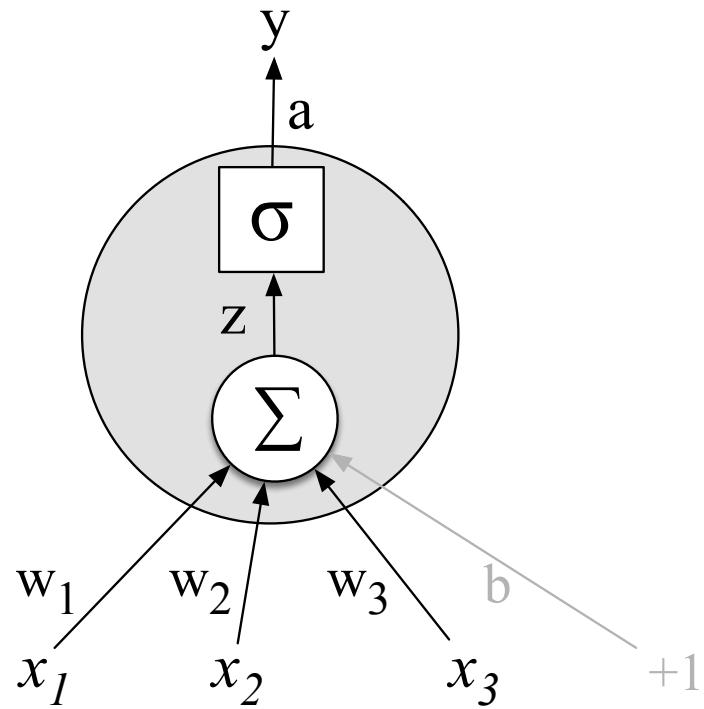


# Neural Networks

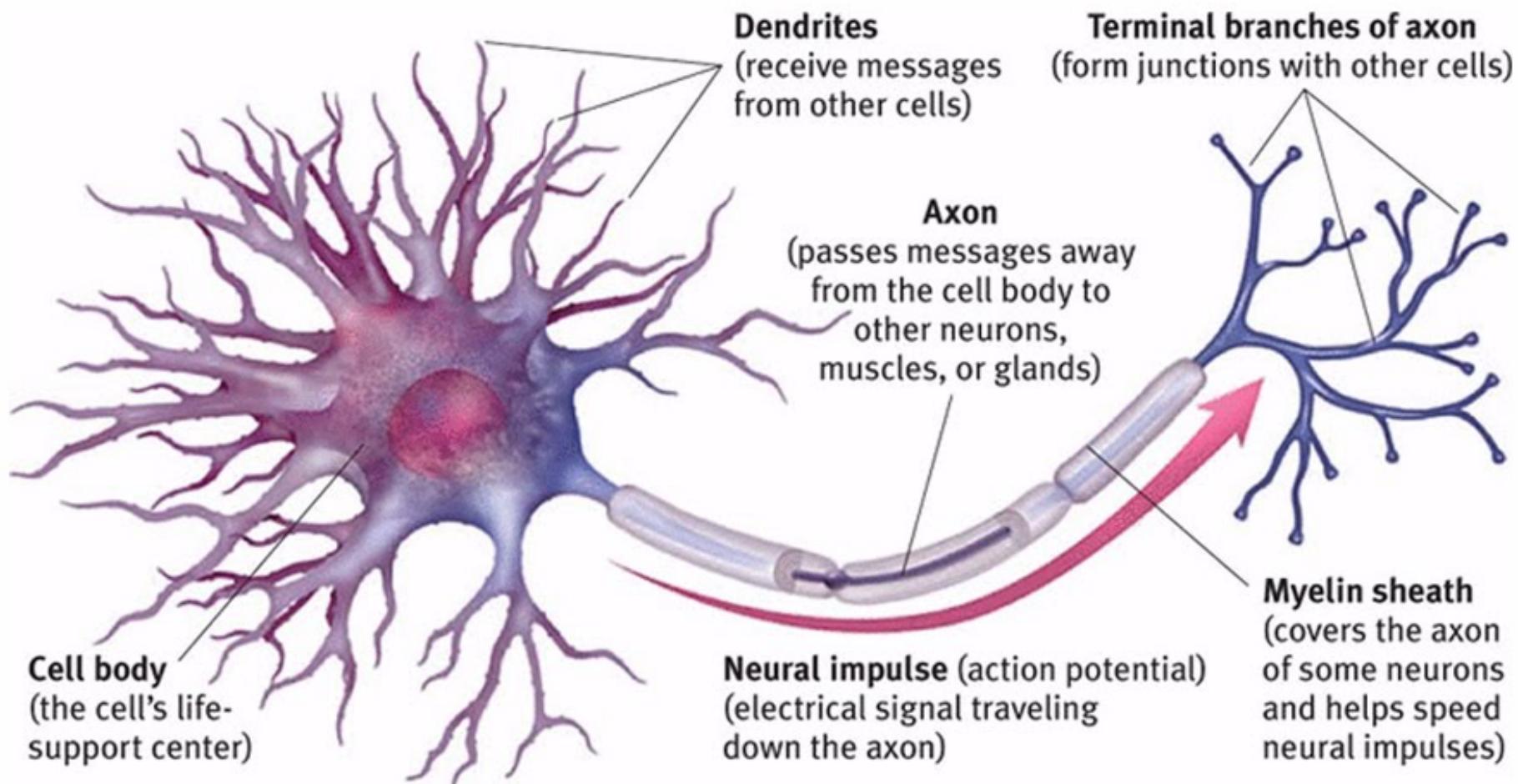
JURAFSKY AND MARTIN CHAPTER 7

# Neural Networks

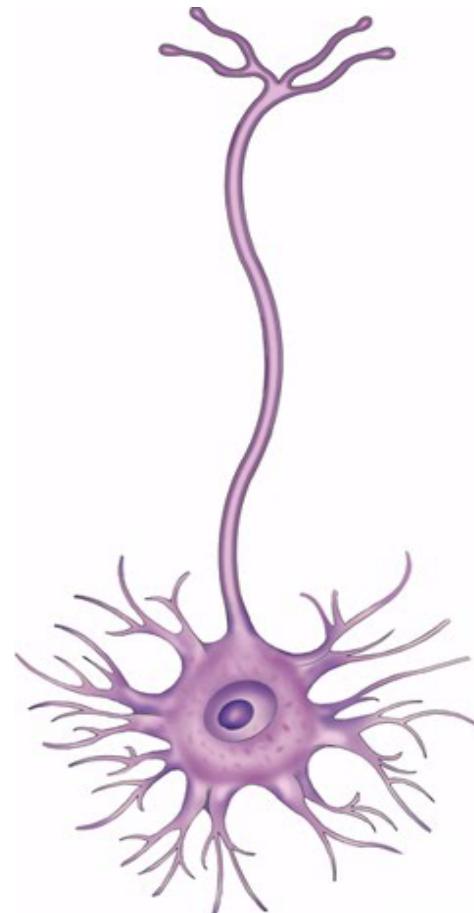
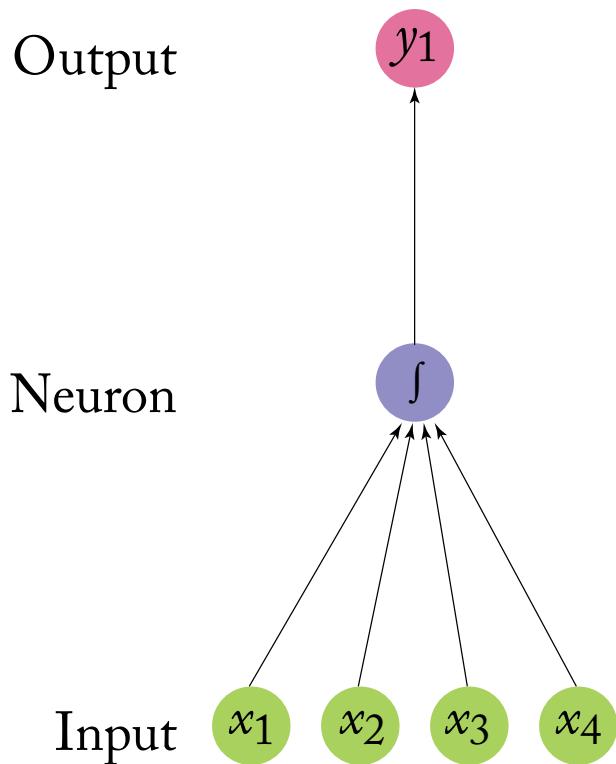
