MASTER OF SCIENCE
IN ENGINEERING

Week 1: Perceptron

TSM_DeLearn

First Neural Nets and a Historical Perspective

Jean Hennebert
Martin Melchior

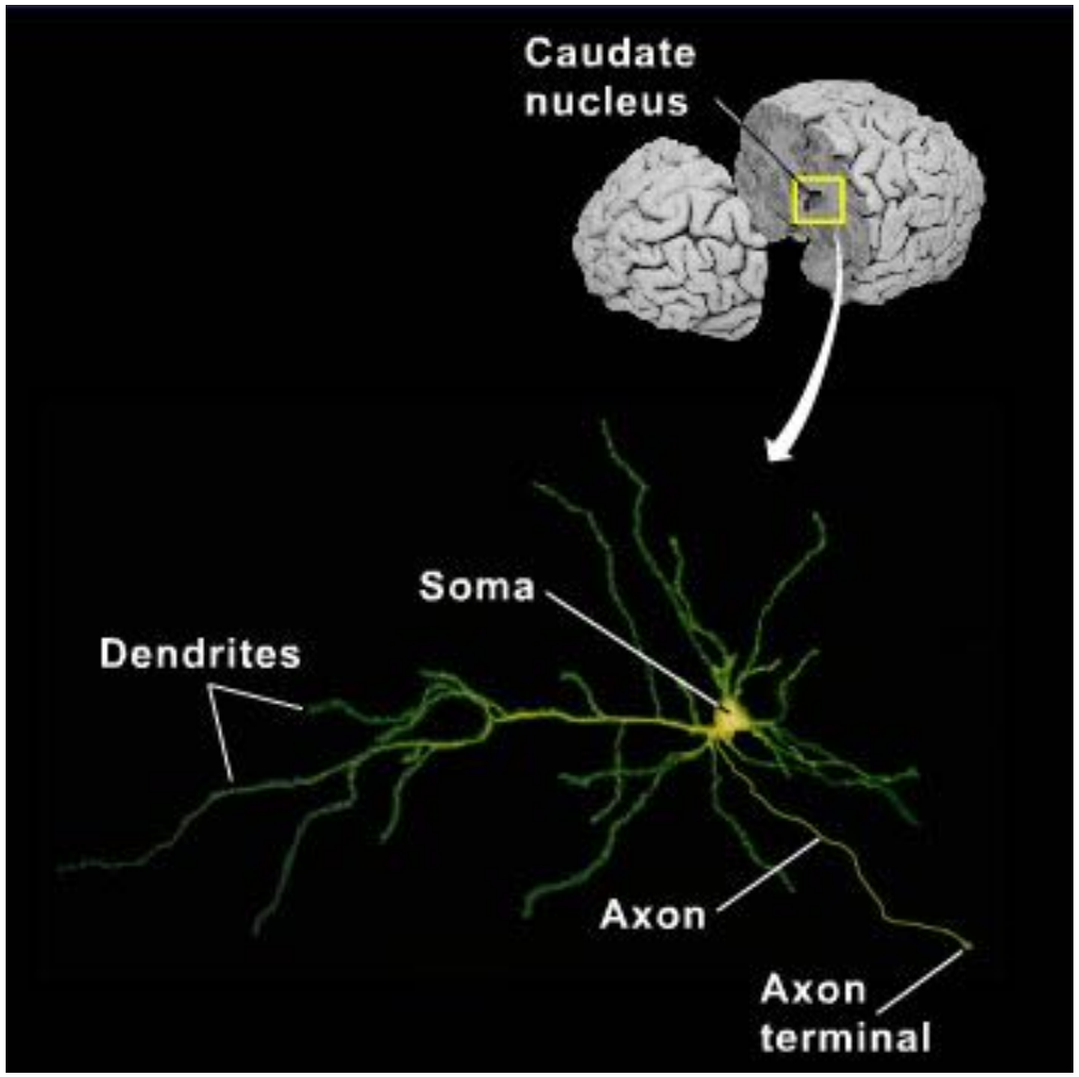
Overview

Biological Neural Systems

Artificial Neuron

Artificial Neural Nets

Biological Neural Systems



Biological Neuron

Biological Neural Nets

Neuro-Science and Artificial Neural Networks

Biological Neuron

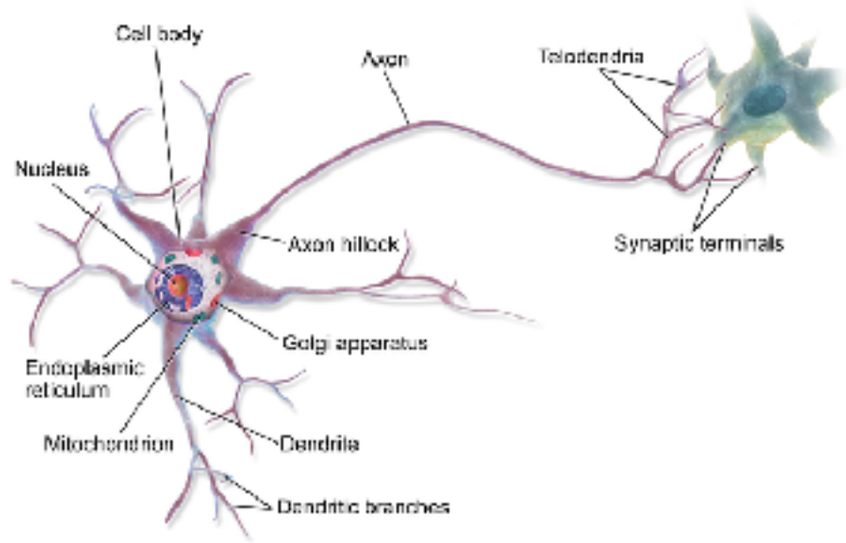
- **Biological neurons** composed of
  - cell body
  - dendrites: many branching extensions
  - axon: very long extension that splits off at its tip into many branches called synaptic terminals

- Composed in a **network** (such as the brain) by synaptic terminals of one neuron connected to dendrites of other neurons.

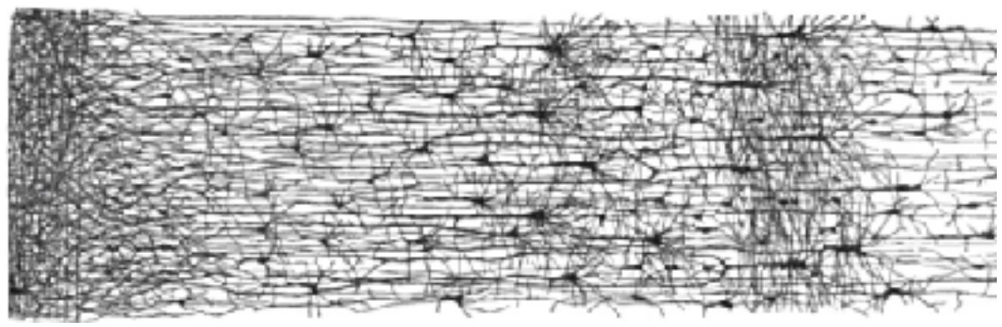
- Electrical impulses (‘**signals**’) are sent from other neurons via these synapses.

- If a neuron receives a sufficient number of signals from other neurons within a few milliseconds, it is excited and fires its own signals (**activation**).

