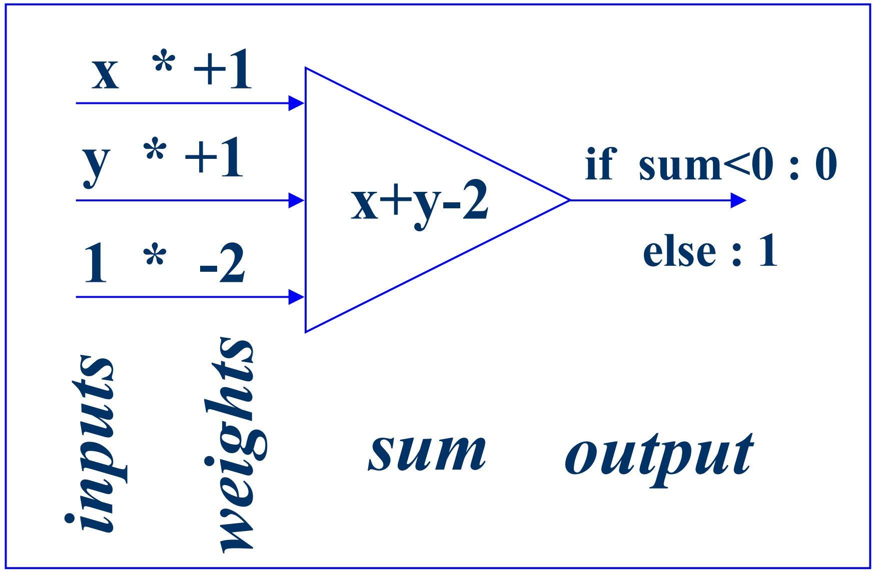
MP neurons are capable of logic functions



McCulloch-Pitts-Neuron

W.S. McCulloch & W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, 115-137.

This seminal paper pointed out that simple artificial "neurons" could be made to perform basic logical operations such as AND, OR and NOT.



Truth Table for Logical AND

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

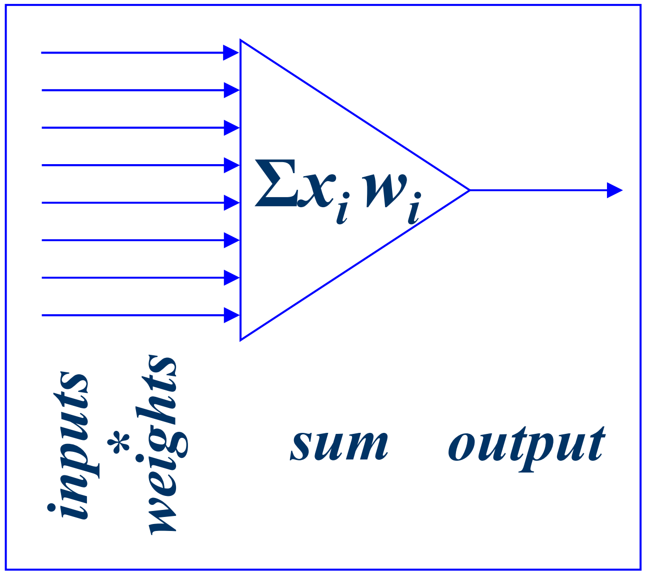
inputs *output*

Perceptron

Frank Rosenblatt (1962). *Principles of Neurodynamics*, Spartan, New York, NY.

Subsequent progress was inspired by the invention of *learning rules* inspired by ideas from neuroscience…

Rosenblatt's *Perceptron* could automatically learn to categorise or classify input vectors into types.



It obeyed the following rule:

If the sum of the weighted inputs exceeds a threshold, output 1, else output -1.

$$1 \text{ if } \Sigma\, input_i * weight_i > threshold$$

$$-1 \text{ if } \Sigma\, input_i * weight_i < threshold$$

Linear neurons

The neuron has a real-valued output which is a weighted sum of its inputs

weight vector

$$\hat{y} = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}$$

Neuron's estimate of the desired output

input vector

The aim of learning is to minimize the discrepancy between the desired output and the actual output

