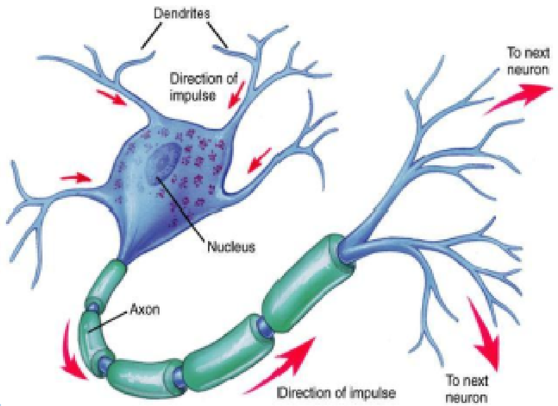
Introduction to Neural Nets

Kailas Maneparambil

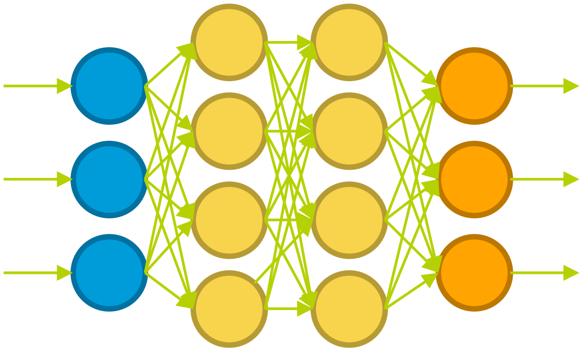
Biological Neural Nets

- Human brain ~86,000,000,000 neurons
- Each neuron connected to ~1000 others
- Electrochemical **inputs**
- **Only fire if** signal exceeds voltage threshold
- Signals are **spikes**
- All-or-nothing response



Motivation for Neural Nets

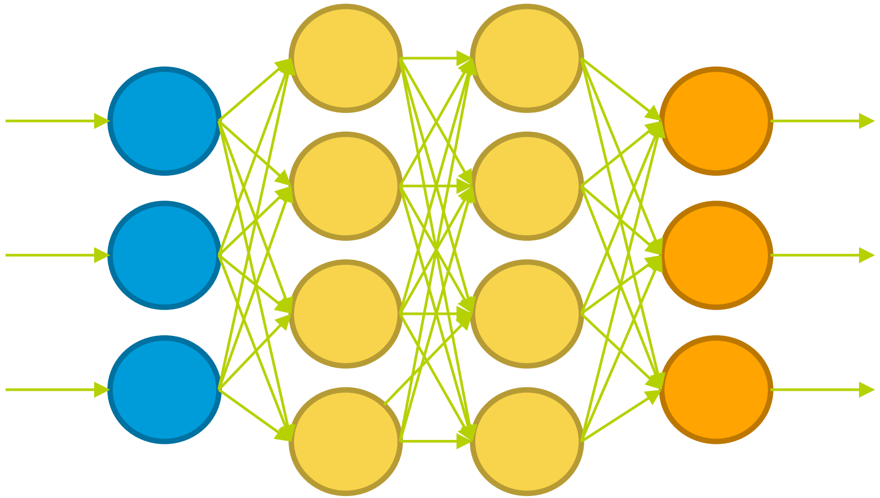
- Use biology as inspiration for mathematical model
- Get signals from previous neurons
- Generate signals (or not) according to inputs
- Pass signals on to next neurons
- By layering many neurons, can create complex model



Neural Net Structure

Input

(Feature Vector)

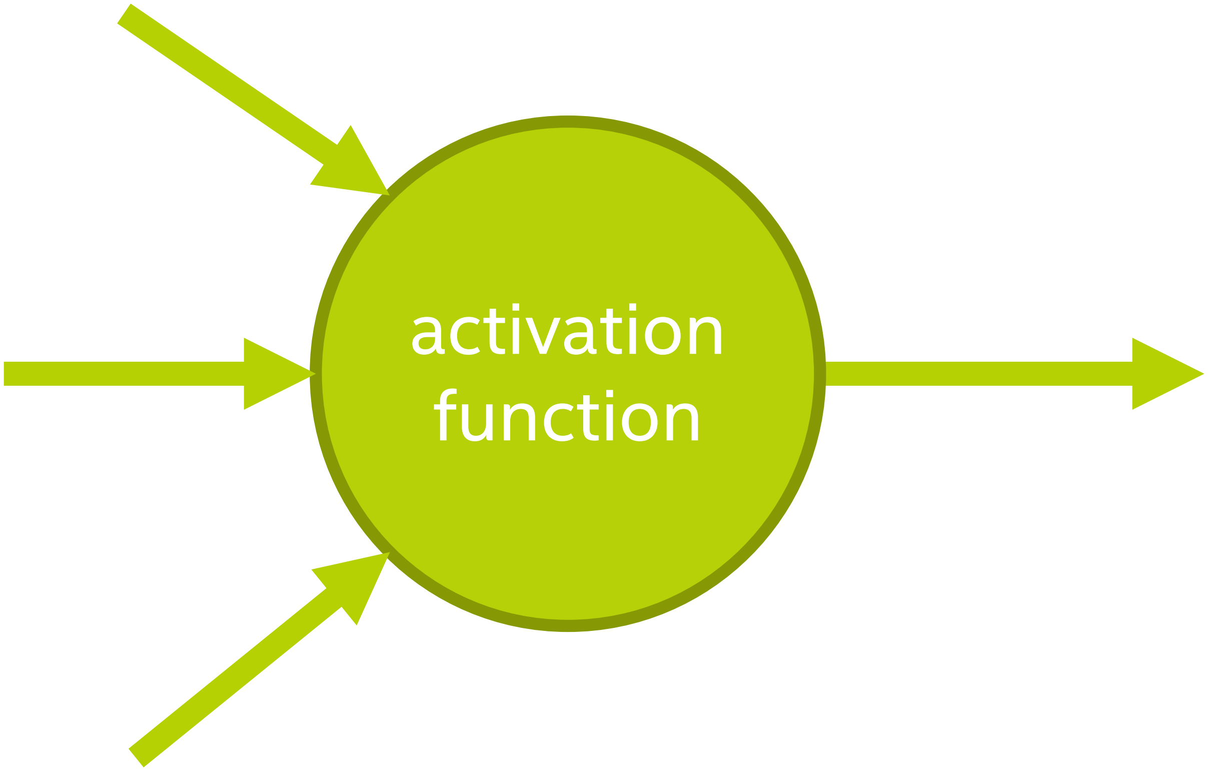


Output

(Label)