The building block of a neural network is a single computational unit. A unit takes a set of real valued numbers as input, performs some computation.



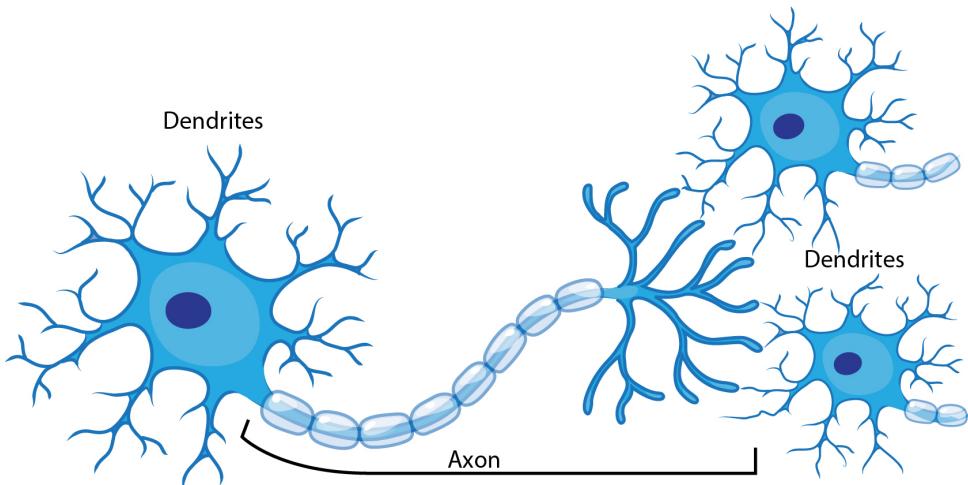
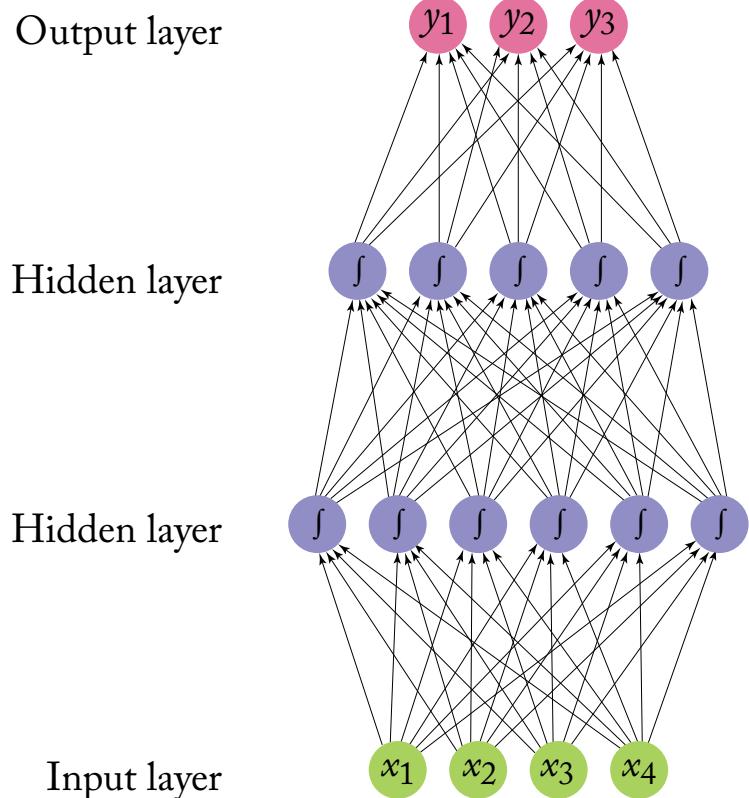
# Neural Networks: A brain-inspired metaphor