MLP: multilayer perceptron
many perceptrons organized into layers

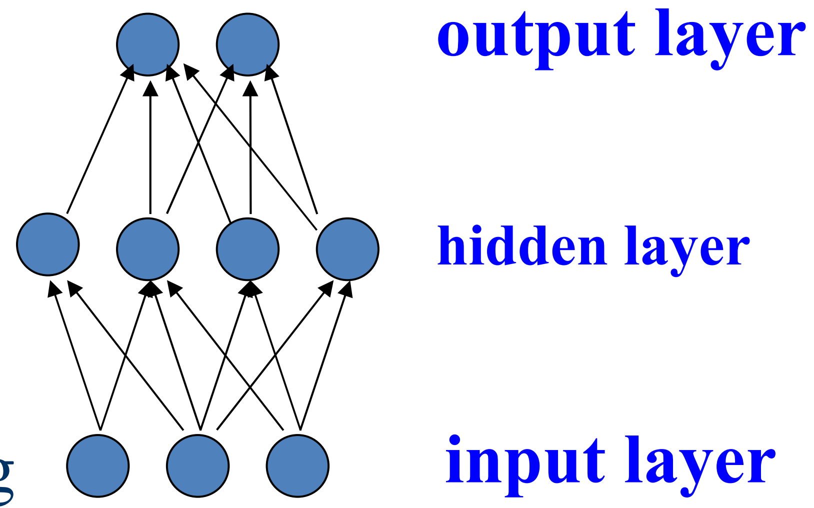
XOR

Nonlinear feature

Multiple layers

Arbitrary activation functions

MLP is the hello world of deep learning

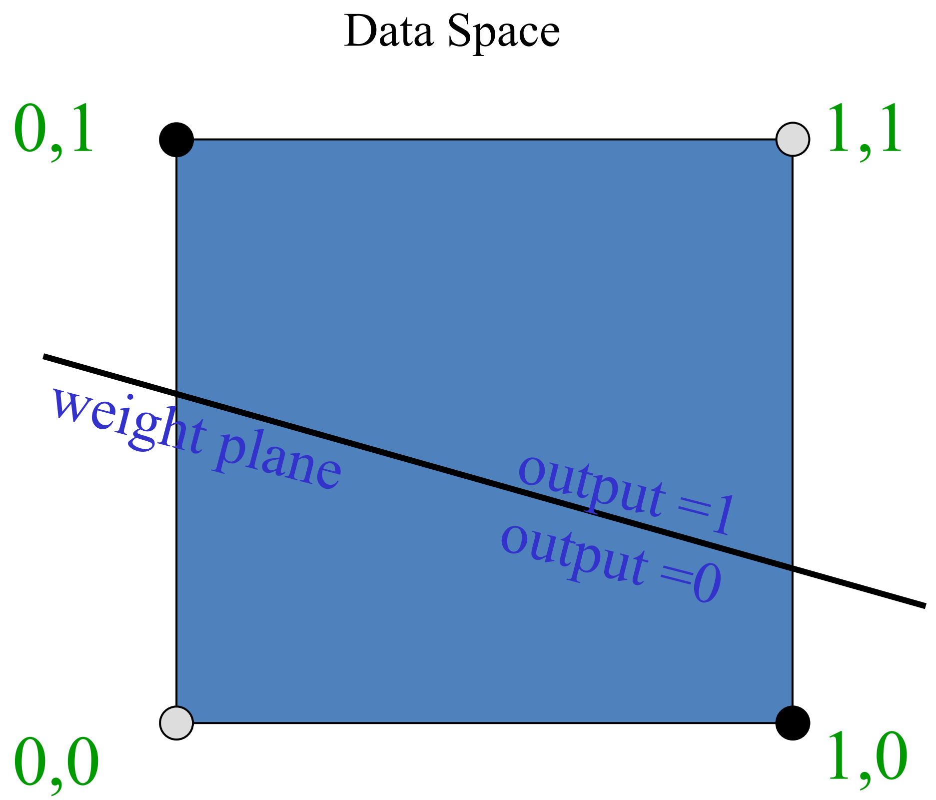


What perceptrons cannot do

The binary threshold output units cannot even tell if two single bit numbers are the same!