Biological Neuron and its connections to other neurons through synapses
(https://en.wikipedia.org/wiki/Neuron)



Biological Neuron and its connections to other neurons through synapses
(https://en.wikipedia.org/wiki/Cerebral_cortex)



Connectivity in Biological Neural Nets

- The connectivity in biological neural systems is huge.
- Huge number of synapses in human brain
  - Number of neurons: ~ 10^{11}
  - Number of connections per neuron: ~10^{4}

  } ~10^{15} synapses

- Synaptic Pruning
  - During adolescence the brain continues to grow, but the overall number of neurons and synapses are reduced by up to 50% to build up more complex and more efficient structures.
- Most neurons in the human brain are designed to last a lifetime.
  - They are only able to divide (neurogenesis) during fetal development and for a few months after birth.
  - The only area of the brain where neurogenesis continues throughout life is the hippocampus, an area essential to memory encoding and storage.

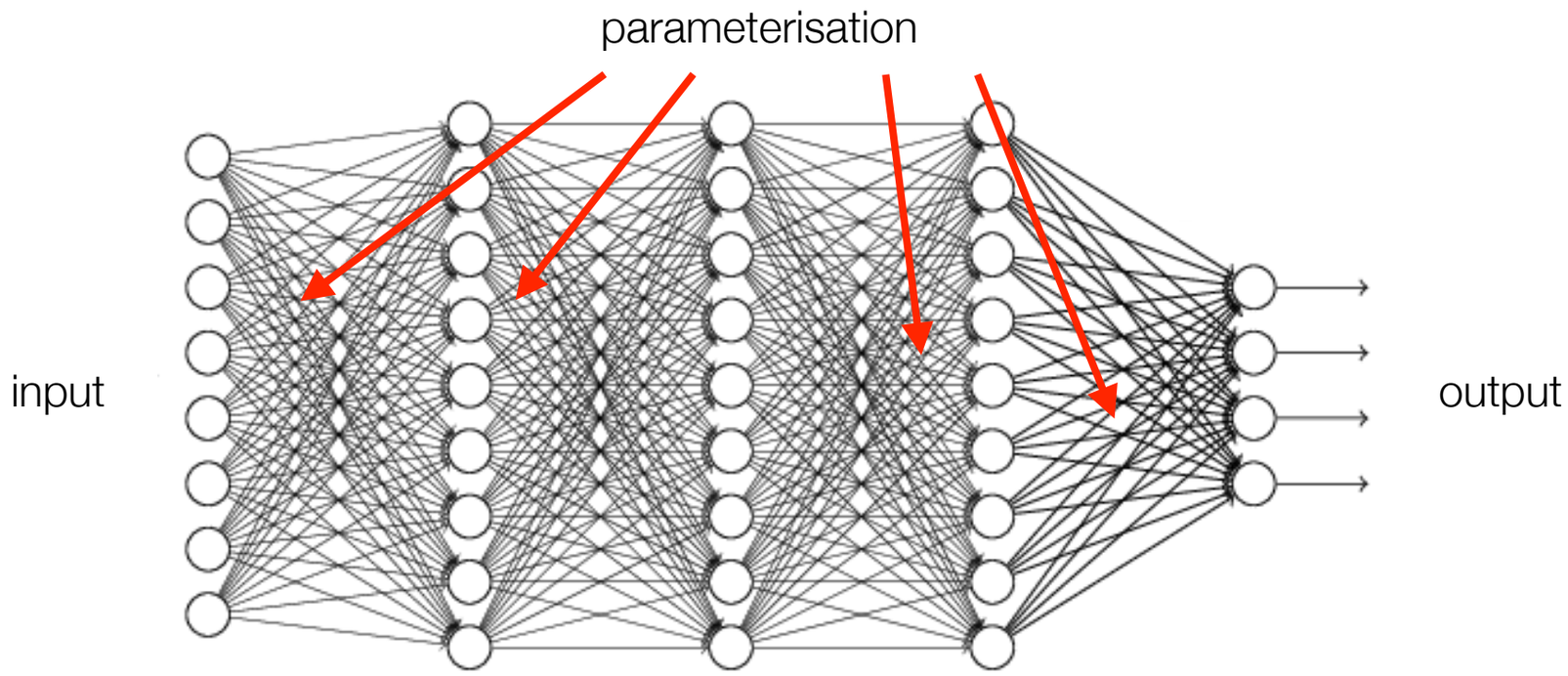
Artificial Neural Networks and Neuro-Science

- Historically: Neuro-Scientific Perspective
  - Artificial neural networks inspired by biological brain to model intelligent behaviour. Reverse-engineer the computational principles behind the brain.
  - Understand the brain and the principles underlying human intelligence.
  - Understand the principles of how learning may happen in the brain.

- Nowadays: Modern Machine Learning Perspective
  - Deep Learning goes beyond the neuro-scientific perspective and appeals to a more general principle of learning with multiple levels of composition.
  - Generally, no claim to model the biological function directly.

Key Elements of Artificial Neural Nets

- **Artificial Neural Nets** are composed of **Artificial Neurons** — relatively simple building blocks that are capable of performing complex tasks when suitably combined.
- The capability of performing the tasks is encoded in the structure and the parameters of the network.
- The network is trained to perform the task from **training data** by applying a **learning algorithm** that adjusts the networks parameters.
- The networks can be trained to perform a wide range of tasks such as predicting outputs from given inputs.