# A single neuron

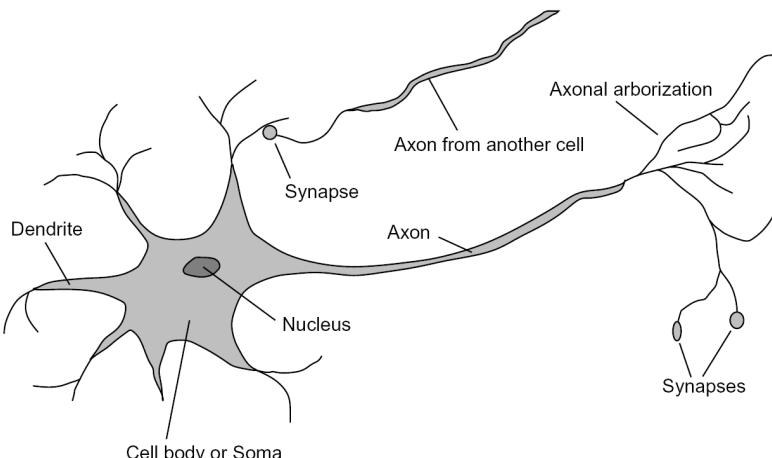
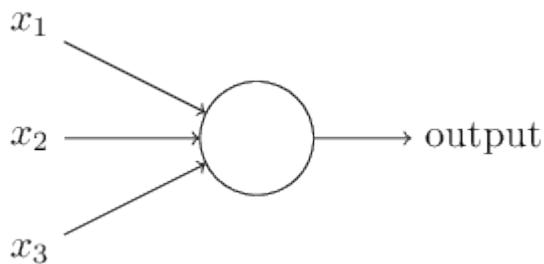


# Neural networks



# Review: Perceptron

Perceptrons were developed in the 1950s and 1960s loosely inspired by the neuron.



## Electronic 'Brain' Teaches Itself

The Navy last week demonstrated the embryo of an electronic computer named the Perceptron which, when completed in about a year, is expected to be the first non-living mechanism able to "perceive, recognize and identify its surroundings without human training or control." Navy officers demonstrating a preliminary form of the device in Washington said they hesitated to call it a machine because it is so much like a "human being without life."

Dr. Frank Rosenblatt, research psychologist at the Cornell Aeronautical Laboratory, Inc., Buffalo, N. Y., designer of the Perceptron, conducted the demonstration. The machine, he said, would be the first electronic device to think as the human brain. Like humans, Perceptron will make mistakes at first, "but it will grow wiser as it gains experience," he said.

The first Perceptron, to cost about \$100,000, will have about 1,000 electronic "association cells" receiving electrical impulses from an eyelike scanning device with 400 photocells. The human brain has ten billion

recognize the difference between right and left, almost the way a child learns.

When fully developed, the Perceptron will be designed to remember images and information it has perceived itself, whereas ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons, Dr. Rosenblatt said, will be able to recognize people and call out their names. Printed pages, longhand letters and even speech commands are within its reach. Only one more step of development, a difficult step, he said, is needed for the device to hear speech in one language and instantly translate it to speech or writing in another language.

### Self-Reproduction

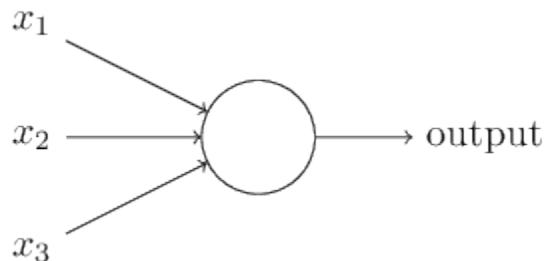
In principle, Dr. Rosenblatt said, it would be possible to build Perceptrons that could reproduce themselves on an assembly line and which would be "conscious" of their existence.

Perceptron, it was pointed out, needs no "priming." It is not necessary to introduce it to surround-

# Review: Perceptron

Perceptron has inputs,  $x_1, x_2, \dots, x_N$ , and weights  $w_1, w_2, \dots, w_N$

The perceptron outputs 0 or 1, based on the weighted sum is less than or greater than a threshold value

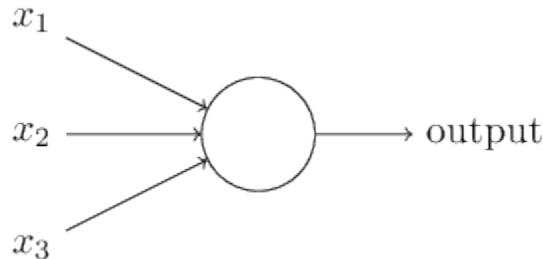


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

# Perceptrons for decision making

We can think about the perceptron or the sigmoid neuron as a device that makes decisions by weighing up evidence.

Example: Suppose there's a cheese festival in your town. You like cheese.



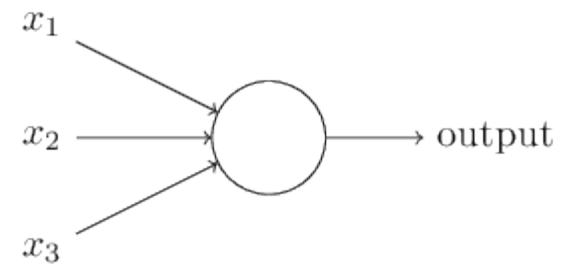
Example from Michael Nielsen's book [Neural Networks and Deep Learning](#)

# Perceptrons for decision making

You might use 3 factors to decide whether to go.