Same: (1,1) → 1; (0,0) → 1

Different: (1,0) → 0; (0,1) → 0

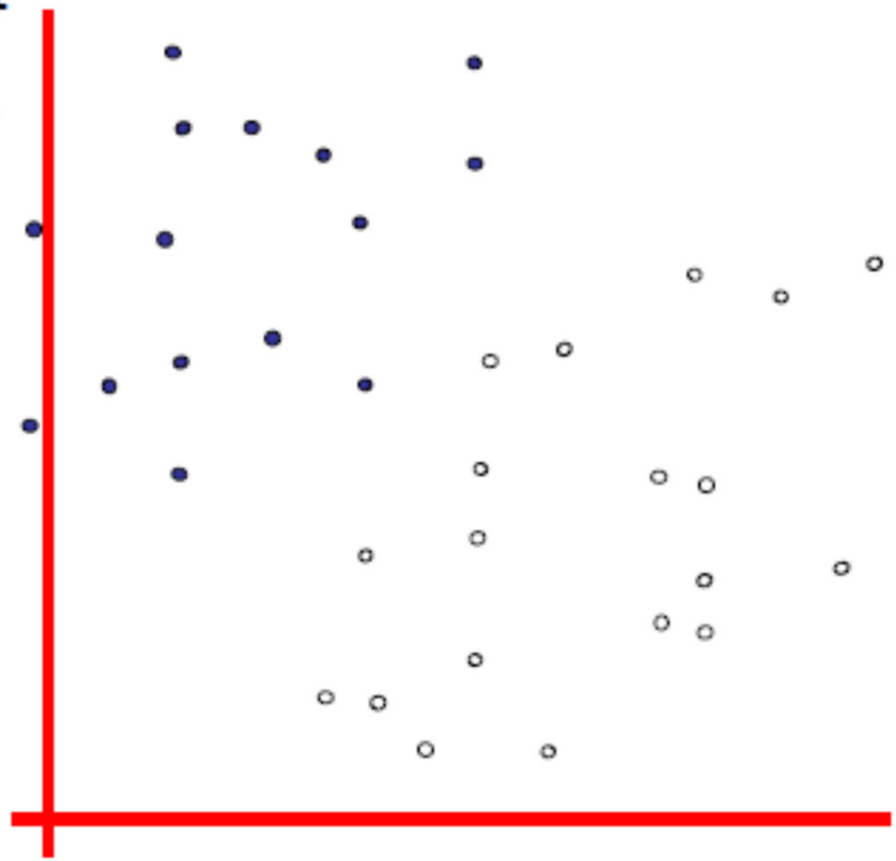


The positive and negative cases cannot be separated by a plane

$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

• denotes +1

° denotes -1

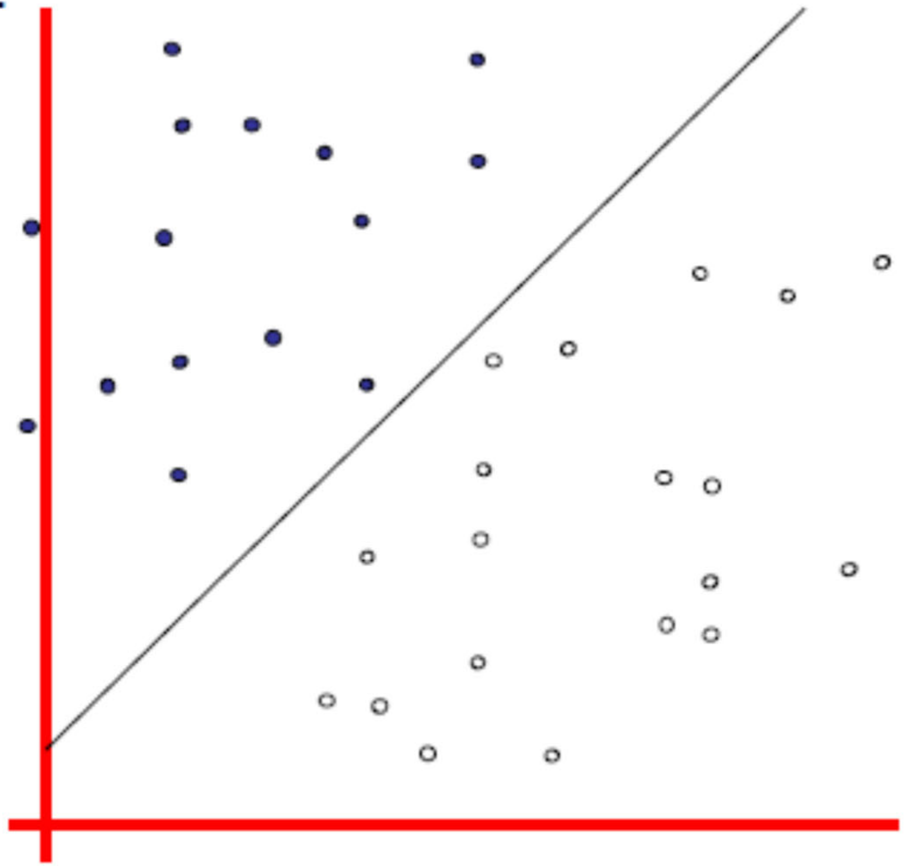


How would you classify this data?