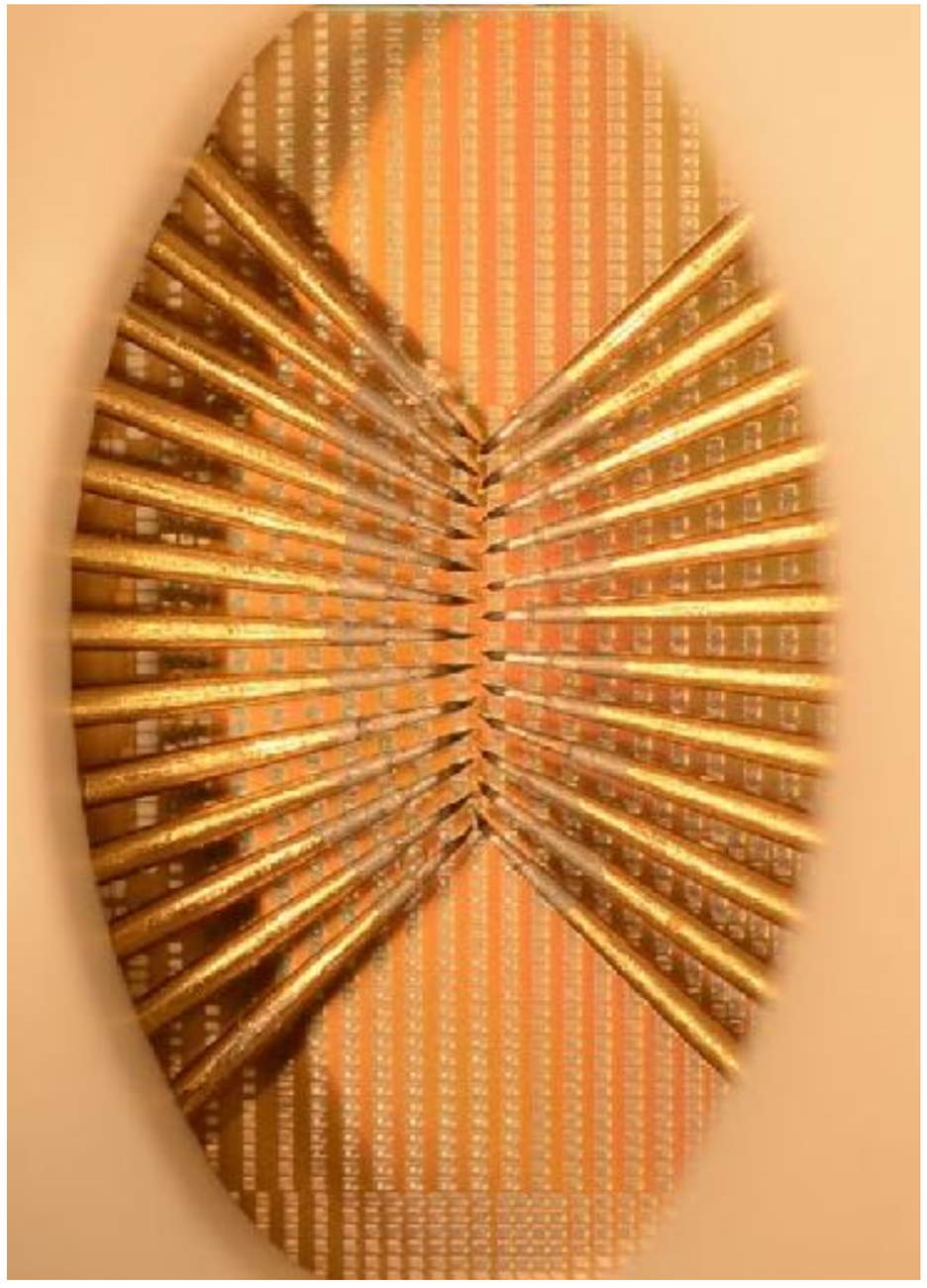


Artificial Neuron

McCulloch-Pitts Neuron

Rosenblatt's Perceptron

Perceptron Convergence Theorem

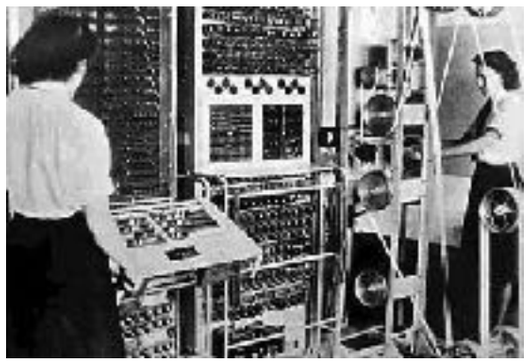


(phase-change neurons developed by IBM Research)

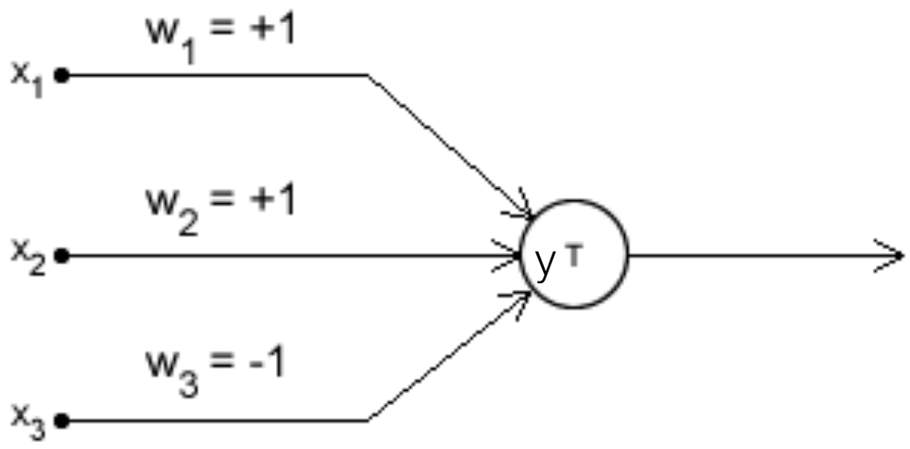
McCulloch-Pitts Neuron (1943)

First artificial neuron as a model for the activation of a neuron

- Input signals: either 0 or 1: $x_k = 0, 1 \quad (1 \leq k \leq n)$
- Weighted with +1 (excitatory) or -1 (inhibitory): $w_k = \pm 1 \quad (1 \leq k \leq n)$
- Sum of all input signals: $S = \sum_{k=1}^{n} w_k x_k$
- Output signal: $y = \begin{cases} +1 & (S \geq \theta) \\ 0 & (S < \theta) \end{cases}$



1943 - in the era of World War II and where the first computers were constructed (here: *Colossus*)



What kind of tasks can be performed with such neurons?

Representational Power of Networks with McCulloch-Pitts Neurons

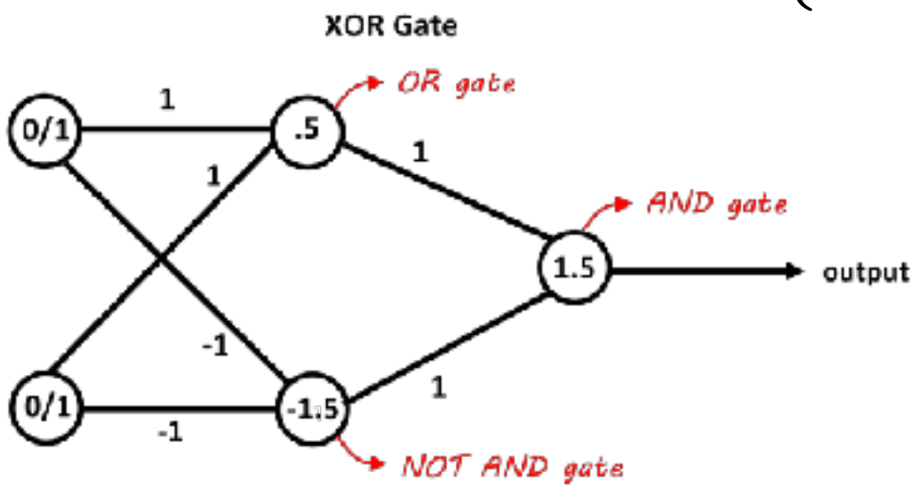
- McCulloch and Pitts showed (see https://link.springer.com/article/10.1007%2FBF02478259) that, in principle, any logical or arithmetic function can be computed with networks of such neurons.