1. Is the weather good?
2. Can your loyal companion come with you?
3. Is the festival near public transit?

These can be the binary input values to a perceptron

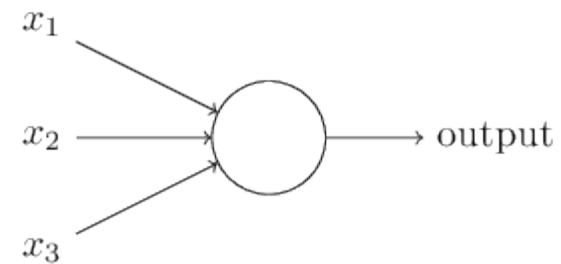
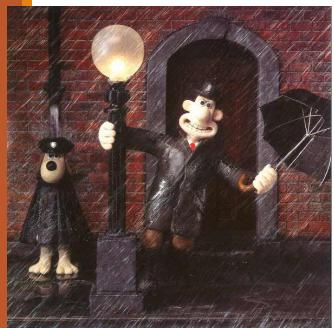


# Perceptrons for decision making

By varying weights and the threshold we get different models of decision making

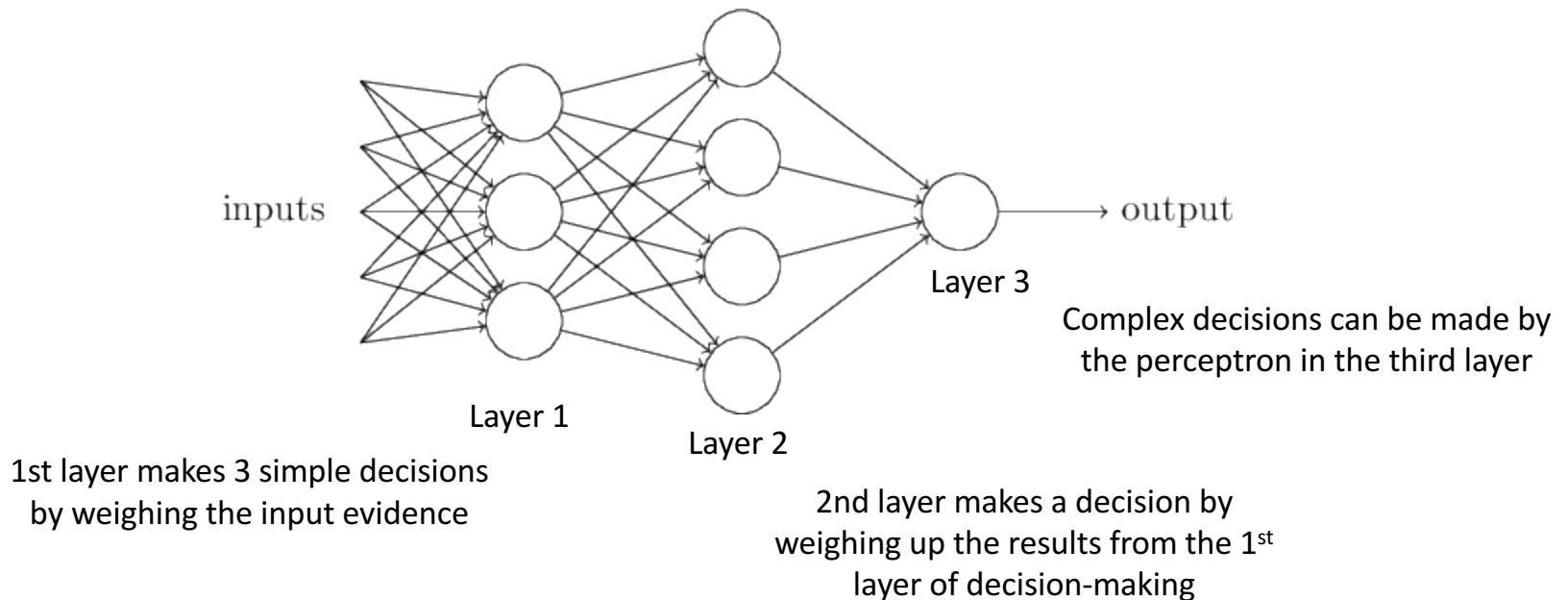
Example 1:  $w_1 = 6 \quad w_2 = 2 \quad w_3 = 2$ , threshold = 5

Example 2:  $w_1 = 6 \quad w_2 = 2 \quad w_3 = 2$ , threshold = 3



# Perceptrons for decision making

- A complex network of perceptrons could make quite subtle decisions:



# Weights, bias and dot products

- Two notational changes simplify the way that perceptrons are described.
- The first change is to replace the weighted sum as a dot product

$$w \cdot x \equiv \sum_j w_j x_j$$

- The second change is to move the threshold to the other side of the inequality, and to replace it by a *bias*,  $b$ -threshold

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

# Activation Functions

Instead of directly outputting  $z = w \cdot x + b$ , which is a linear function of  $x$ , neuron units apply a non-linear function  $f$  to  $z$ .

The output of this function is called the **activation value** for the unit, represented by the variable  $a$ . The output of a neural network is called  $y$ , so if the activation of a node is the final output of a network then

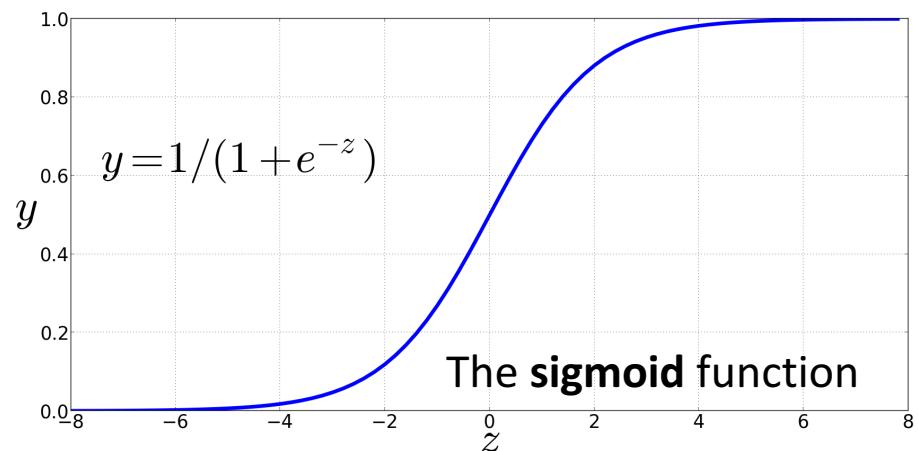
$$y=a=f(z)$$

There are 3 commonly used non-linear functions used for  $f$ :