$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

• denotes +1

° denotes -1

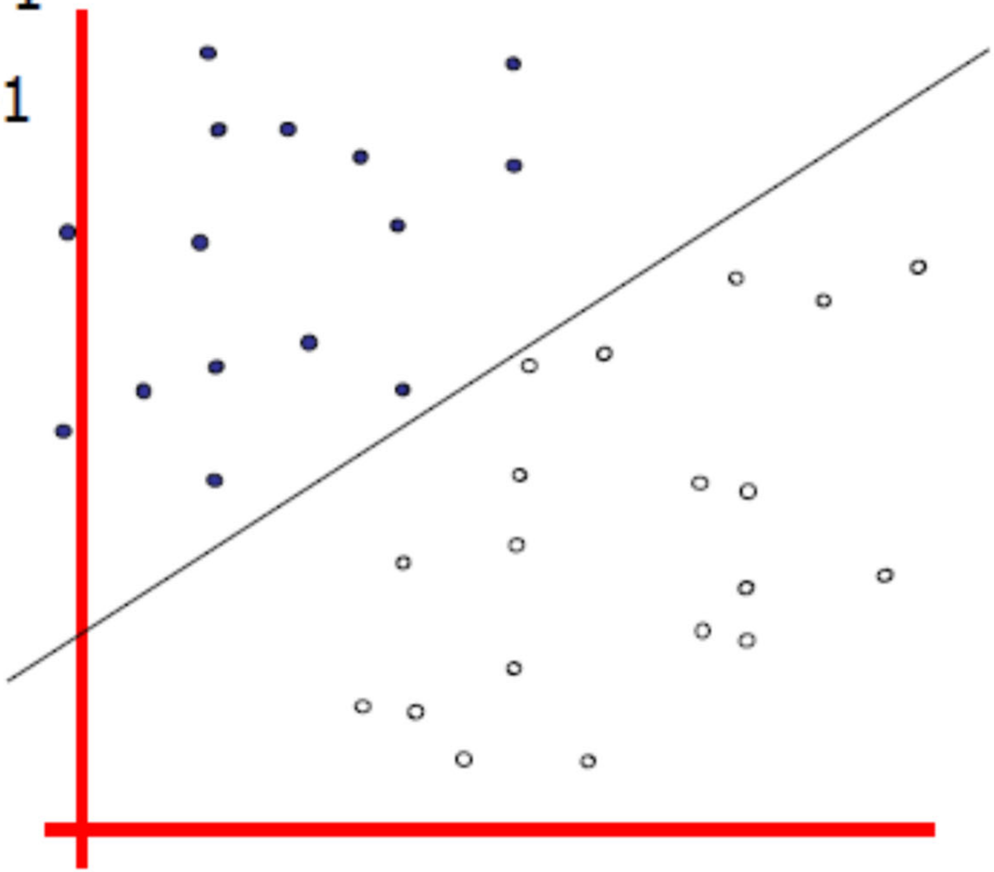


How would you classify this data?