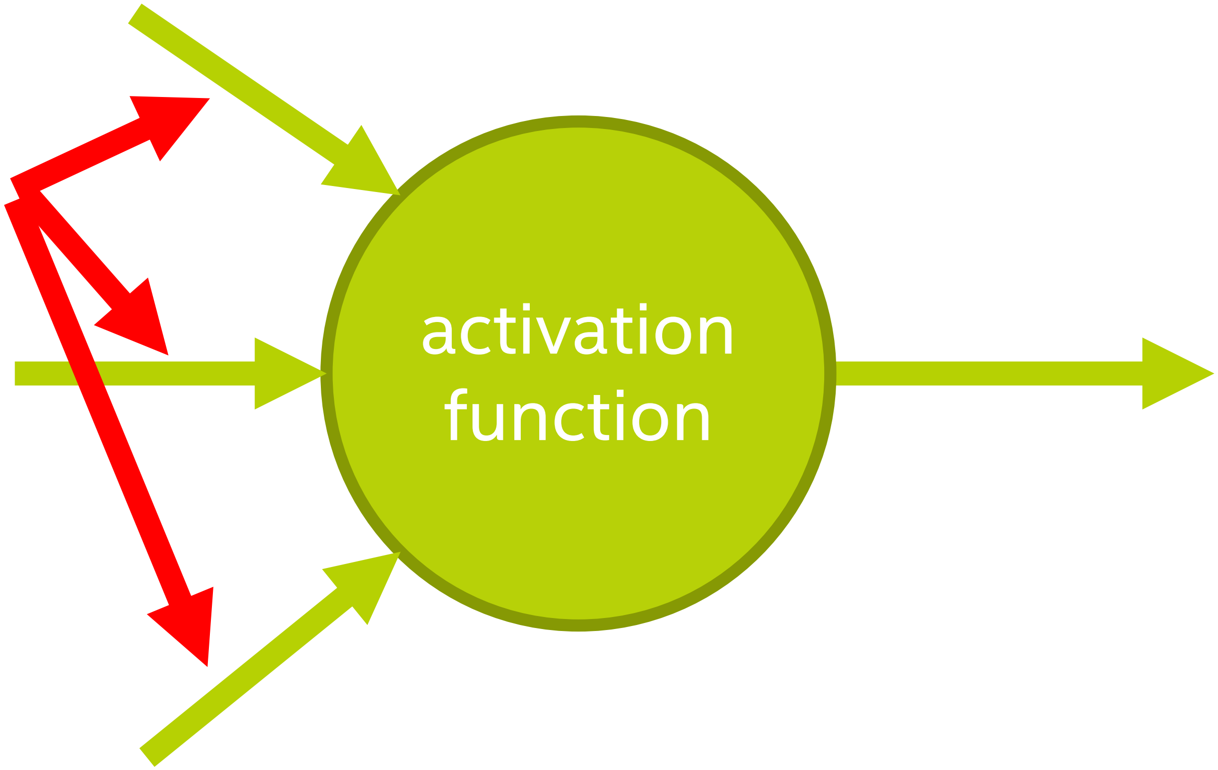
- Can think of it as a complicated computation engine
- We will ”train it” using our training data
- Then (hopefully) it will give good answers on new data