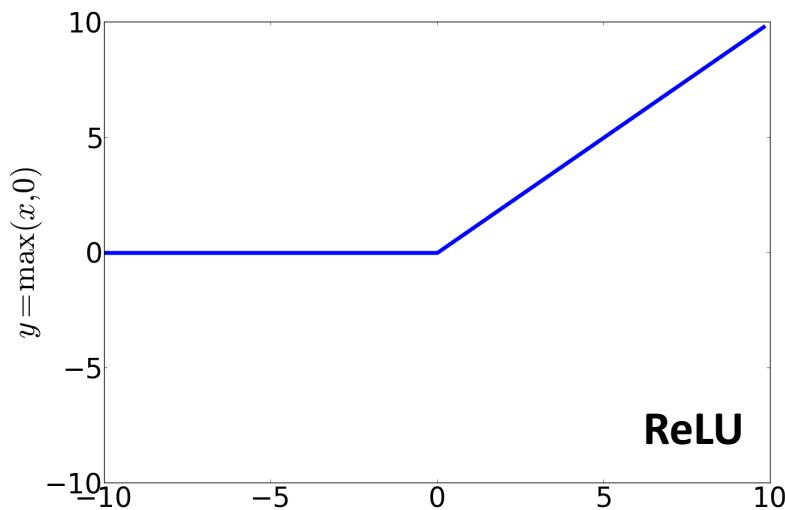
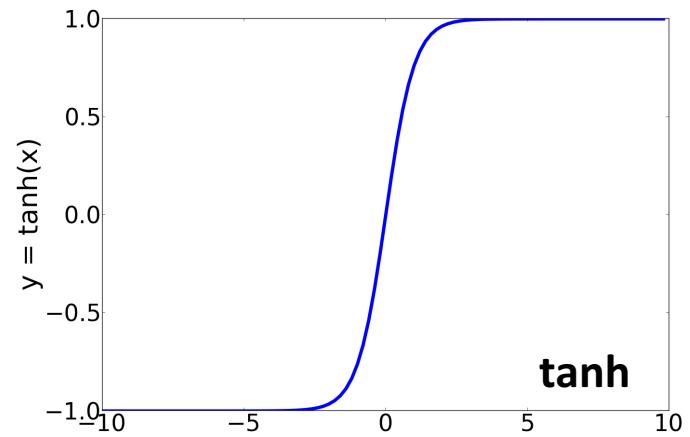
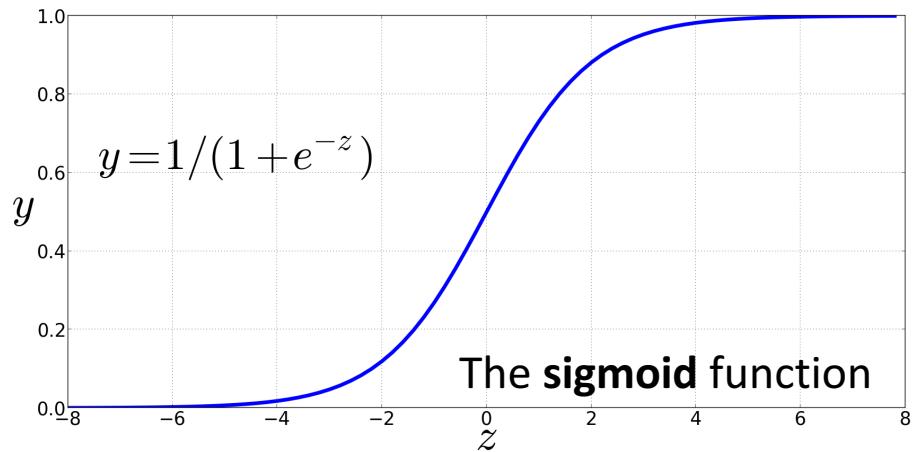
The **sigmoid** function

The **tanh** function

The **rectified linear unit ReLU**



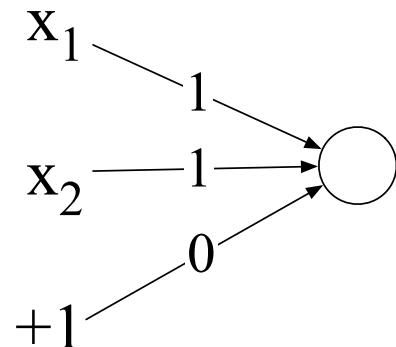
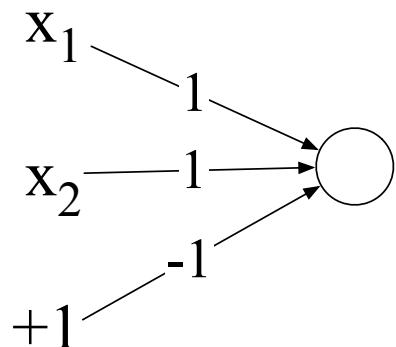
# Activation Functions



# Decision making OR logical functions

Perceptrons can be used to compute logical functions like AND, OR and NAND

Example:

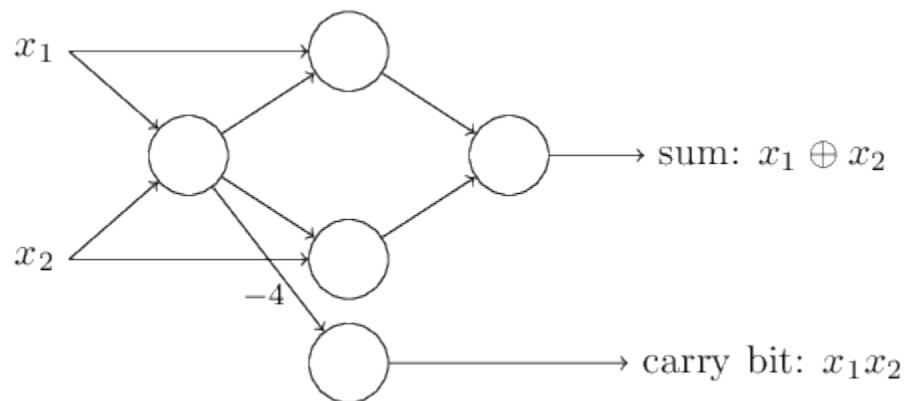
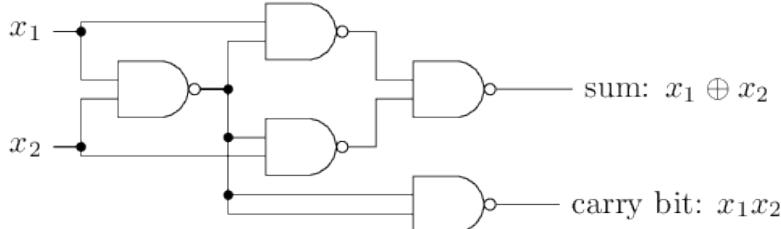