$$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w}.\ \mathbf{x} - b)$$

• denotes +1

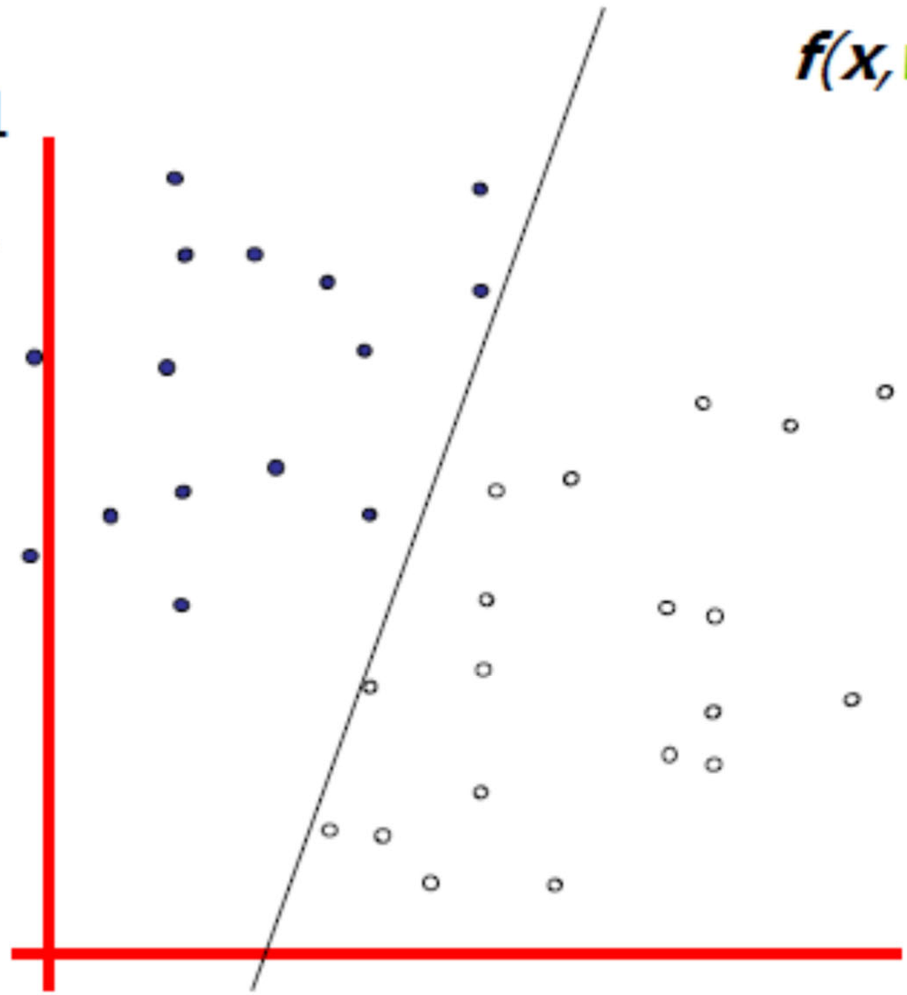
° denotes -1



How would you classify this data?

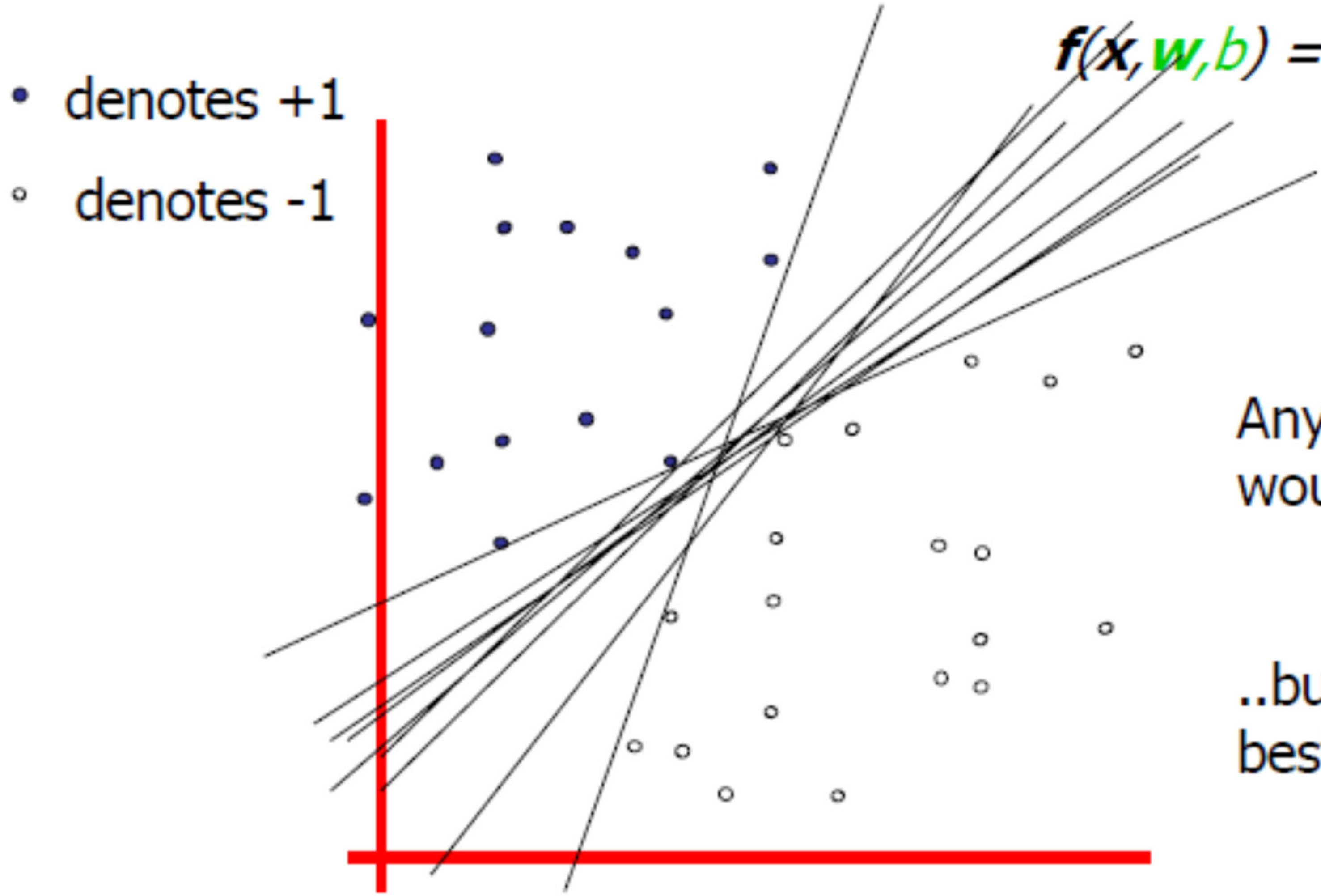
$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