- Examples
  - AND: $y = H(x_1 + x_2 - 2)$
  - OR: $y = H(x_1 + x_2 - 1)$
  - XOR: $y = H(H(x_1 + x_2 - 0.5) + H(1.5 - x_1 - x_2) - 1.5)$

where we have used the Heaviside-function $H(z) = \begin{cases} 1 & (z \geq 0) \\ 0 & (z < 0) \end{cases}$

AND

x1	0	1	0	1
x2	0	0	1	1
z	-2	-1	-1	0
y	0	0	0	1



For a given problem at hand: **How should the structure and the weights be chosen?**

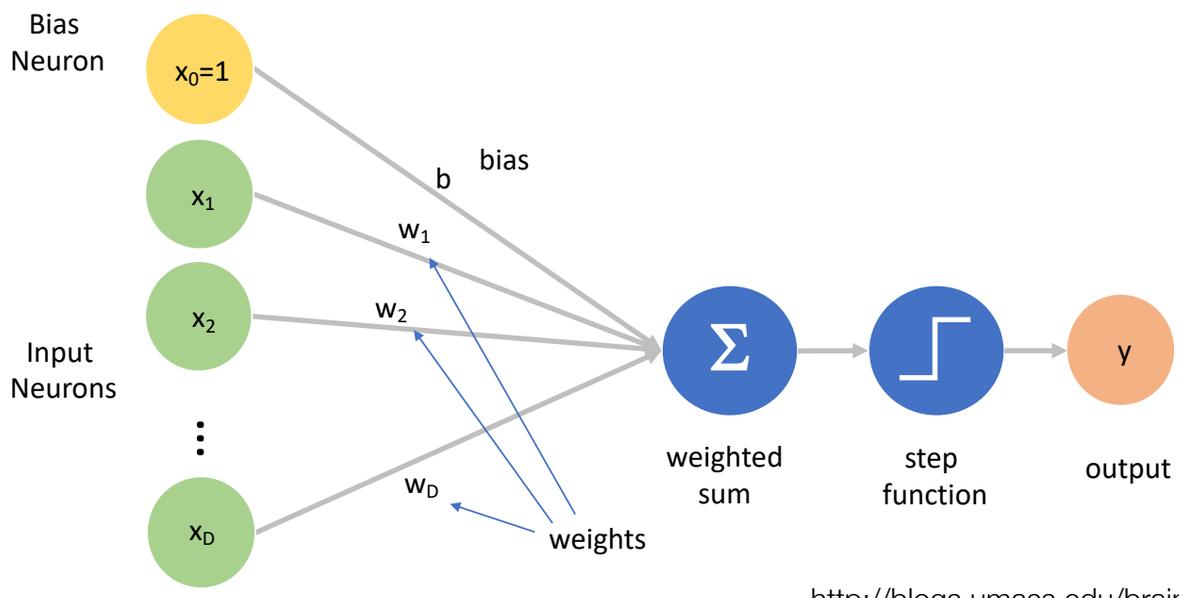
Rosenblatt's Perceptron (1958)

Single Perceptron, also referred to as Linear Threshold Unit (LTU):

- Inputs and outputs are *real* numbers.
- Each input connection associated with a *real* weight.
- Activity computed by applying the Heaviside function to the weighted sum of the inputs; the activation threshold incorporated in form of a bias term (*b*).

$$y = H(\sum_{k=1}^{n} w_k x_k + b) \text{ for } x_k, w_k, b \in \mathbb{R}$$

The output can take on two distinct values (0 or 1).



What kind of tasks can be performed with such neurons?

http://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf

Capabilities of a Single LTU

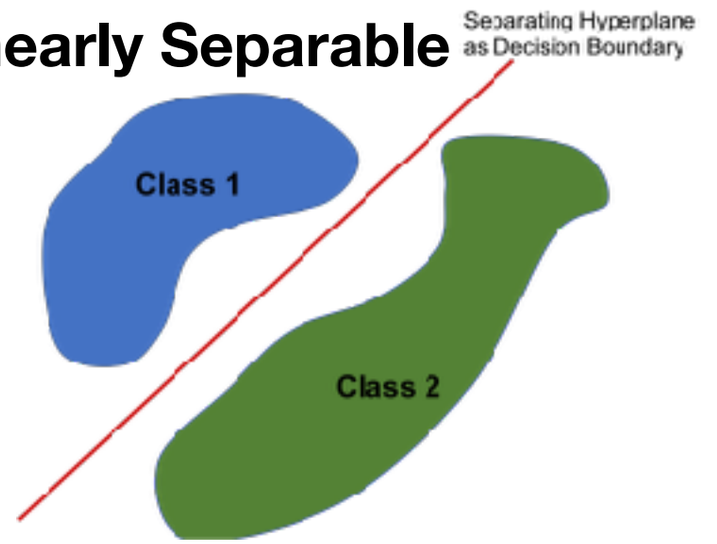
- Output can assume two distinct values: 1 or 0.
  —> Useful for **binary classification problems**.
- Decision boundary is given by

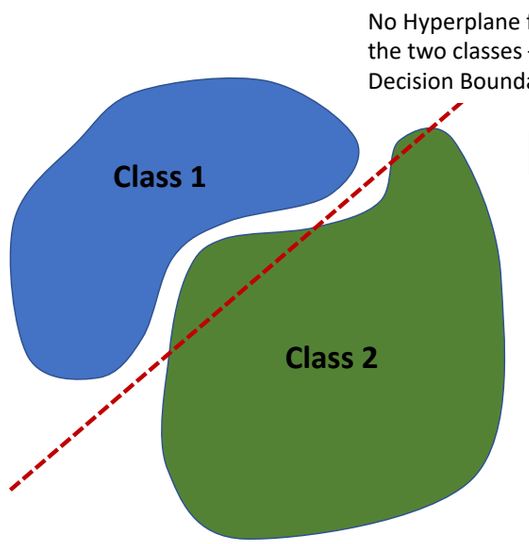
$$H_{w,b} = \left\{ \mathbf{x} \in \mathbb{R}^D \,|\mathbf{w} \cdot \mathbf{x} + b = 0 \right\}$$

  for suitably adjusted, but fixed ***w*** and b. Defines a hyperplane in $\mathbb{R}^D$.
- Suited for **linearly separable binary classification** problems.