# Logical functions

Networks of perceptrons to compute *any* logical function

We can build any computation up out of NAND gates.

For example, a circuit which adds two bits  $x_1$  and  $x_2$



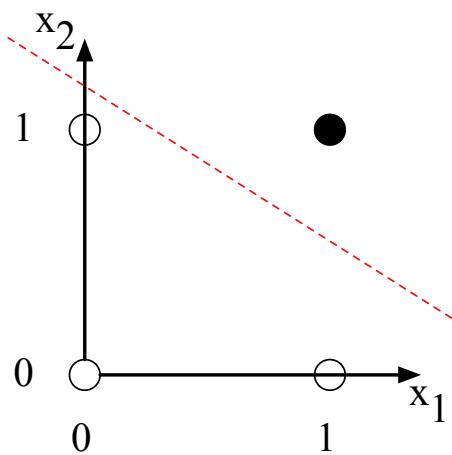
All unlabeled weights are -2, all biases =3.

# The XOR problem

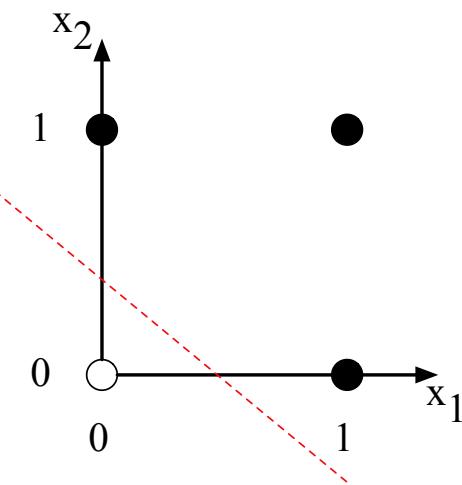
A single neural unit cannot be used to compute the XOR function

AND		OR		XOR	
x1	x2	y	x1	x2	y
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	0