Basic Neuron Visualization

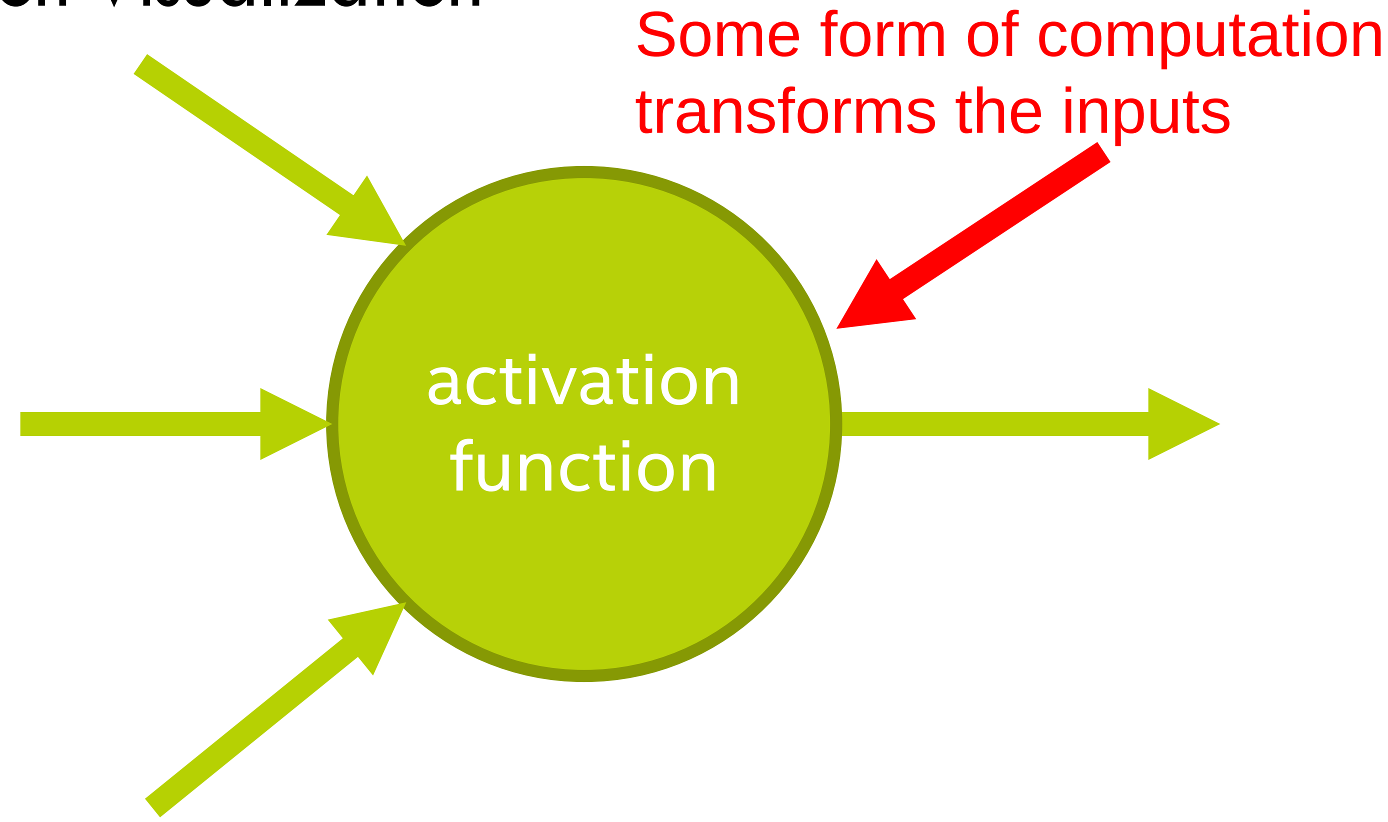


Basic Neuron Visualization

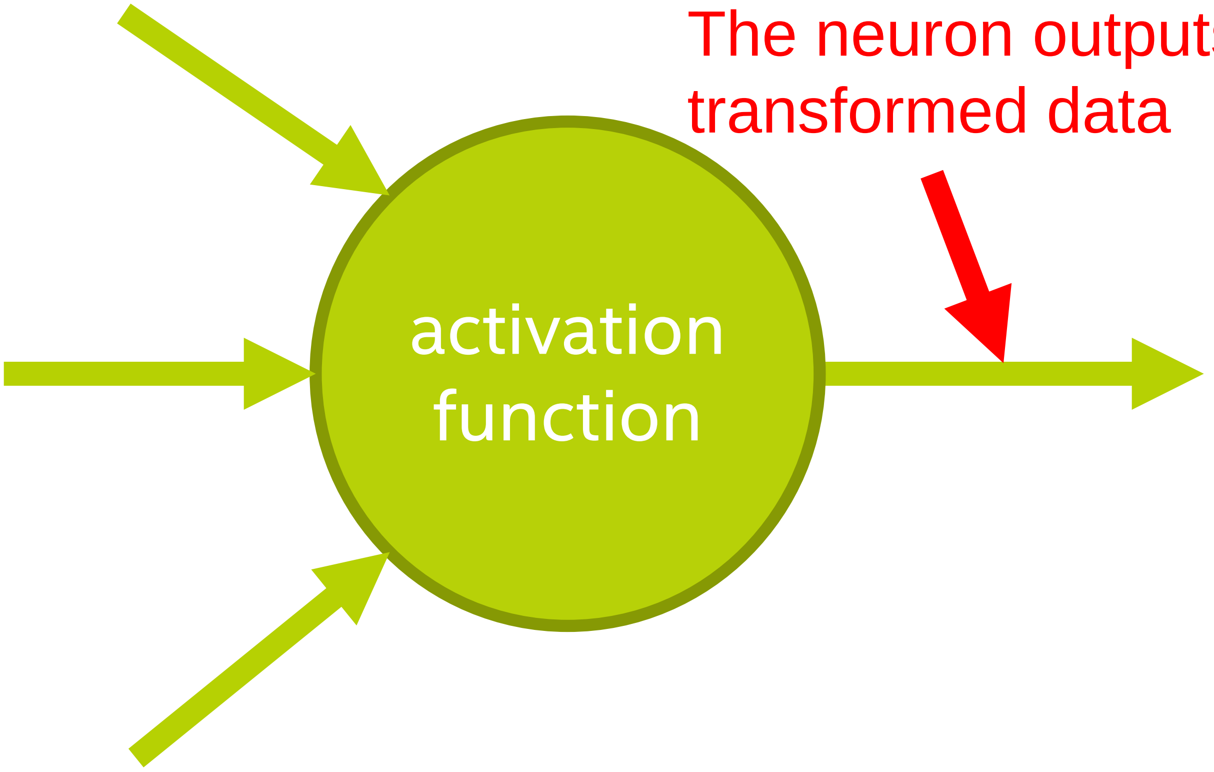
Data from previous layer



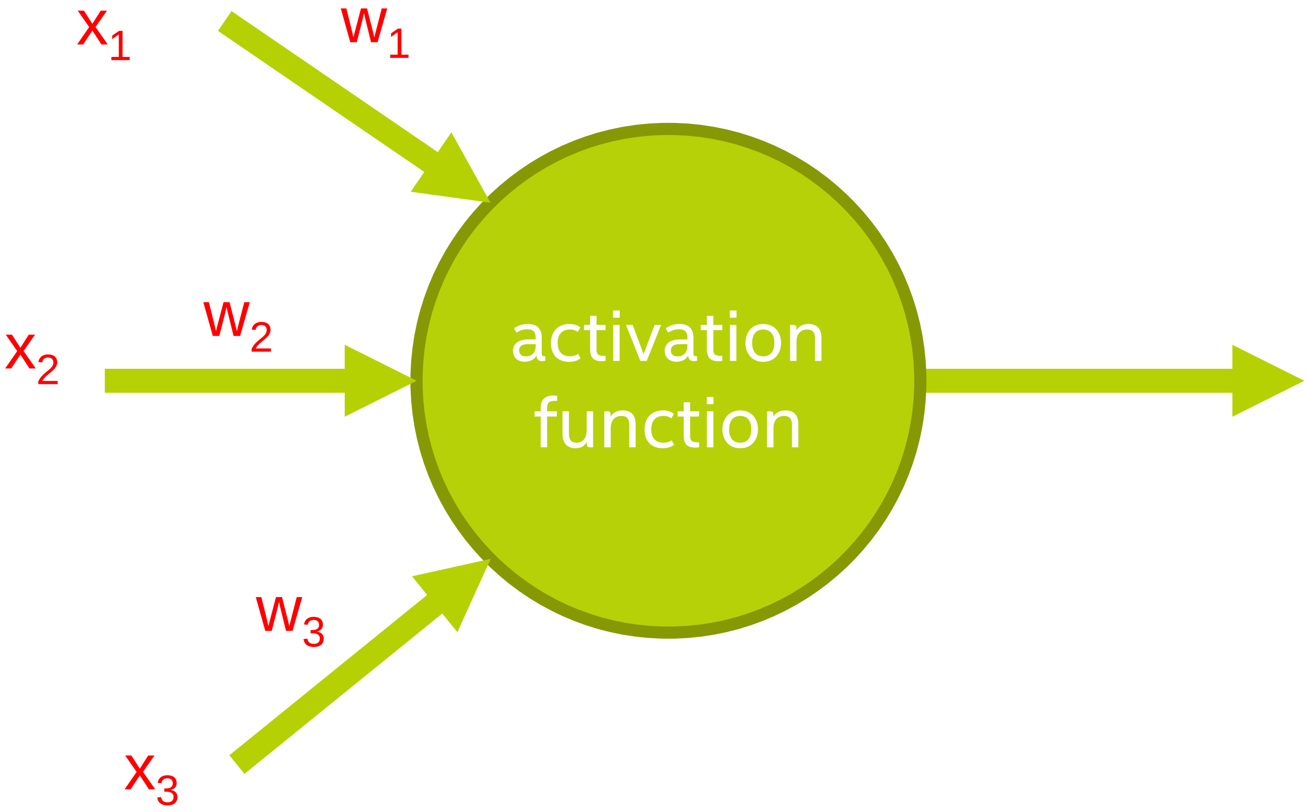
Basic Neuron Visualization



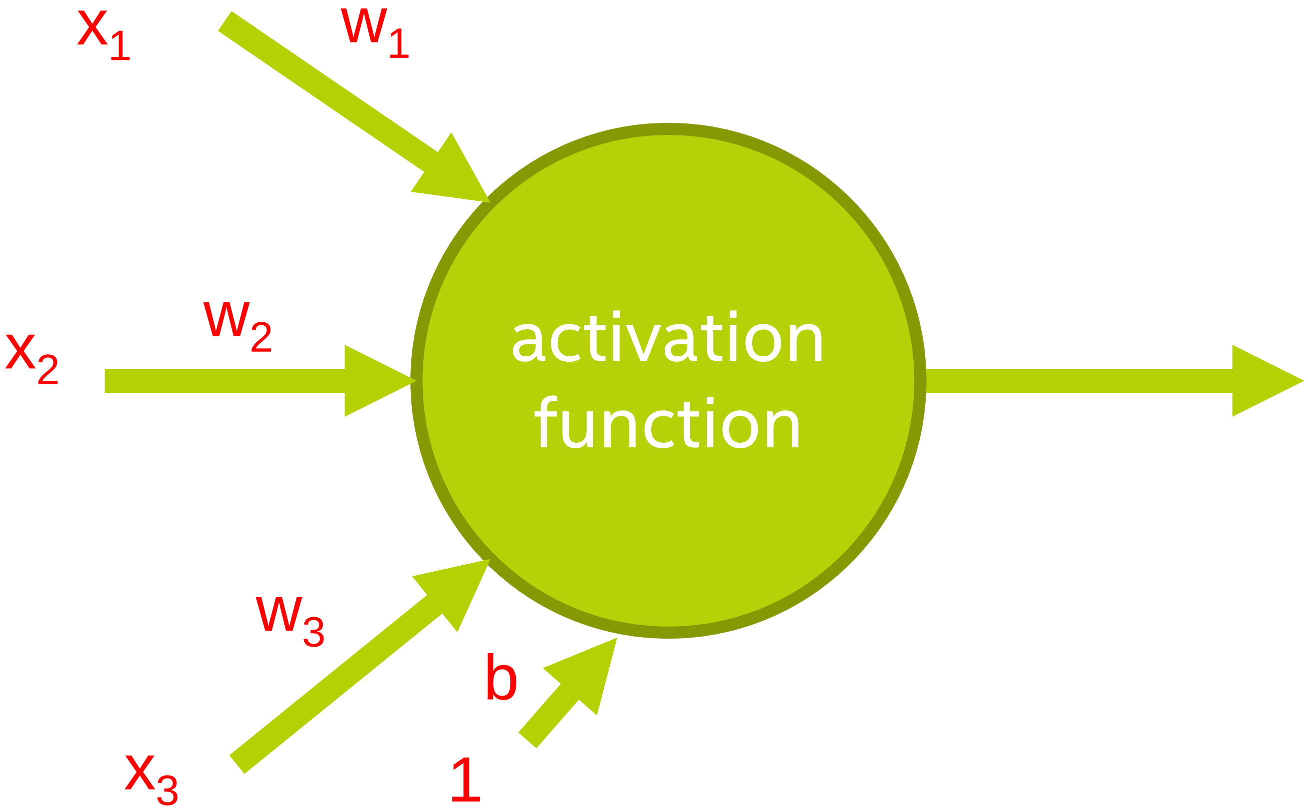
Basic Neuron Visualization



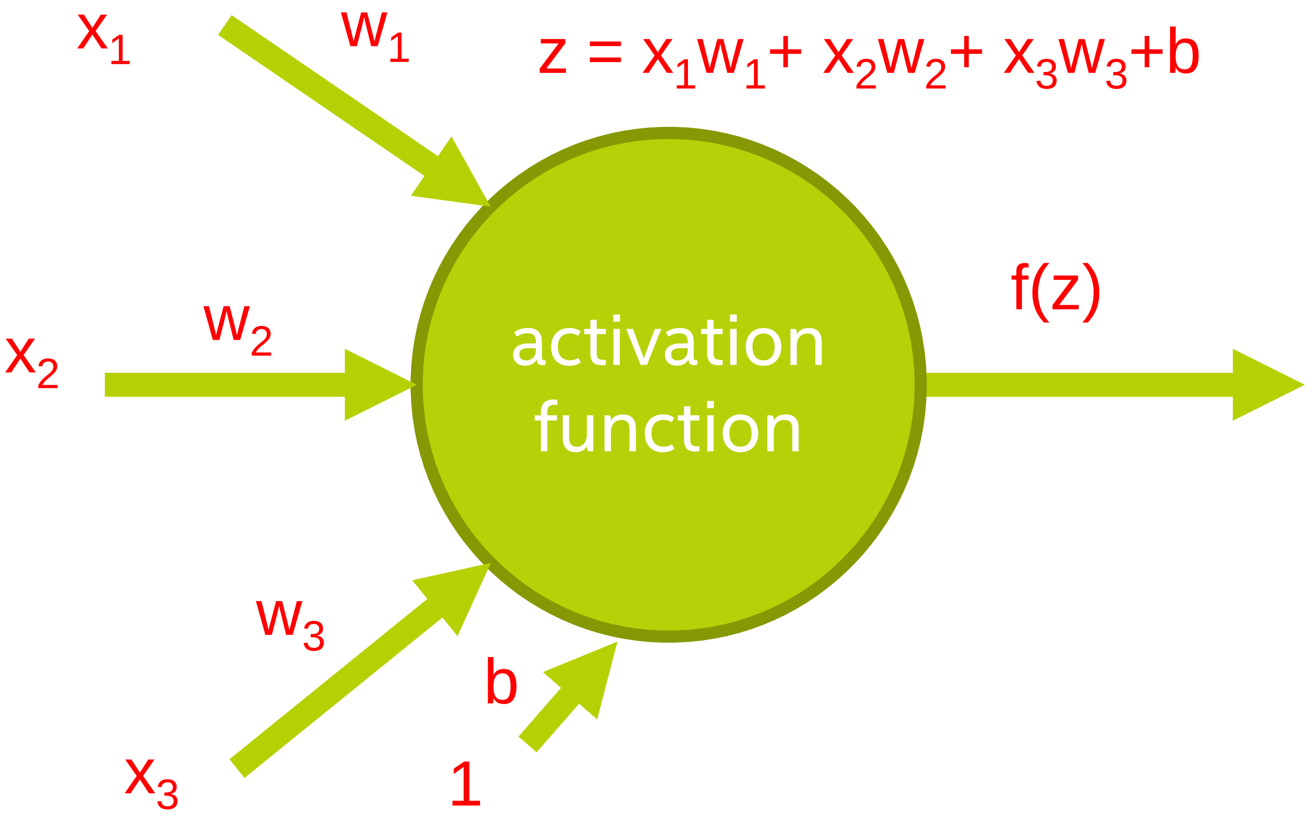
Basic Neuron Visualization



Basic Neuron Visualization



Basic Neuron Visualization



In Vector Notation

z = “net input”

b = “bias term”

f = activation function

a = output to next layer

$$z = b + \sum_{i=1}^{m} x_i w_i$$

$$z = b + x^T w$$

$$a = f(z)$$

Relation to Logistic Regression

When we choose: $f(z) = \frac{1}{1+e^{-z}}$

$$z = b + \sum_{i=1}^{m} x_i w_i = x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b$$