$$\mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix}$$

Linearly Separable

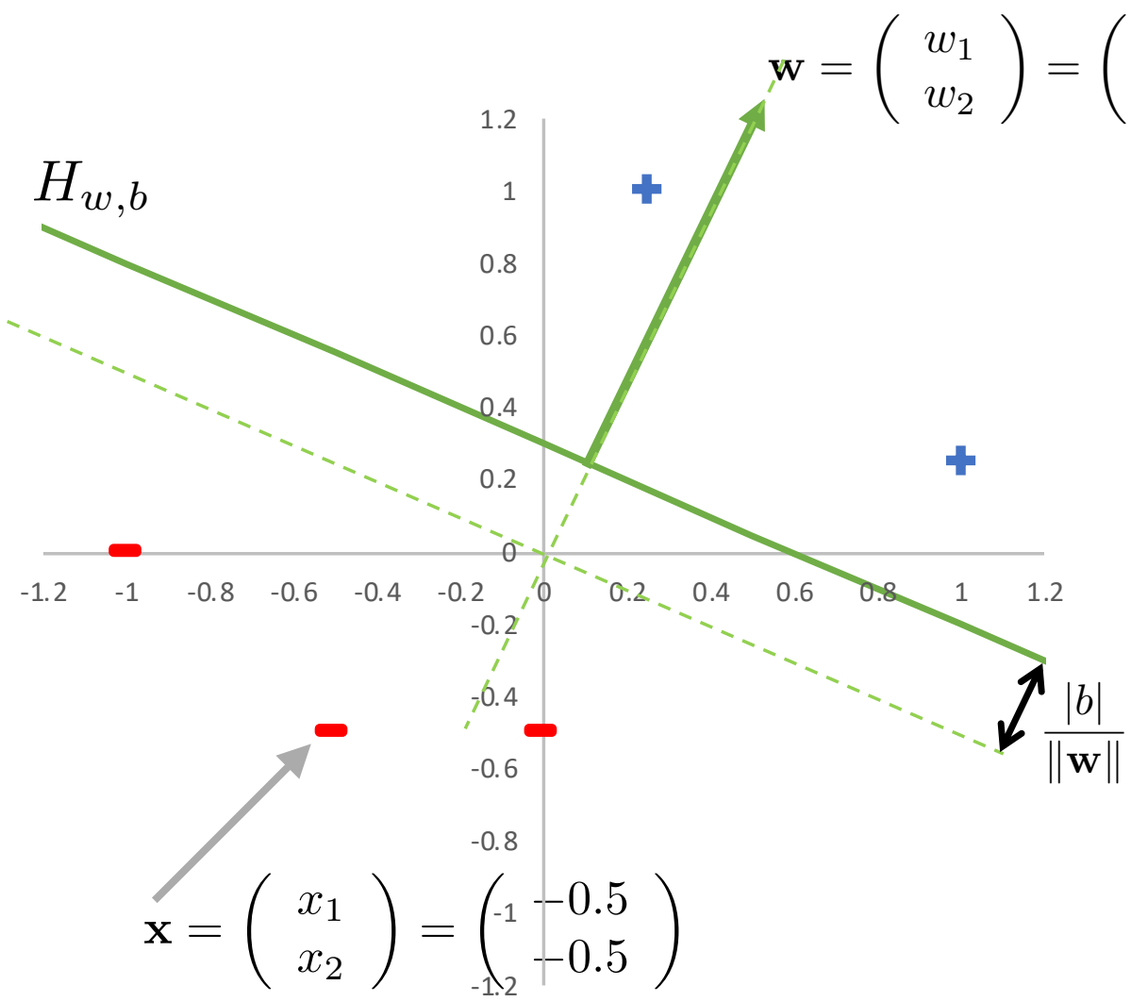




Not Linearly Separable

How to learn the weights for a specific problem at hand?

Example with 2D Input Data



Data Points: -,+

Decision Boundary

$$H_{w,b}: \ \mathbf{w} \cdot \mathbf{x} = w_1 x_1 + w_2 x_2 + b = 0$$

$$\Leftrightarrow x_2 = -\frac{b}{w_2} - \frac{w_1}{w_2} x_1$$

Example

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1.0 \end{pmatrix}, \ b = -0.3$$

$$H_{w,b}: \ \mathbf{w} \cdot \mathbf{x} = 0.5\, x_1 + x_2 - 0.3 = 0$$

$$\Leftrightarrow x_2 = 0.3 - 0.5\, x_1$$

Perceptron Learning Algorithm

Labelled Data: $\{(\mathbf{x}^{(i)}, y^{(i)}) \mid i = 1, \ldots, N\}$
where $\mathbf{x}^{(i)}$ are d-vectors

Learning algorithm:

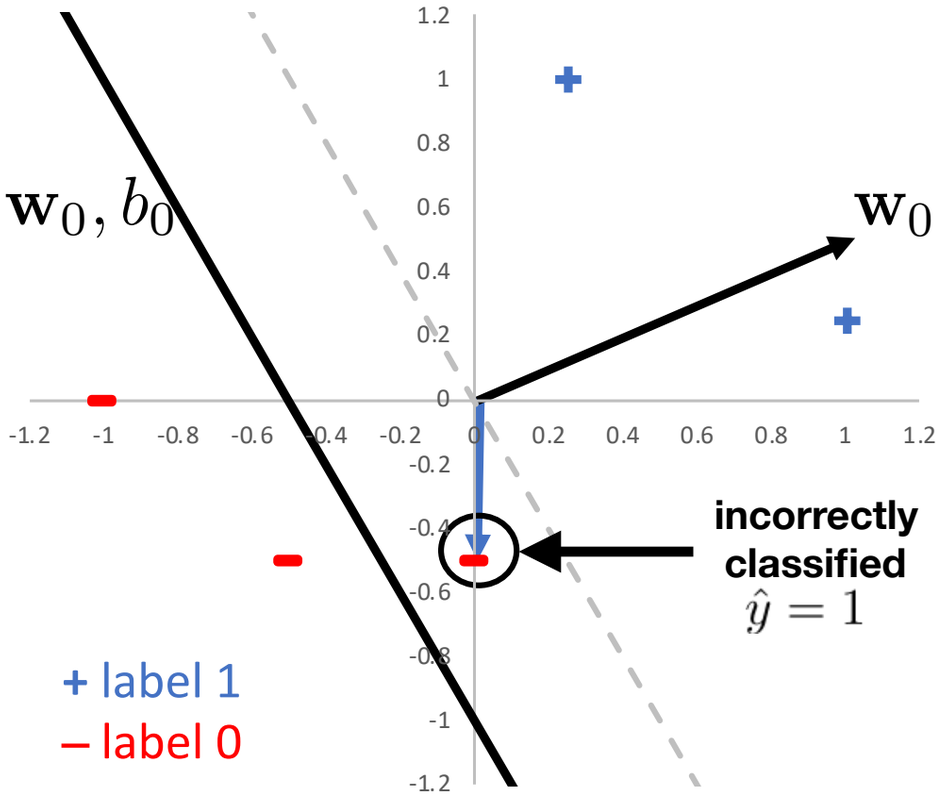
1. Initialise parameters — zero or small random number
2. Iterate by updating weights according to
   - A. Pick sample — $(\mathbf{x}^{(i)}, y^{(i)})$
   - B. Compute predicted values: — $\hat{y}^{(i)} = H(\mathbf{w} \cdot \mathbf{x}^{(i)} + b)$
   - C. Parameter update rule: — only update if $\hat{y}^{(i)} \neq y^{(i)}$:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha \cdot (\hat{y}^{(i)} - y^{(i)})\mathbf{x}^{(i)} \\ b &\leftarrow b - \alpha \cdot (\underbrace{\hat{y}^{(i)} - y^{(i)}}_{\text{this term is zero or one!}}) \end{aligned}$$

Learning rate: $\alpha > 0$ (e.g. $\alpha = 0.1$) is a parameter of the algorithm

