# COMP9444: Neural Networks and Deep Learning

Week 1b. Neurons and Perceptrons

Raymond Louie
School of Computer Science and Engineering
Feb 18, 2025



## Biological motivation of neural networks

Neural networks are designed as simplified models of how the human brain processes information

1. **How does the brain process information?**
2. What is the function and structure of the brain?

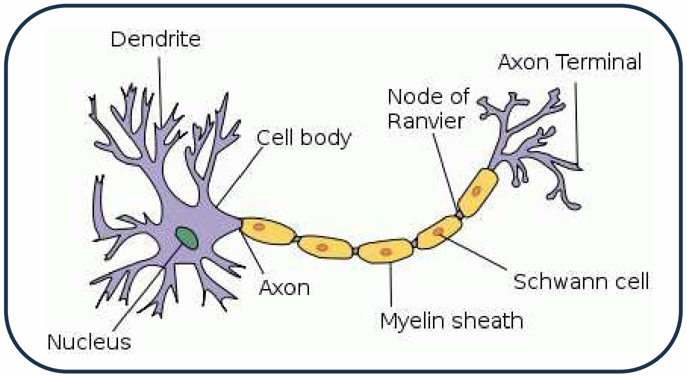


## Neurons as Body Cells

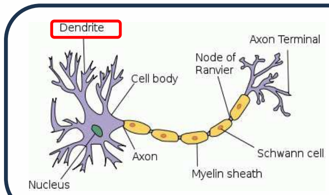
- The body is made up of billions of cells. Cells of the nervous system, called *neurons*, are specialized to carry "messages" through an electrochemical process.
- The human brain has about 100 billion neurons, and a similar number of support cells called "glia".
- Neurons are similar to other cells in the body in some ways, such as:
  - neurons are surrounded by a cell membrane
  - neurons have a nucleus that contains genes (DNA)
  - neurons carry out basic cellular processes like protein synthesis and energy production



## Structure of a Typical Neuron



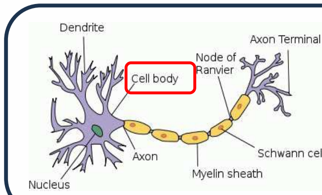
## Dendrite (input)




- Bring information to the cell body
- Number of dendrites
  - → Unipolar and Bipolar neurons have only one dendrite
  - → Purkinje neurons can have up to 100,000 dendrites
- Typically less than a millimetre in length



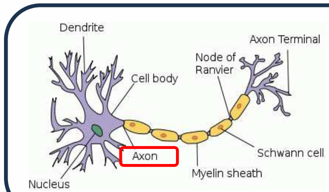
## Cell body




- Responsible for processing and transmitting information
- Contains genetic information, maintains neuron's structure and provides energy for activities



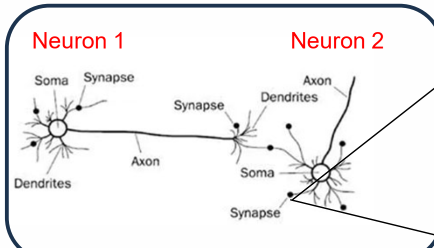
## Axon (output)

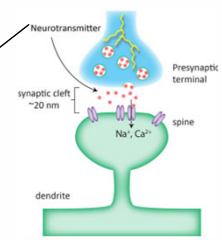



- T ake information away from the cell body.
- Most neurons have one output
- Axons can vary in length from less than a millimetre to more than a metre (motor neurons)
- Long axons are sometimes surrounded by a myelinated sheath, which prevents the electrical signal from dispersing, and allows it to travel faster (up to 100 m/s).



## Synapse connections for communication

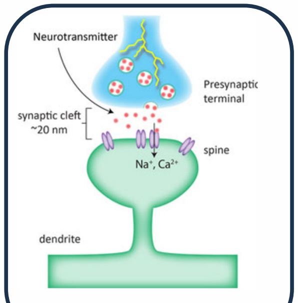



Synapse (zoom in)

- The axon of one neuron can connect to the dendrite of another neuron through an electrochemical junction called a *synapse*.