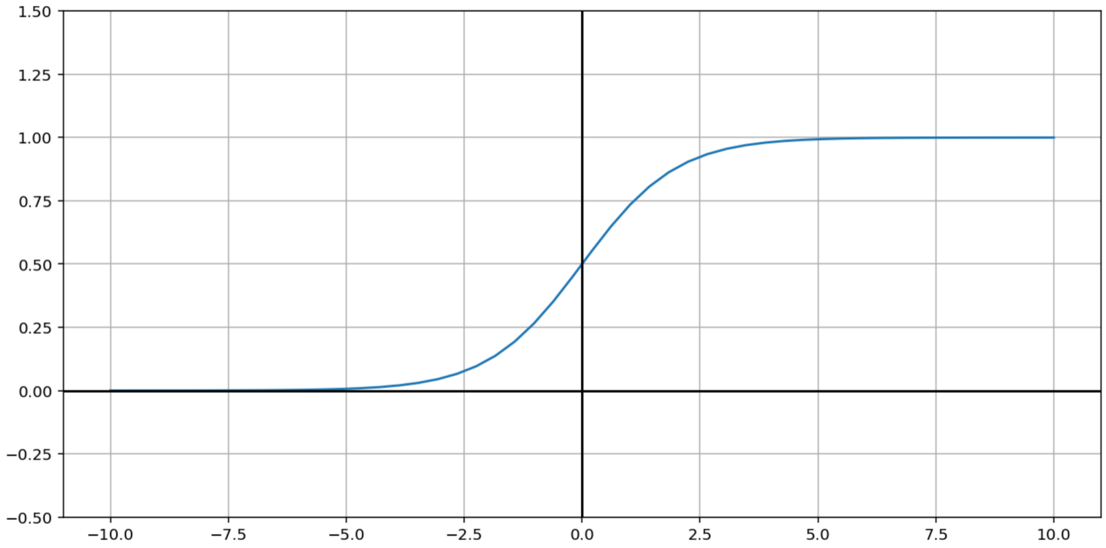
Then a neuron is simply a ”unit” of logistic regression!

weights ⇔ coefficients   inputs ⇔ variables

bias term ⇔ constant term

Relation to Logistic Regression

This is called the "sigmoid" function: $\sigma(z) = \frac{1}{1+e^{-z}}$



Nice Property of Sigmoid Function

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

Quotient rule

$$\frac{d}{dx} \cdot \frac{f(x)}{g(x)} = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$$

$$\sigma'(z) = \frac{0-(-e^{-z})}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2}$$

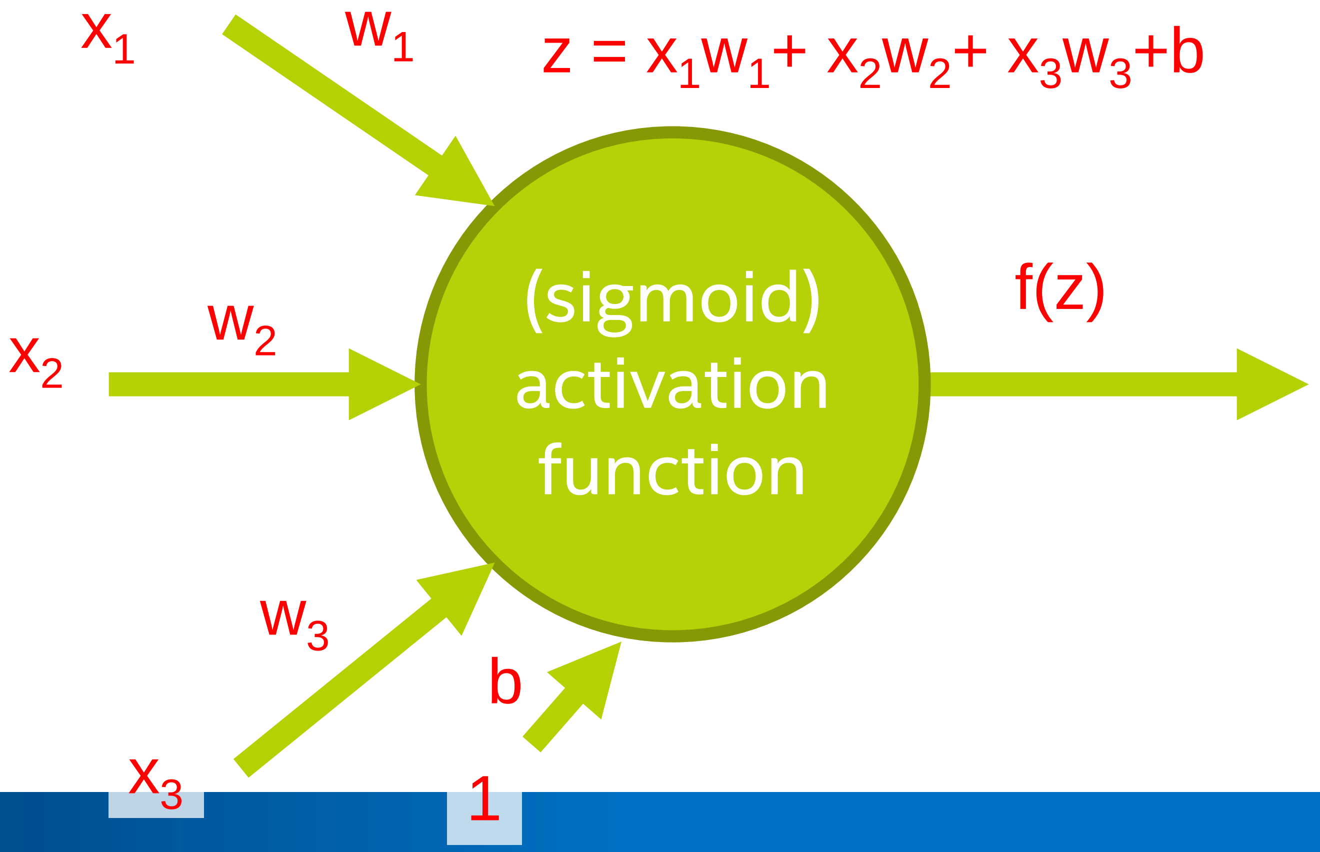
$$= \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \cancel{\frac{1+e^{-z}}{(1+e^{-z})^2}} - \frac{1}{(1+e^{-z})^2}$$