Learning Algorithm Explained Geometrically

Before Update

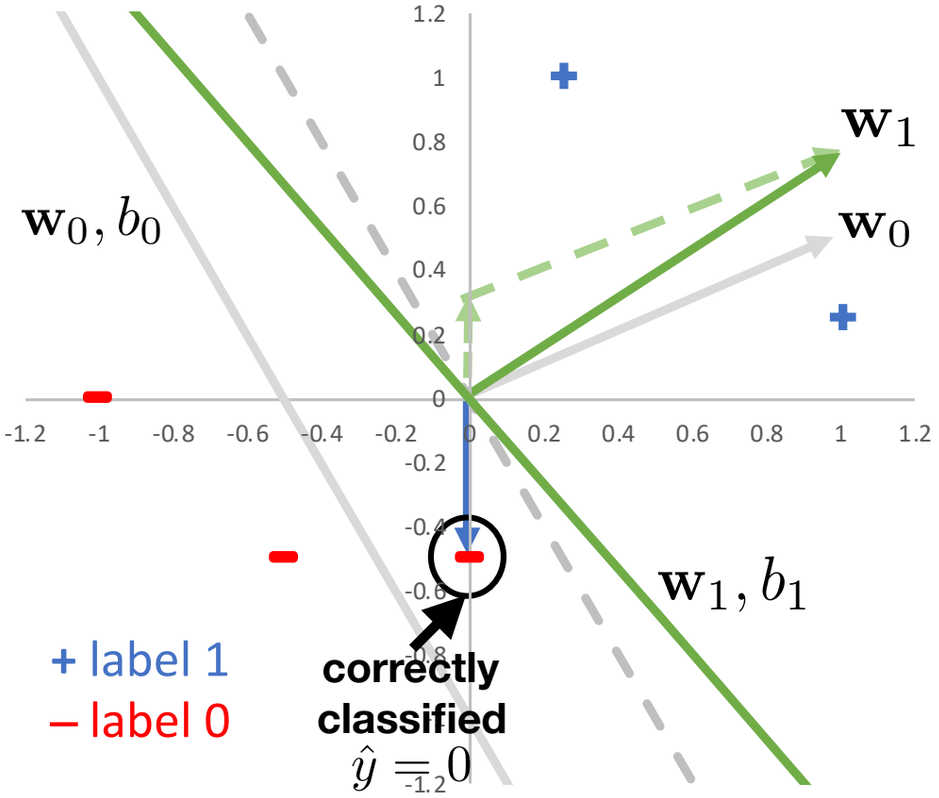


Decision boundary (**black line**) is parametrised by

$$\mathbf{w}_0 = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, b_0 = 0.5$$

The point $(0, -0.5)$ is classified incorrectly as **+** since it is above the decision boundary.

After Update



With a learning rate $\alpha = 0.5$ the new parameters are

$$\mathbf{w}_1 = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} - \alpha \begin{pmatrix} 0 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.75 \end{pmatrix}$$

$$b_1 = b_0 - \alpha = 0$$

which leads to the new decision boundary (**green line**) which correctly classifies all points.

Perceptron Learning Algorithm

- The learning rule searches for a weights vector that defines a hyperplane that separates the points associated with the two classes. This is only possible for linearly separable input sets.

- In case of wrongly classified points, the weights update rule leads to a correction of the weights such that the hyperplane is tilted to (rather) bring the wrongly classified points to the correct side.

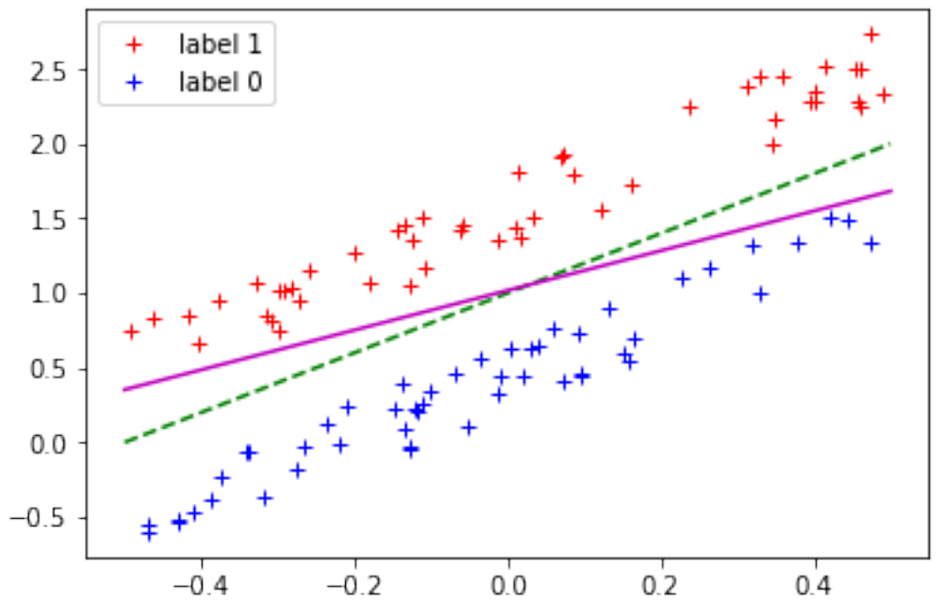
Perceptron Convergence Theorem

Perceptron Learning Algorithm converges to a weights vector and bias that separates the two classes - provided that the two classes are linearly separable.

For a proof, see for instance https://www.pearsonhighered.com/assets/samplechapter/0/1/3/1/0131471392.pdf

- The solutions obtained for linearly separable inputs is not unique and not optimal. The optimal solution (with the widest separating corridor) is known as the linear support vector machine (SVM).

- The perceptron may be a reasonable model also for problems that are not linearly separable. However, the application of the Perceptron Learning Algorithm does not converge in this case.