## How communication occurs?




Synapse

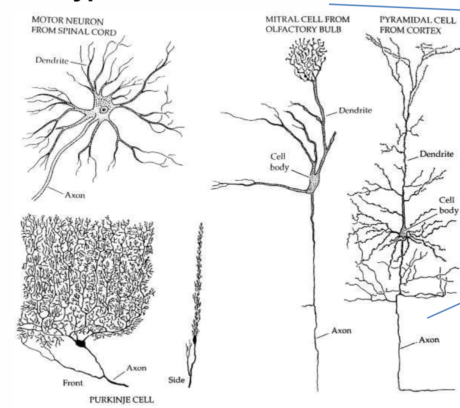
1. Electrical signal (action potential) travels down axon of the first neuron
2. Upon reaching axon terminal, it triggers the release of neurotransmitters into the synaptic cleft, which bind to receptors on the second neuron
3. This binding causes a change in voltage across the membrane to either move towards (excitation) or away (inhibition) from threshold
4. If threshold is crossed (-55 mV), an electrical signal is released in the second neuron



## Variety of Neuron Types

Large cell body, multiple dendrites, long axon

Extensive, tree-like dendritic structure and a relatively thick axon




branching dendritic structure, a cell body, and a long axon

triangular-shaped cell body with extensive branching dendrites and a long axon

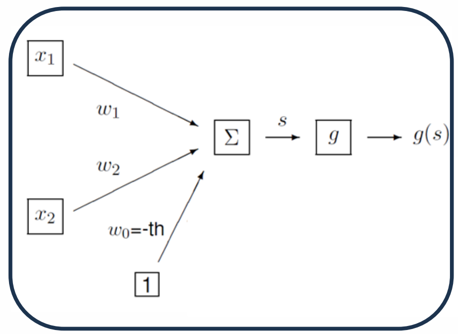


## The Big Picture

- human brain has 100 billion neurons and 100 trillion million synapses each
- latency is about 3-6 milliseconds
- therefore, at most a few hundred "steps" in any mental computation, but massively parallel



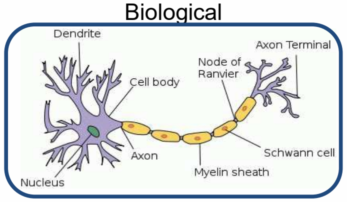
## McCulloch & Pitts Model of a Single Artificial Neuron (1943)



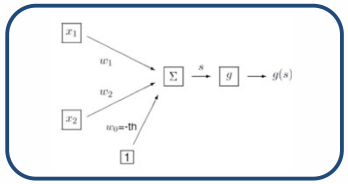

- **Inputs:** $x_1$ and $x_2$
- **Weights**
  - $w_1$, $w_2$: input weights
  - $w_0$: bias weight, th is threshold
- **Input function**
  - $s=w_1x_1+w_2x_2-w_o$
- **Transfer/activation function**: g

## Biological and artificial neuron comparison

Biological




Artificial




| | Biological | Artificial |
|---|---|---|
| **Inputs** | Dendrites | $x_1$ and $x_2$ |
| **Processing** | Cell body | $s$ |
| **Output** | Axon | $g(s)$ |
| **Connection** | Synapses (axon-dendrite) | $w_1$ and $w_2$ |



## Synapses and activation

**Biological neural network**

- Synapses can be *exitatory* or *inhibitory* and may change over time.
- When the inputs reach some threshhold an *action potential* (electrical pulse) is sent along the axon to the outputs.

**Artificial neural networks**

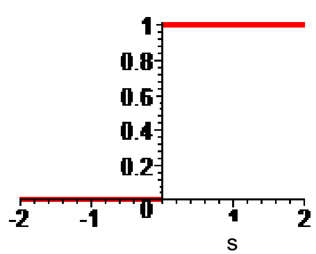
- Weights can be positive or negative and may change over time (learning).
- The *input function* is the weighted sum of the activation levels of inputs.
- The activation level is a non-linear *transfer* function $g$ of this input:

$$\boxed{\text{activation}_i = g(s_i) = g(\sum_j w_{ij} x_j)}$$



## Transfer function

- Originally, a (discontinuous) step function was used for the transfer function:




$$\boxed{g(s) = \begin{cases} 1, & \text{if} \quad s > 0 \\ 0, & \text{if} \quad s < 0 \end{cases}}$$

- Technically, this is called the **step** function if $g(0) = 1$ and the **Heaviside** function if $g(0) = 0.5$ (but, we will use the two terms interchangeably).
- (Later, other transfer functions were introduced, which are continuous and smooth)



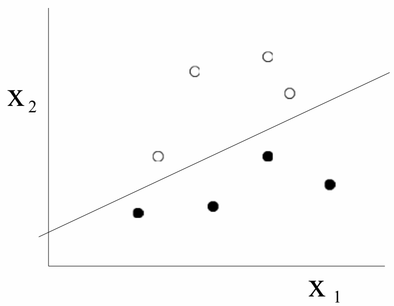
## Supervised learning example

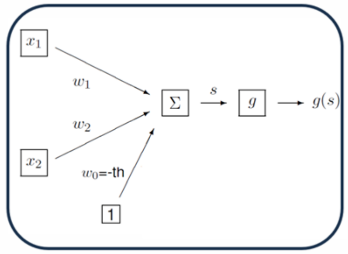
input → f() → output

- **Supervised learning:** Given (input,output), what is f()? (or what are the weights in neural network?)