$$= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$$

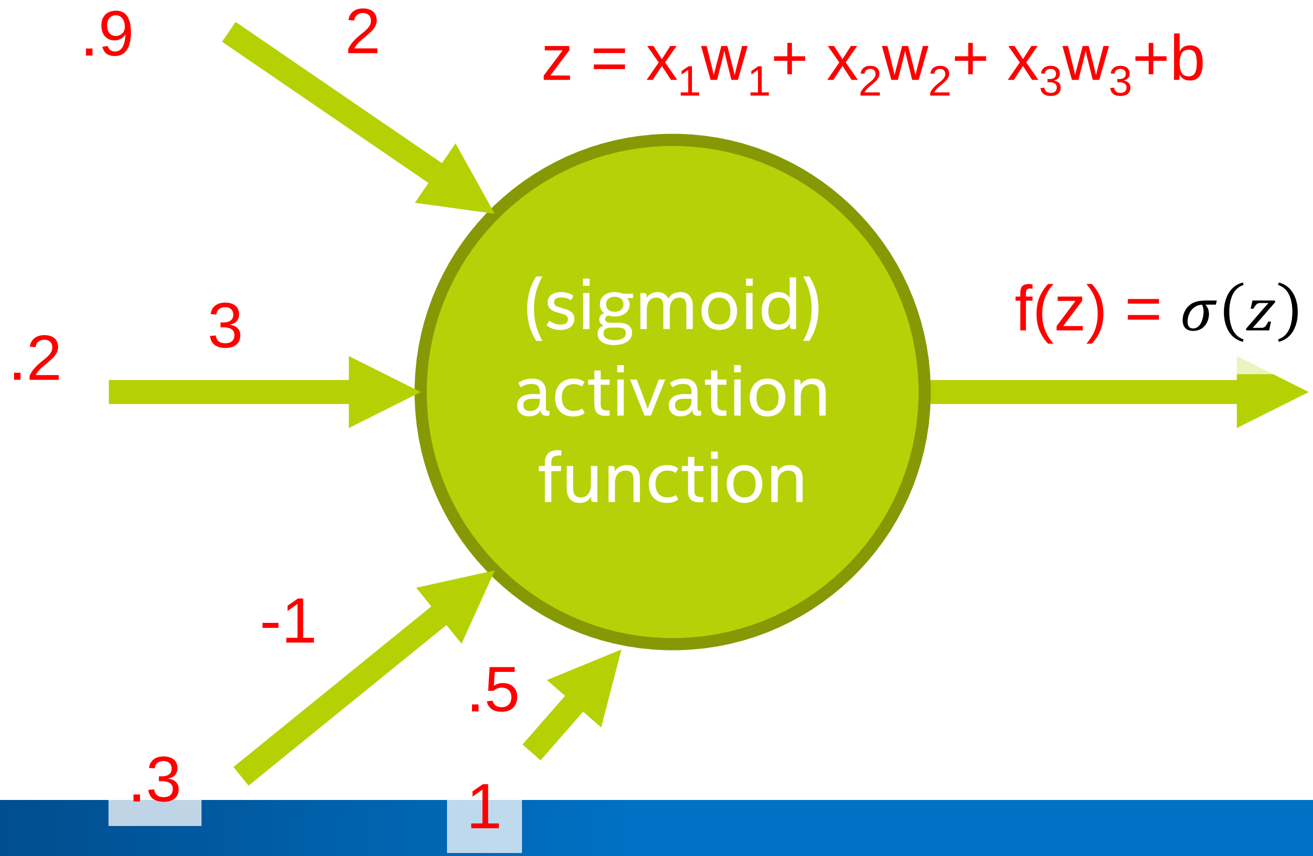
$$\sigma'(z) = \sigma(z)(1-\sigma(z))$$

This will be helpful!