- denotes +1
- denotes -1



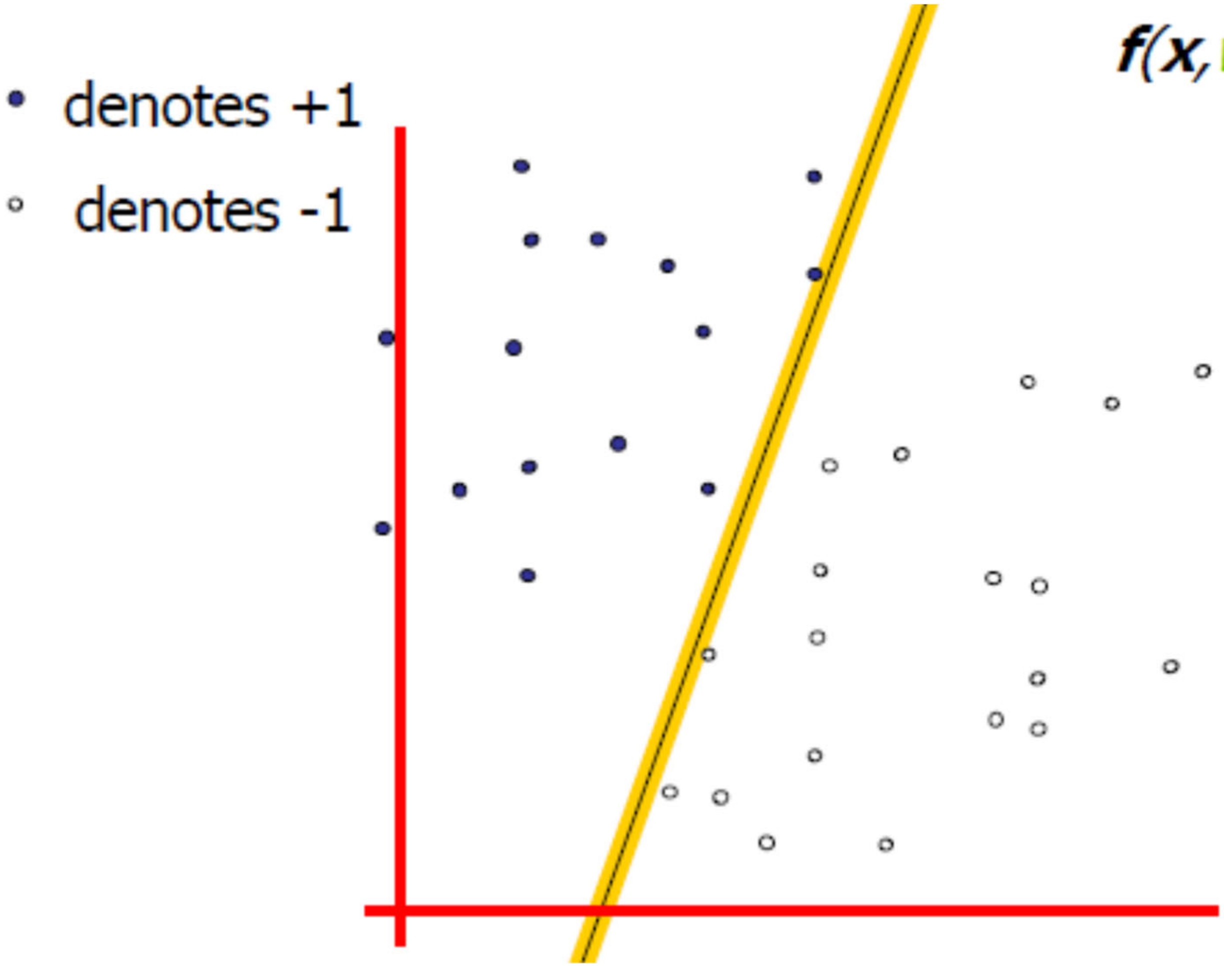
How would you classify this data?