## Example (supervised learning) problem

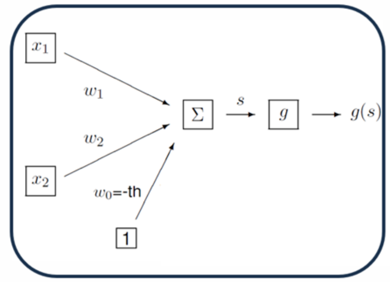
- **Assumption**: 2 inputs, ($x_1$,$x_2$ coordinates) and corresponding output (1=black, 0=white), 8 training samples
- **Problem statement:** Determine the weights to match the input/output






## Other assumptions




Begin with random weights

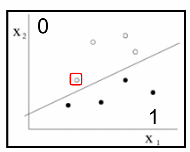
$w_1 = 0.2$
$w_2 = 0.0$
$w_0 = -0.1$

$s = 0.2\ x_1 + 0.0\ x_2 - 0.1$

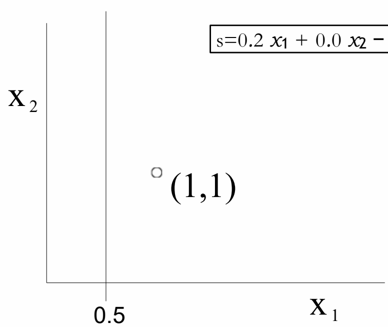
1. Are these weights consistent with the training sample?
2. If not, how do we adjust the weights to make it consistent?



## Are these weights consistent with training sample?




$s = 0.2\ x_1 + 0.0\ x_2 - 0.1$

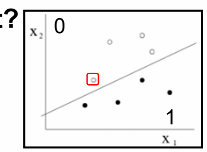



Remember

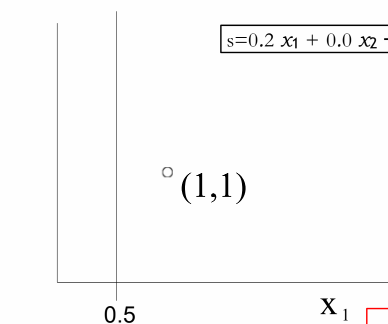
$$g(s) = \begin{cases} 1, & \text{if } s > 0 \\ 0, & \text{if } s < 0 \end{cases}$$



## How do we adjust the weights so it's consistent?




$s = 0.2\ x_1 + 0.0\ x_2 - 0.1$




Remember

$$g(s) = \begin{cases} 1, & \text{if } s > 0 \\ 0, & \text{if } s < 0 \end{cases}$$

- 1 if $s = 0.2\ x_1 + 0.0\ x_2 - 0.1 > 0$
- g(s) = 1 (incorrectly predict (1,1) is black dot)

- Should we increase or decrease weights?
- When do we increase?

## Perceptron Learning Rule

- Adjust the weights as each input is presented.
- Recall: $s = w_1x_1 + w_2x_2 + w_0$

if $g(s) = 1$ but should be $0$,

$$\boxed{\begin{aligned} w_k &\leftarrow w_k - \eta\, x_k \\ w_0 &\leftarrow w_0 - \eta \\ \text{so}\quad s &\leftarrow s - \eta\,(1 + \sum_k x_k^2) \end{aligned}}$$

if $g(s) = 0$ but should be $1$,

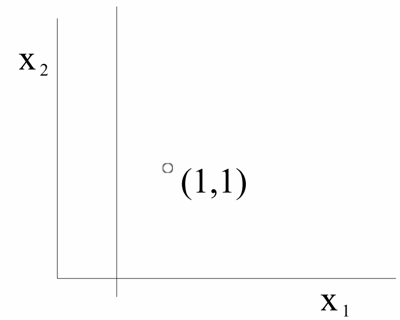
$$\boxed{\begin{aligned} w_k &\leftarrow w_k + \eta\, x_k \\ w_0 &\leftarrow w_0 + \eta \\ \text{so}\quad s &\leftarrow s + \eta\,(1 + \sum_k x_k^2) \end{aligned}}$$