Solution found by perceptron learning rule

Solution with the optimal widest corridor

Artificial Neural Nets

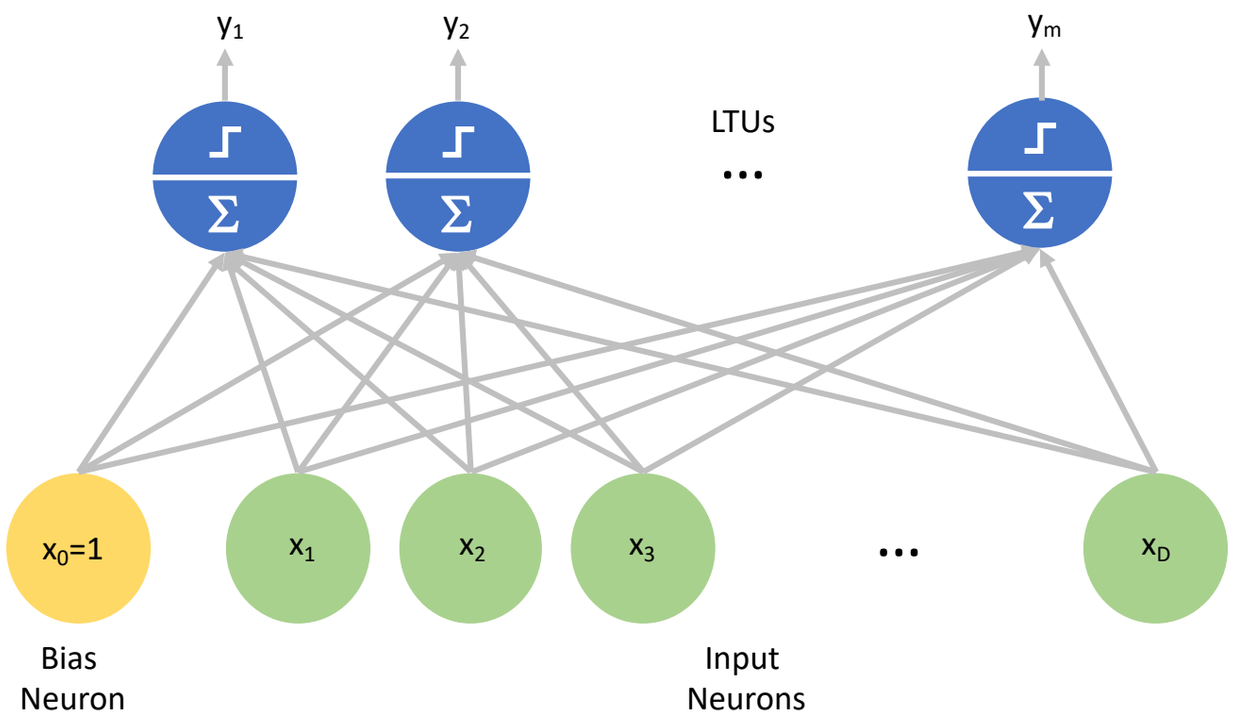
Single Layer LTU

XOR Problem

Multi-Layer Perceptron

A first Neural Network: Single Layer LTUs

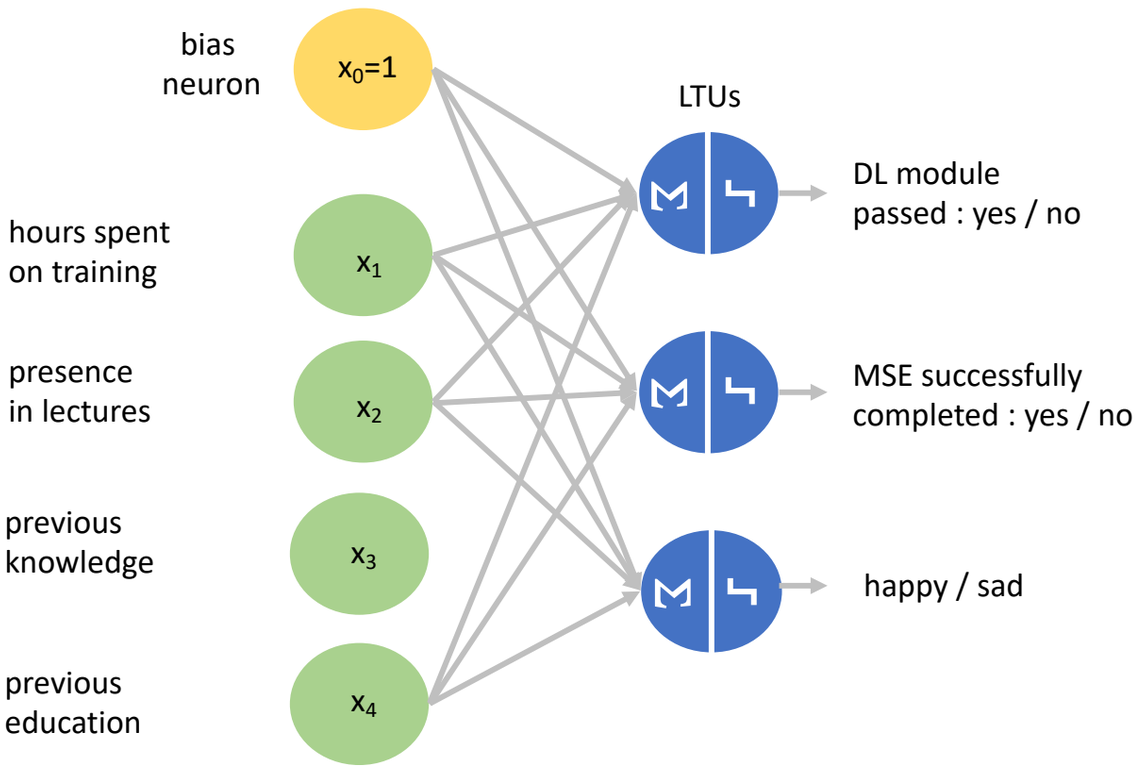
- The inputs are often represented using special *passthrough* neurons called **input neurons**.
- Each neuron is connected to all the inputs, the layer is **fully connected**.
- Generally, an extra bias feature is added ($x_0 = 1$) — often represented as a **bias neuron** that just outputs 1.



Possible Applications of Single Layer LTUs

- A single layer LTU with *m* units can classify instances simultaneously into *m* different binary classes, i.e. is a multi-output classifier that can perform multiple tasks.
- Apply the learning algorithm independently to each output.

Example: Students' Performance and Happiness



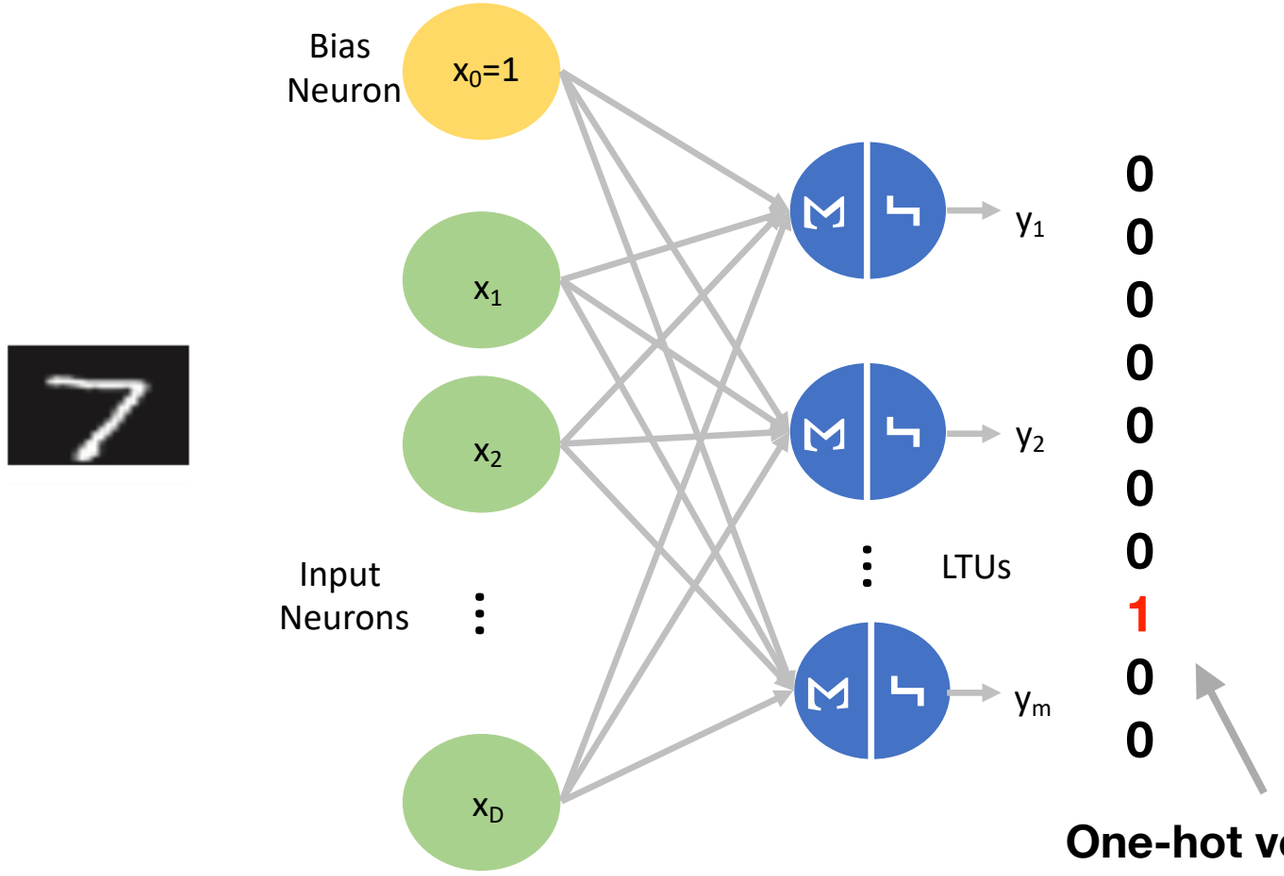
Some of the outputs may be related to each other, others not.