Example Neuron Computation



Example Neuron Computation

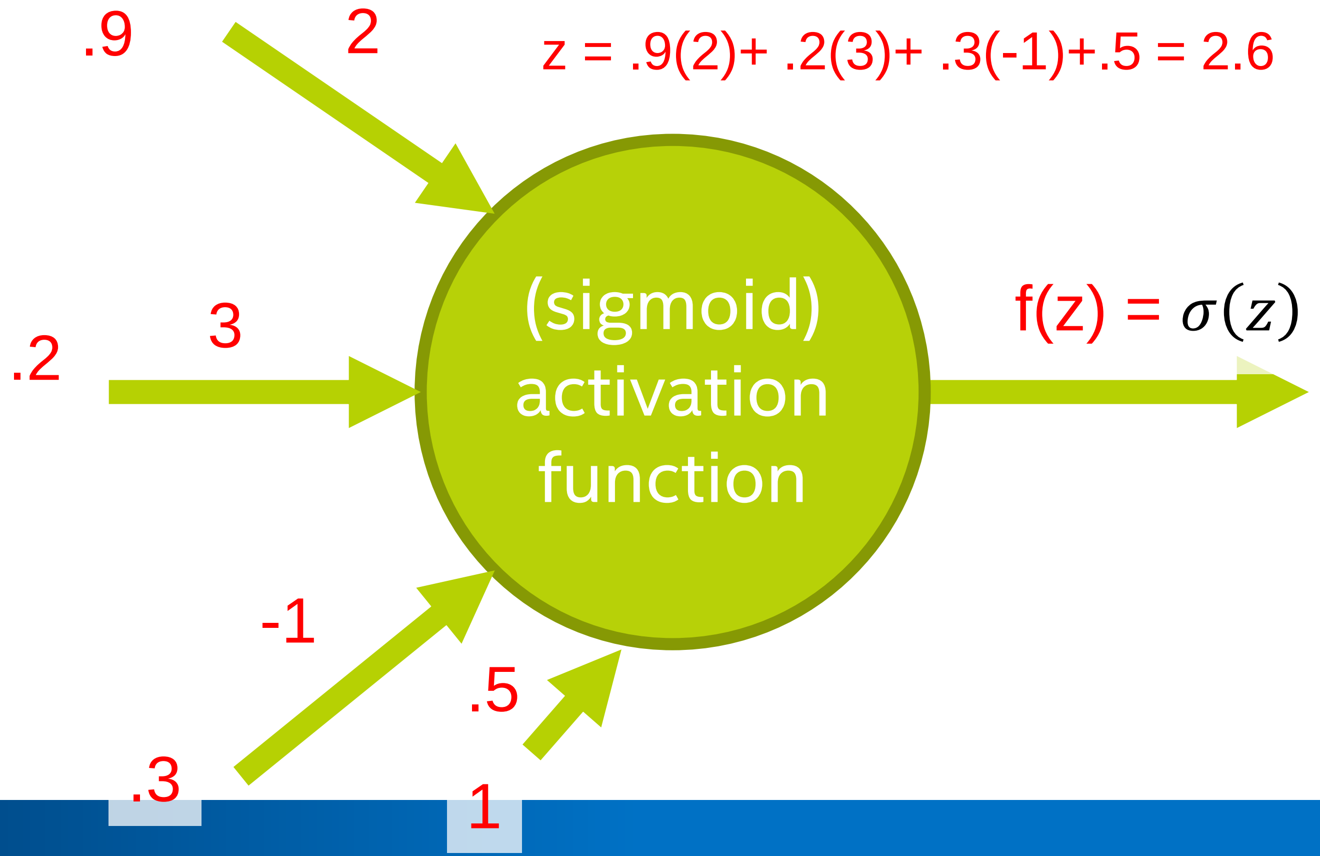


$z = x_1w_1 + x_2w_2 + x_3w_3 + b$

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

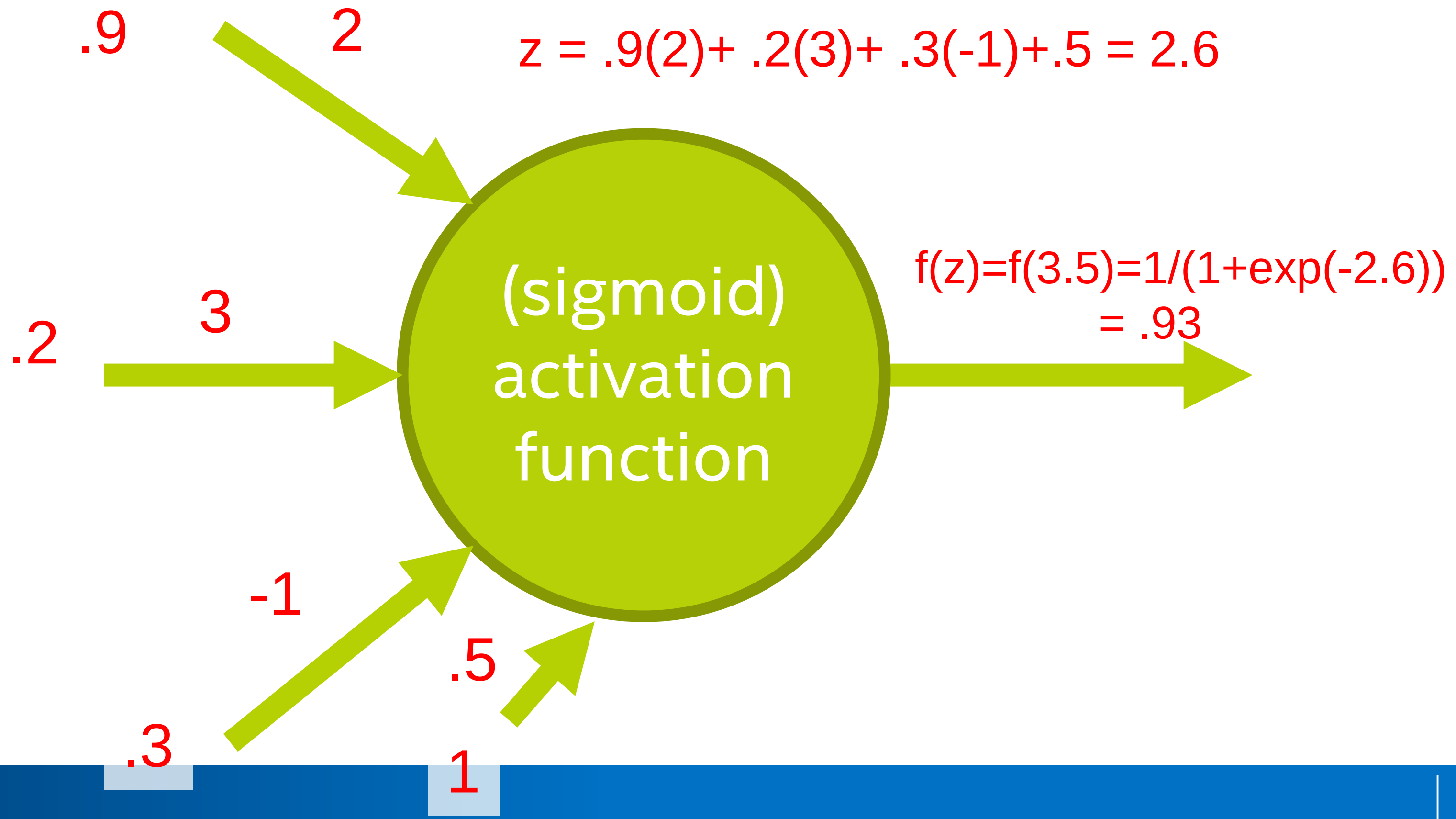
Example Neuron Computation

$z = .9(2) + .2(3) + .3(-1) + .5 = 2.6$



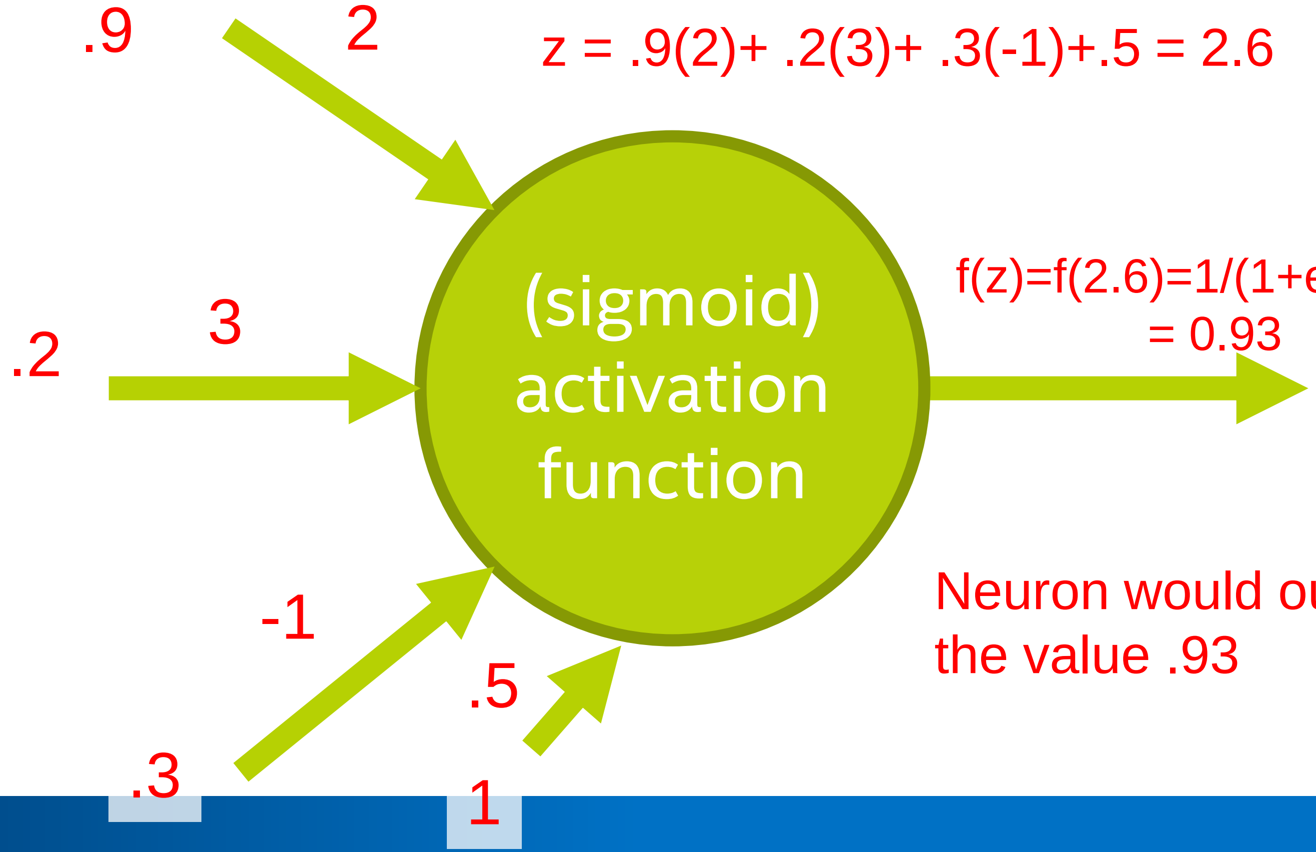