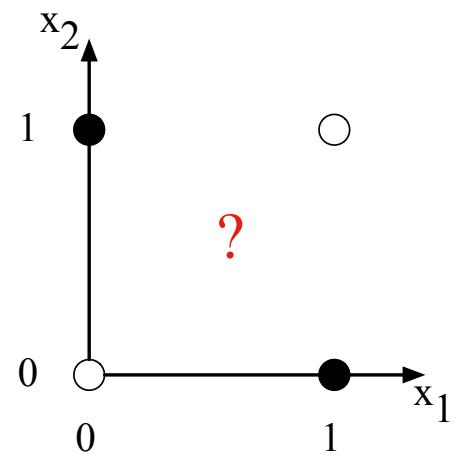
# The XOR Problem



a)  $x_1$  AND  $x_2$

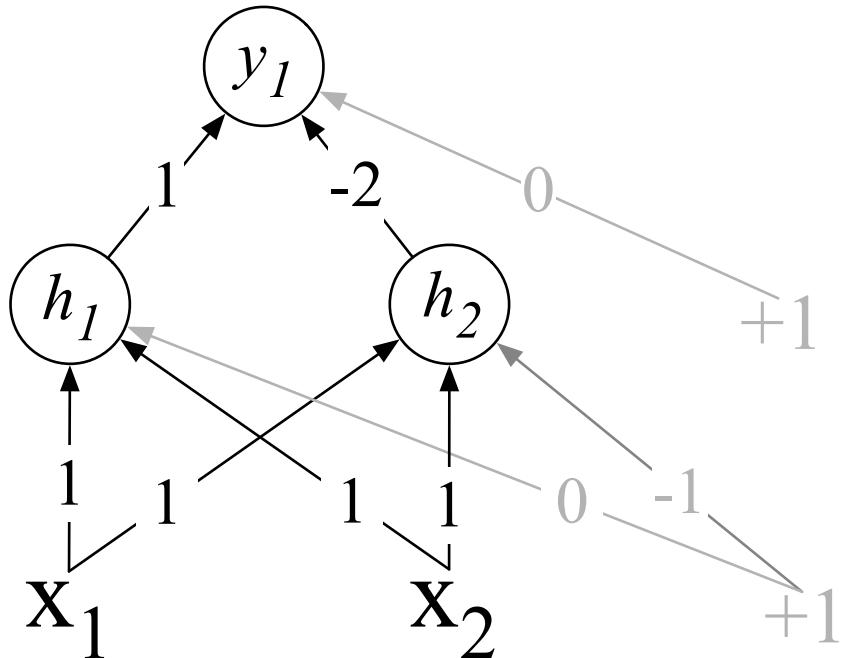


b)  $x_1$  OR  $x_2$

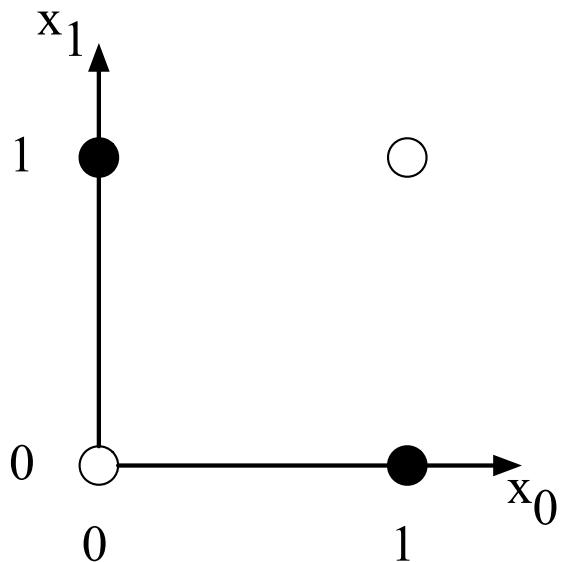


c)  $x_1$  XOR  $x_2$

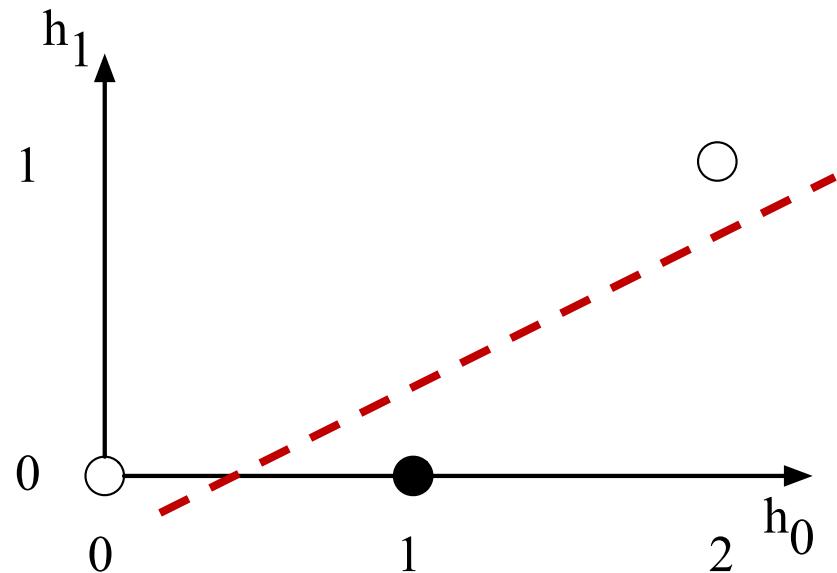
# The XOR Solution



# The XOR Solution



a) The original  $x$  space



b) The new  $h$  space

# Power of Perceptrons

Networks of Perceptrons are universal for computation, like NAND gates  
Perceptrons can be as powerful as any other computing device!

We can devise *learning algorithms* to automatically tune the weights and biases of a network of artificial neurons

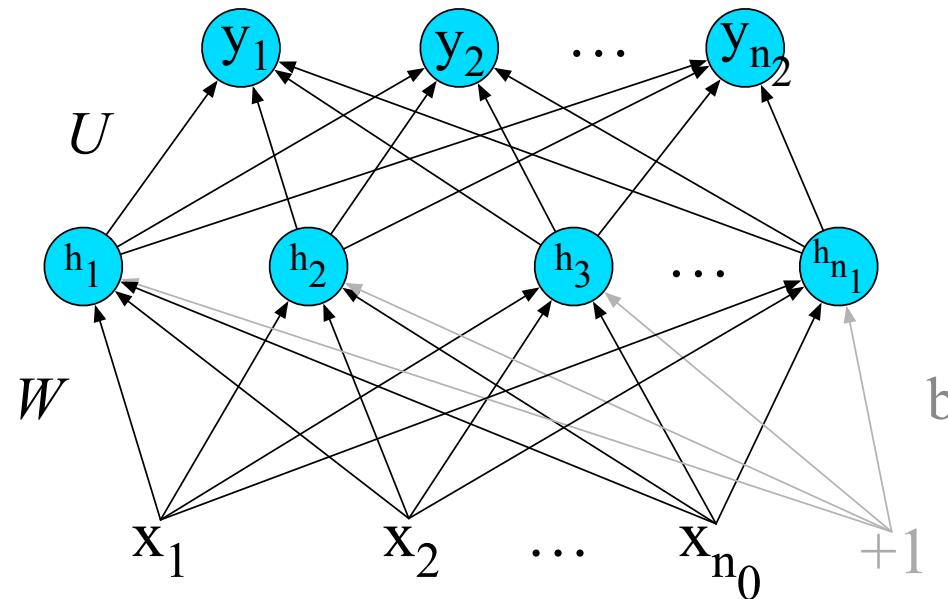
Instead of laying out a circuit of NAND and other gates, neural networks can simply learn to solve problems

# Feed-Forward Neural Network

The simplest kind of is the **Feed-Forward Neural Network**

**Multilayer** network, all units are usually **fully-connected**, and **no cycles**.

The outputs from each layer are passed to units in the next higher layer, and no outputs are passed back to lower layers.



# Learning to classify digits

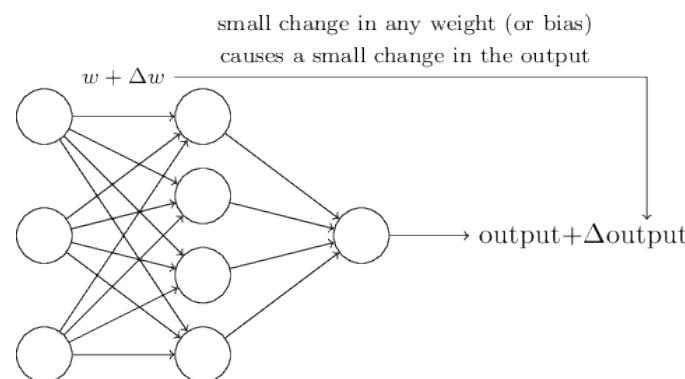
Input: image encoded input as a vector of intensities:

$$1 = \langle 0.0 \ 0.0 \ 0.3 \ 0.8 \ 0.7 \ 0.1 \dots 0.0 \rangle$$

Output: is this a one or not?

Goal: set the parameters of the network to correctly classify the digits

Learning = changing the weights and biases in the network.