$f(x, w, b) = sign(w \cdot x - b)$



Any of these would be fine..

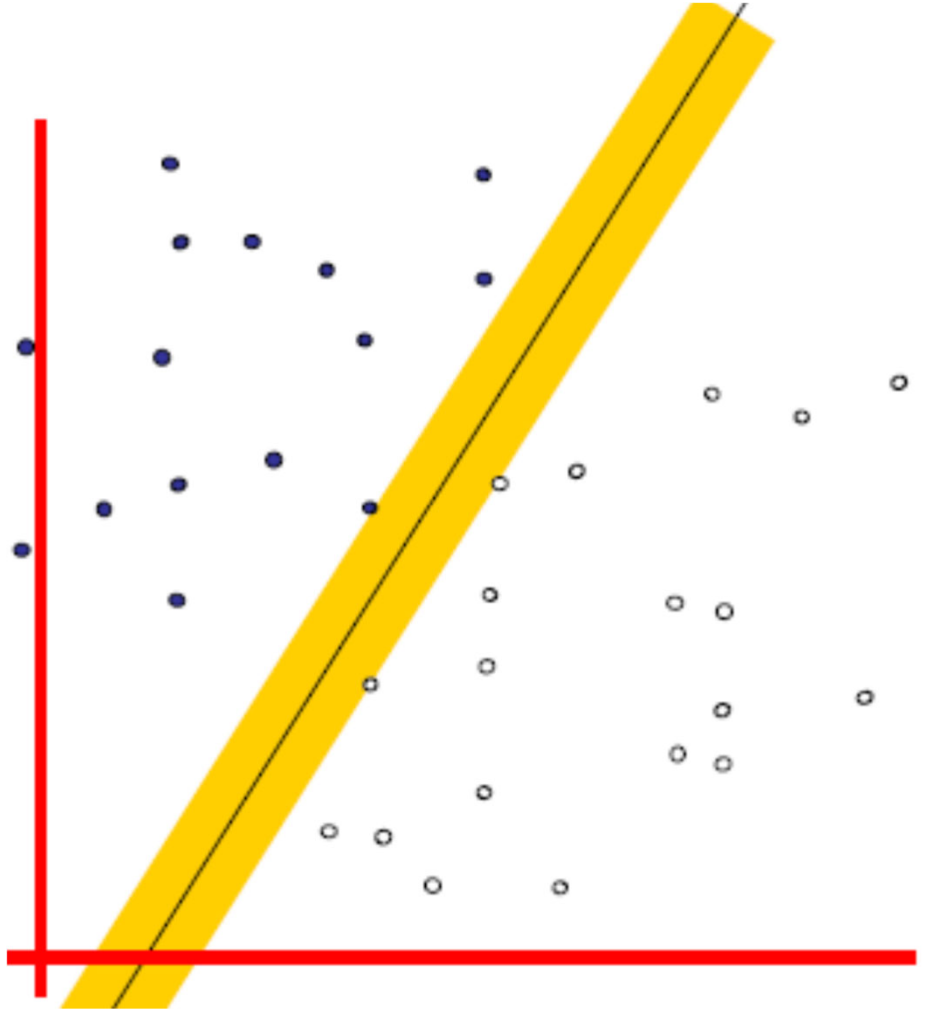
..but which is best?



$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Linear Classifier: Perceptron - SVM