Student	Training	Presence	Prev Knowledge	Prev Education	Module passed	MSE successfully completed	Happy / sad
a	90	100%	medium	ec.	1	1	1
b	60	85%	high	c.s.	1	1	1
c	70	100%	medium	c.s.	1	1	0
d	40	30%	high	c.s.	0	1	1
e	10	90%	low	sys. eng.	1	1	1
f	85	50%	high	c.s.	1	1	1
g	30	20%	low	sys. eng.	0	0	1
h	70	30%	high	el.eng.	1	0	0
…	…	…	…	…			

Classifier for *m* Mutually Exclusive Classes?

- In the previous example we had three binary output quantities.
- This is not the same as the situation where we want mutually exclusive outputs: For each sample, only one output is 1 and all the others 0.

Example: Digits classification



Some digits may also be interpreted as ‘1’

One-hot vector: One output for each of the digits. Only one of the entries is non-zero.

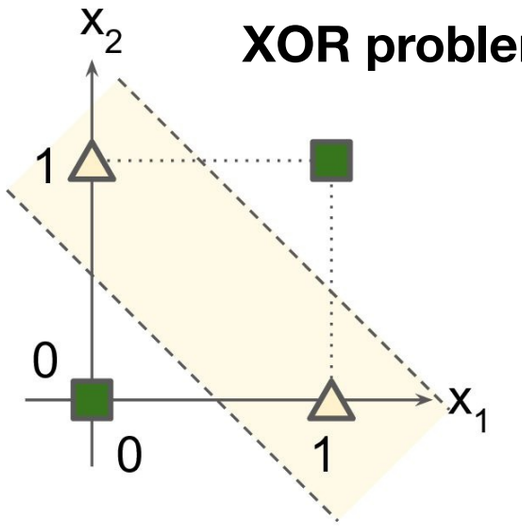
Representational Power of Perceptron and XOR Problem

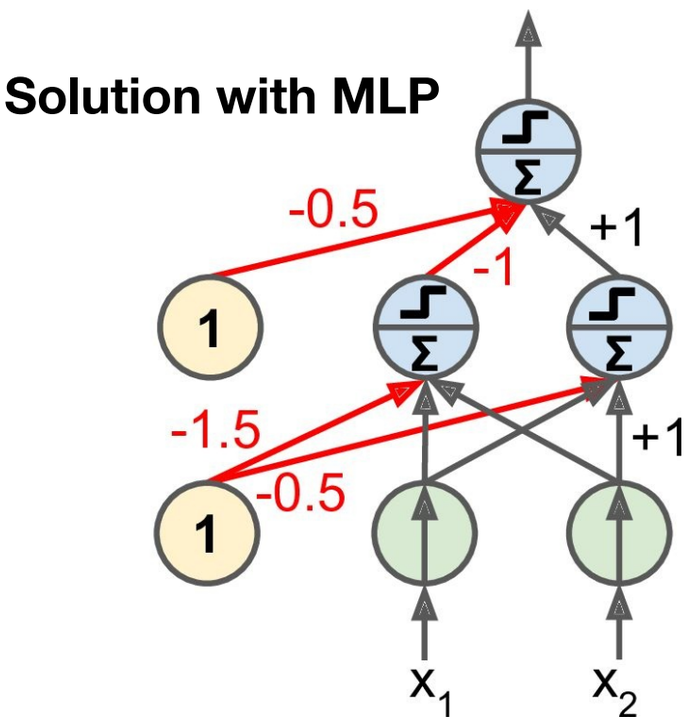
- Serious weaknesses of perceptrons: Incapable of solving some rather simple problems (*).
  - Example: XOR

$$h(x_1, x_2) = \begin{cases} 0 & (x_1 = x_2) \\ 1 & (x_1 \neq x_2) \end{cases}$$

for given $x_1, x_2 \in 0, 1$

- These limitations can be overcome by stacking multiple perceptrons, so called Multi-Layer Perceptrons (MLP).
  - MLP can solve XOR



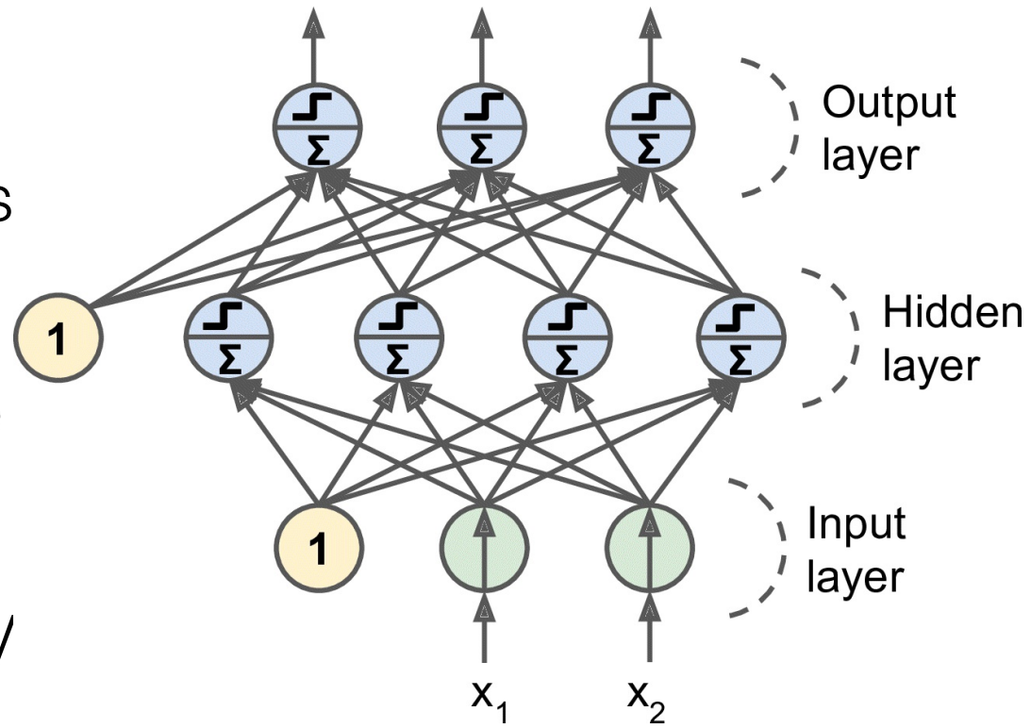


Figures from A. Géron, Hands-On Machine Learning with Scikit-Learn and Tensorflow

(*) See M. Minsky and S. Papert, *Perceptrons*, Cambridge, MA: MIT Press, 1969.

Multi-Layer Perceptron (MLP)

- An MLP is composed of
  - Input layer: Inputs passed through
  - Hidden layers: One or more layers of LTUs
  - Output layer: Final layer of LTUs
- Input layer and hidden layers include a bias neuron.
- Input layer and hidden layers are fully connected to the next layer.
- Activation Function: Generalisation of the *hard threshold* replaced by a smooth functions.



Does the Perceptron Learning Algorithm work here?

Outlook

- How to learn MLPs?
  - Cost Function
  - Learning as an Optimisation Problem
  - Optimisation Algorithm, Gradient Descent

- Illustrated with example based on MNIST dataset