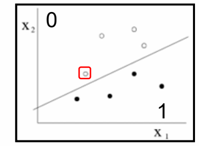
otherwise, weights are unchanged. ($\eta > 0$ is called the **learning rate**)

- If input is large and there's an error, adjusted weight needs to be larger to fix the error



## Training Step 1




$0.2\ x_1 + 0.0\ x_2 - 0.1 > 0$

$$\boxed{\begin{aligned} w_1 &\leftarrow w_1 - \eta\, x_1 &= 0.1 \\ w_2 &\leftarrow w_2 - \eta\, x_2 &= -0.1 \\ w_0 &\leftarrow w_0 - \eta &= -0.2 \end{aligned}}$$

$0.1\ x_1\ .\ 0.1\ x_2 - 0.2 < 0$

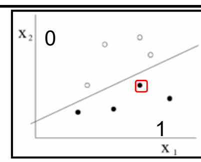
This point is now correctly classified

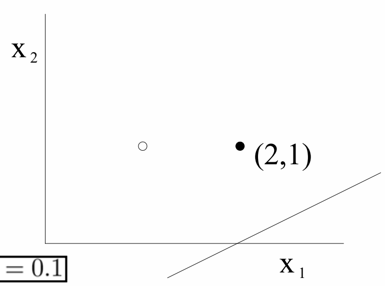
$\boxed{\eta = 0.1}$



## Training Step 2 – correct? Increase/decrease weights?

$\boxed{s=0.1\ x_1\ .\ 0.1\ x_2 - 0.2}$



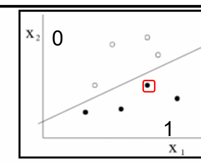



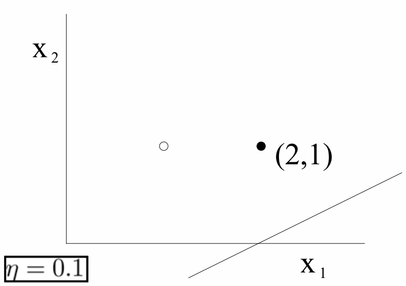

$\boxed{\eta = 0.1}$



## Training Step 2

$\boxed{s=0.1\ x_1\ .\ 0.1\ x_2 - 0.2}$




- s=0.1 *2 -0.1 *1 − 0.2 < 0
- g(s) = 0 (incorrectly predict (2,1) is white dot)

$$\boxed{\begin{aligned} w_1 &\leftarrow w_1 + \eta\, x_1 &= 0.3 \\ w_2 &\leftarrow w_2 + \eta\, x_2 &= 0.0 \\ w_0 &\leftarrow w_0 + \eta &= -0.1 \end{aligned}}$$

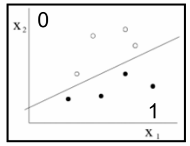
$0.3\ x_1 + 0.0\ x_2 - 0.1 > 0$

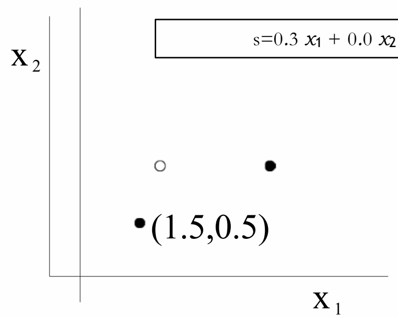
This point is now correctly classified. But?

$\boxed{\eta = 0.1}$

## Training Step 3 – correct?

$s=0.3\,x_1 + 0.0\,x_2 - 0.1$



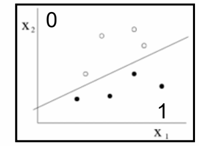



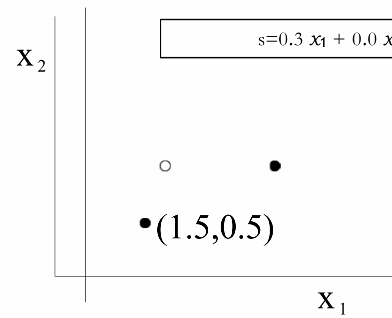

$\eta = 0.1$



## Training Step 3

$s=0.3\,x_1 + 0.0\,x_2 - 0.1$



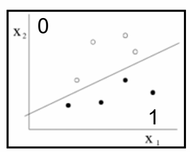

- s=0.3 *1.5 +0.0 *0.5 − 0.1 > 0
- g(s) = 1 (correctly predict (1.5,0.5) is black dot)
- 3rd point correctly classified, so no change