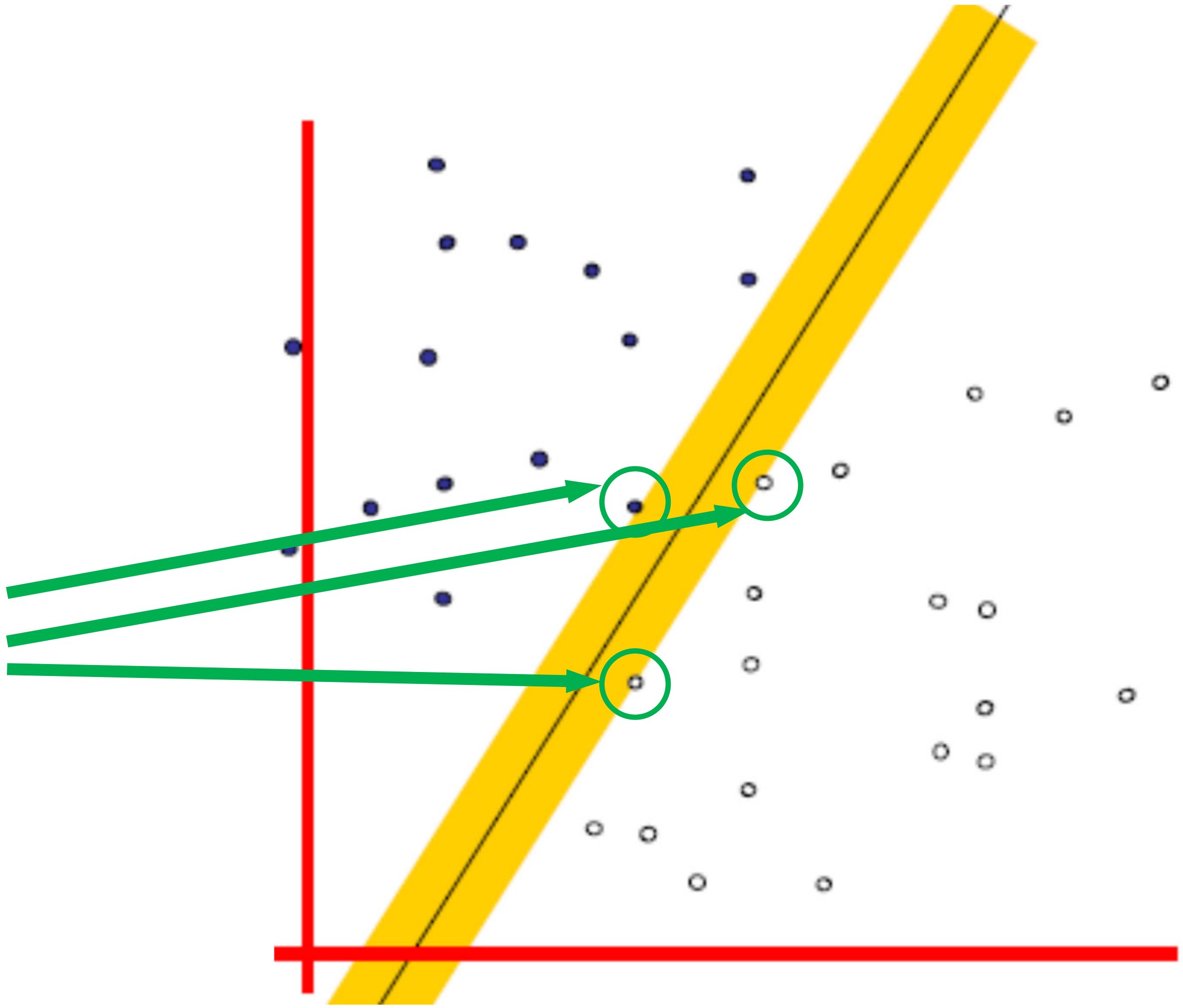


The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of Support Vector Machine

Linear Classifier: Perceptron - SVM

Support Vectors:
datapoints (vectors) that the margin pushes up against



The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of Support Vector Machine

Limitations of Linear Classifiers

- Linear Learning Machines cannot deal with
  - Non-linearly separable data
  - Noisy data

Overcome Limitations

- **Neural networks solution:**

multiple layers of thresholded linear functions – multi-layer neural networks.

Learning algorithms – back-propagation.

Deep Learning …

- **SVM solution:** kernel representation.

Approximation-theoretic issues are **independent** of the learning-theoretic ones.

Learning algorithms are decoupled from the specifics of the application area, which is encoded into design of kernel.

Overcome Limitations

Neural networks solution:

multiple layers of thresholded linear functions – multi-layer neural networks.

Learning algorithms – back-propagation. Deep Learning

	Artificial neurons in ANNs	Biological neurons
Morphology	Nodes as point neurons, each node can be a multilayered perceptron, symmetric tree	Complex dendritic tree with several branch points, not symmetric
Sparsification	Dropout layers	Inputs are distributed across the tree
Integration	Activation functions used in each node (e.g., sigmoid, tanh, ReLU)	Dendritic spikes (Na^{+}, Ca^{2+}, NMDA, dCaAPs, back-propagating action potentials) or passive (sublinear) integration
Learning	Backpropagation-of-error algorithm (most effective)	Cooperative synaptic plasticity (local), plasticity of excitability (dendritic/somatic), bursting
Homeostasis	Regularization of weights/normalization of layers	The total synaptic strength of all synapses (in a branch, neuron, or network) scales up/down
Rewiring	Not typically used, would amount to dynamically changing the graph	Synapses turn over dynamically, and only those that fire with neighbors are stabilized

Chavlis & Poirazi, Current Opinion in Neurobiology (2021)