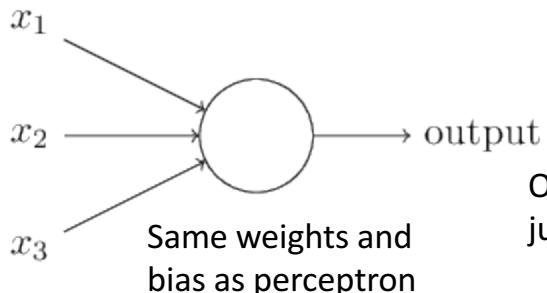
0	0
1	1
0	0
0	0
1	1
0	0

# Sigmoid neurons

Problem: a small change in the weights or bias of any single perceptron in the network can causes the output to completely flip from 0 to 1.

Solution: sigmoid neuron

$$\text{Perceptron output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$



Inputs: any real-valued number

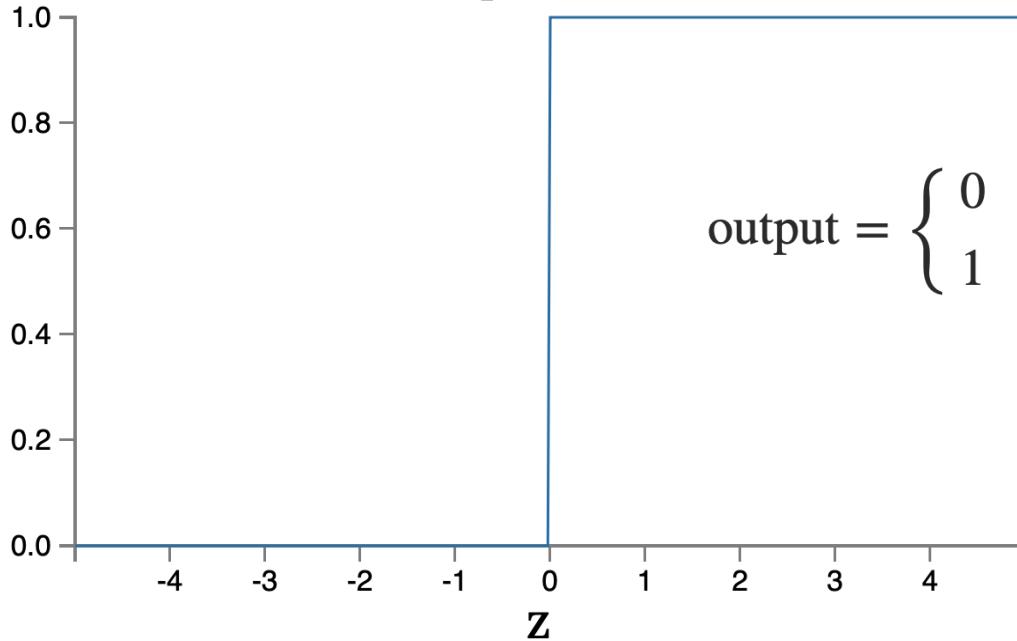
Output is no longer just 1 or 0.

$$\text{Sigmoid neuron output} = \sigma(w \cdot x + b)$$

$$\text{Sigmoid function: } \sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

# Perceptron

$$z \equiv w \cdot x + b$$



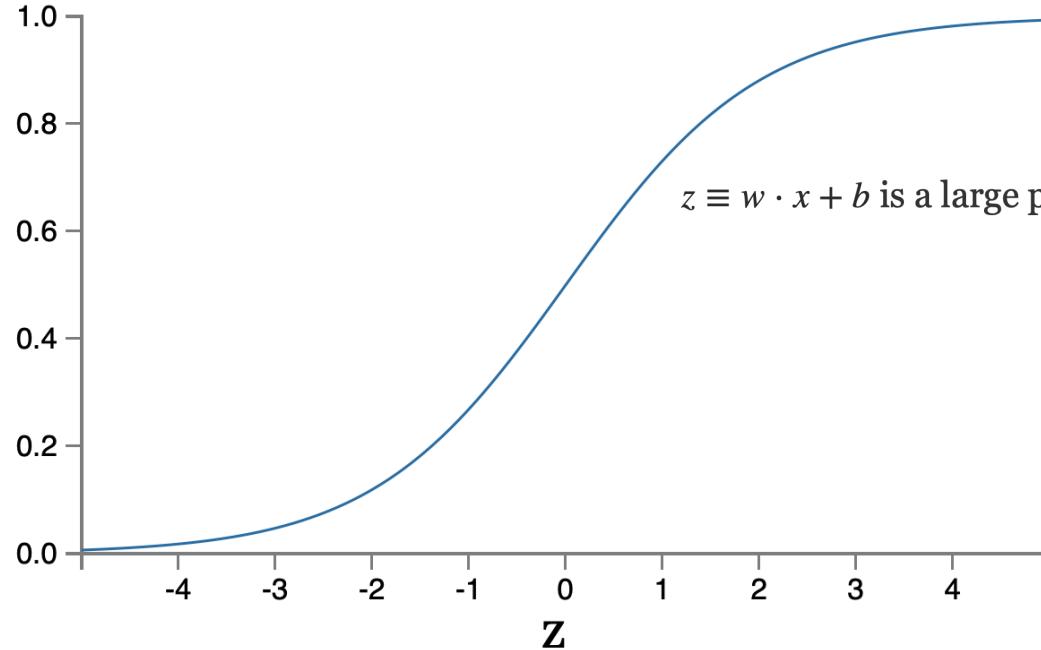
step function

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

# Sigmoid neuron

$$z \equiv w \cdot x + b$$

sigmoid function



$z \equiv w \cdot x + b$  is a large positive number. Then  $e^{-z} \approx 0$

$z = w \cdot x + b$  is very negative. Then  $e^{-z} \rightarrow \infty$ , and  $\sigma(z) \approx 0$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

# Smoothness is crucial

Smoothness of  $\sigma$  means that small changes in the weights  $w_j$  and in the bias  $b$  will produce a small change the output from the neuron

$$\Delta \text{output} \approx \sum_j \frac{\partial \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \text{output}}{\partial b} \Delta b$$

$\Delta \text{output}$  is a *linear junction* of the changes  $\Delta w_j$  and  $\Delta b$

This makes it easy to choose small changes in the weights and biases to achieve any desired small change in the output