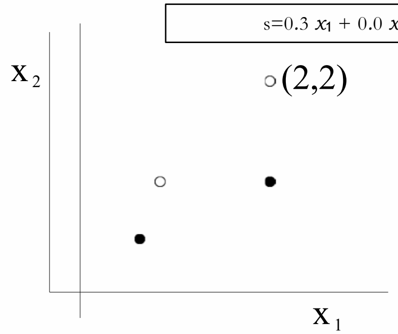
$\eta = 0.1$



## Training Step 4

$s=0.3\,x_1 + 0.0\,x_2 - 0.1$




- s=0.3 *2 +0.0 *2 − 0.1 > 0
- g(s) = 1 (incorrectly predict (2,2) is black)

$$
\begin{aligned}
w_1 &\leftarrow w_1 - \eta\, x_1 &= 0.1\\
w_2 &\leftarrow w_2 - \eta\, x_2 &= -0.2\\
w_0 &\leftarrow w_0 - \eta &= -0.2
\end{aligned}
$$

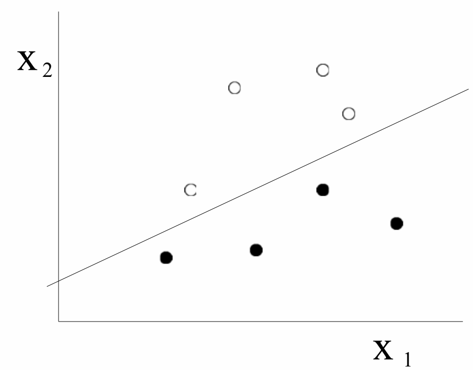
$0.1\,x_1 - 0.2\,x_2 - 0.2 > 0$

This point is now correctly classified

$\eta = 0.1$

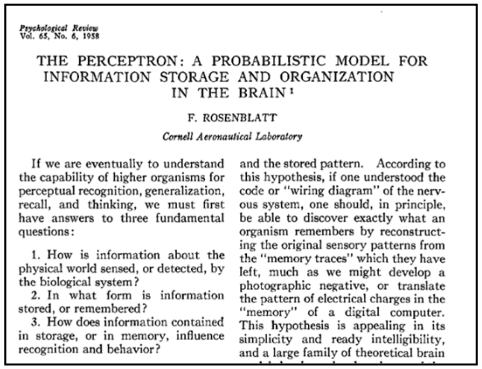


## Final Outcome




- eventually, all the data will be correctly classified

## Rosenblatt 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*

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

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?

and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much as we might develop a photographic negative, or translate the pattern of electrical charges in the "memory" of a digital computer. This hypothesis is appealing in its simplicity and ready intelligibility, and a large family of theoretical brain



## Rosenblatt Perceptron


- An IBM 704 – a 5-ton computer the size of a room – was fed a series of punch cards.
- After 50 trials, the computer taught itself to distinguish cards marked on the left from cards marked on the right.



## Rosenblatt Perceptron

THE MARK I PERCEPTRON



## What can the perceptron learn?

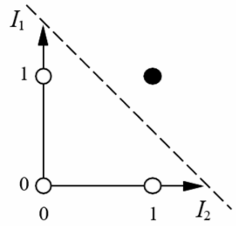
$$\begin{aligned} s &= w_1x_1 + w_2x_2 - \text{th} \\ &= w_1x_1 + w_2x_2 + w_0 \end{aligned}$$

- **Theorem:** The perceptron algorithm will eventually learn to classify the data correctly, as long as they are **linearly separable**.

## AND

$$\boxed{\begin{aligned} s &= w_1 x_1 + w_2 x_2 - \text{th} \\ &= w_1 x_1 + w_2 x_2 + w_0 \end{aligned}}$$

| I1 | I2 | AND out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




**(a)** $I_1$ **and** $I_2$

$w_1 = w_2 = 1.0, \quad w_0 = -1.5$

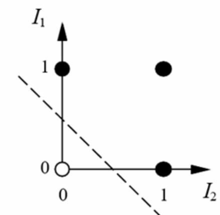
$s = x_1 + x_2 - 1.5$



## OR

$$\boxed{\begin{aligned} s &= w_1 x_1 + w_2 x_2 - \text{th} \\ &= w_1 x_1 + w_2 x_2 + w_0 \end{aligned}}$$

| I1 | I2 | OR out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




**(b)** $I_1$ **or** $I_2$

$w_1 = w_2 = 1.0, \quad w_0 = -0.5$

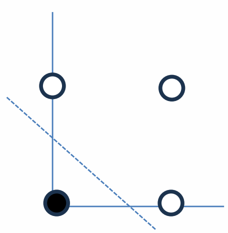
$s = x_1 + x_2 - 0.5$



## NOR

$$\boxed{\begin{aligned} s &= w_1 x_1 + w_2 x_2 - \text{th} \\ &= w_1 x_1 + w_2 x_2 + w_0 \end{aligned}}$$

| I1 | I2 | NOR out |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



$w_1 = w_2 = -1.0, \quad w_0 = 0.5$

$s = -x_1 - x_2 + 0.5$



## What can the perceptron learn?

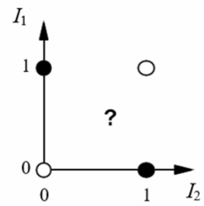
$$\boxed{\begin{aligned} s &= w_1 x_1 + w_2 x_2 - \text{th} \\ &= w_1 x_1 + w_2 x_2 + w_0 \end{aligned}}$$

Linearly separable functions. E.g.,

| | | | |
|---|---|---|---|
| AND | $w_1 = w_2 = 1.0,$ | $w_0 = -1.5$ | $s = x_1 + x_2 - 1.5$ |
| OR | $w_1 = w_2 = 1.0,$ | $w_0 = -0.5$ | $s = x_1 + x_2 - 0.5$ |
| NOR | $w_1 = w_2 = -1.0,$ | $w_0 = 0.5$ | $s = -x_1 - x_2 + 0.5$ |

## XOR

| I1 | I2 | OR out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

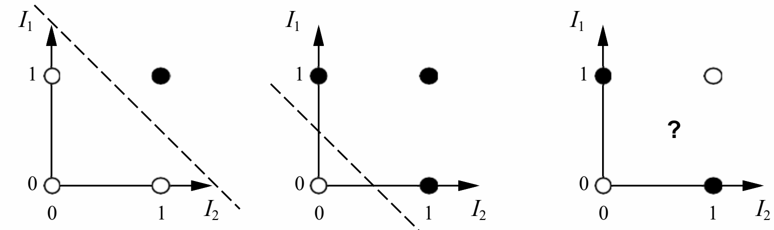



(c) $I_1$ xor $I_2$



## Limitations of (single-layer) Perceptrons

Problem: many useful functions are not linearly separable (e.g. XOR)



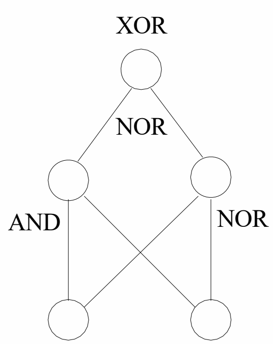

(a) $I_1$ and $I_2$ (b) $I_1$ or $I_2$ (c) $I_1$ xor $I_2$

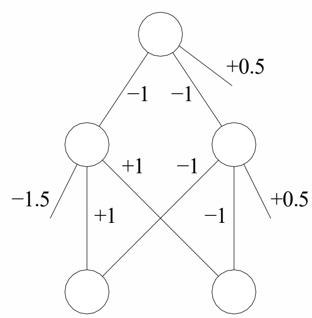
Possible solution:

- $x_1$ XOR $x_2$ can be written as: ($x_1$ AND $x_2$) NOR ($x_1$ NOR $x_2$)
- Recall that AND, OR and NOR can be implemented by perceptrons.



## Multi-Layer Neural Networks




Problem: How can we train it to learn a new function?



## Historical Context

- In 1969, Minsky and Papert published a book highlighting the limitations of Perceptrons, and lobbied various funding agencies to redirect funding away from neural network research, preferring instead logic-based methods such as expert systems.
- It was known as far back as the 1960's that any given logical function could be implemented in a 2-layer neural network with step function activations. But, the the question of how to learn the weights of a multi-layer neural network based on training examples remained an open problem. The solution, which we describe in the next section, was found in 1976 by Paul Werbos, but did not become widely known until it was rediscovered in 1986 by Rumelhart, Hinton and Williams.