$f(z) = \sigma(z) = \frac{1}{1+e^{-z}}$

Example Neuron Computation



Example Neuron Computation

$z = .9(2) + .2(3) + .3(-1) + .5 = 2.6$

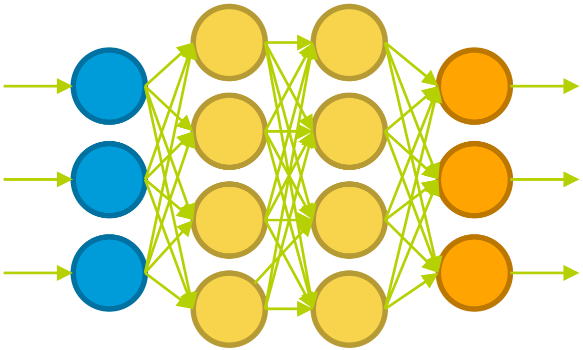


$f(z) = f(2.6) = 1/(1+\exp(-2.6))$
$= 0.93$

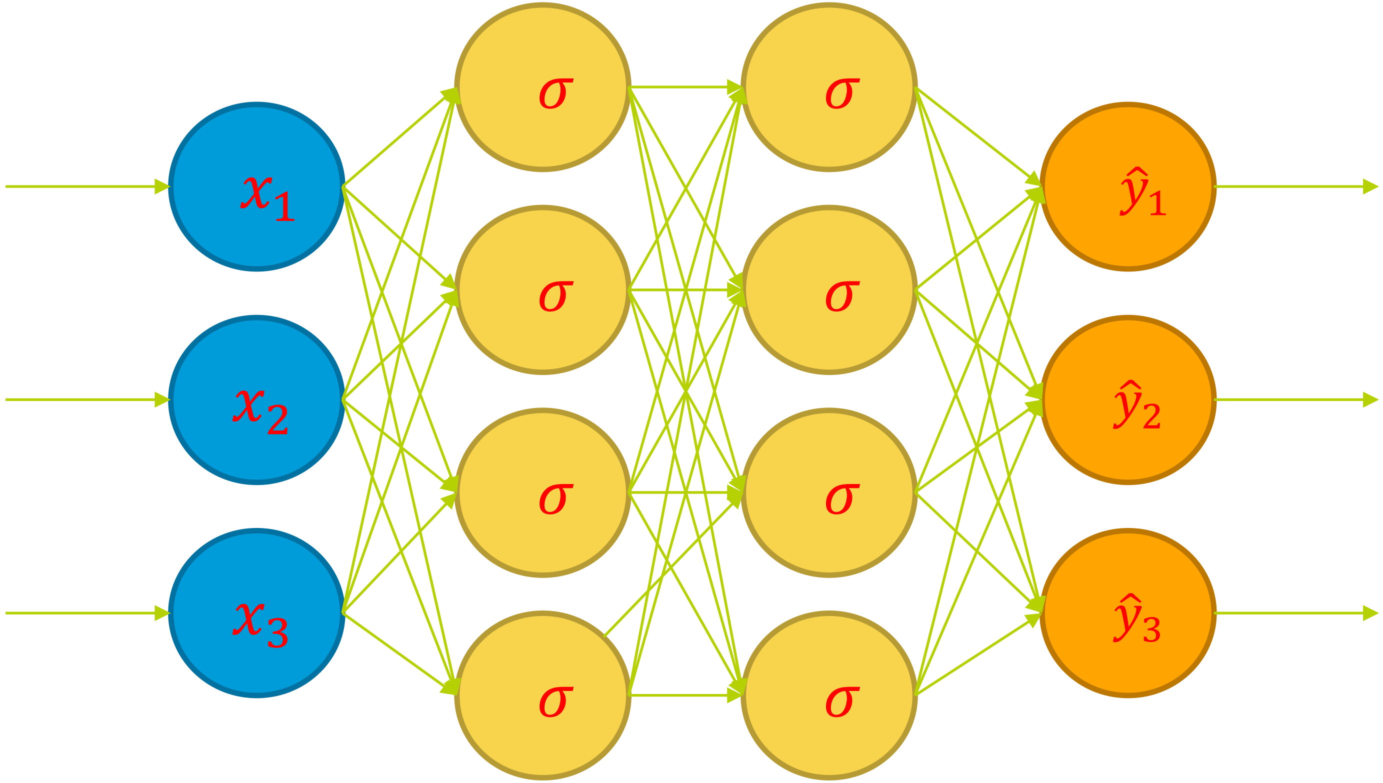
Neuron would output the value .93

Why Neural Nets?

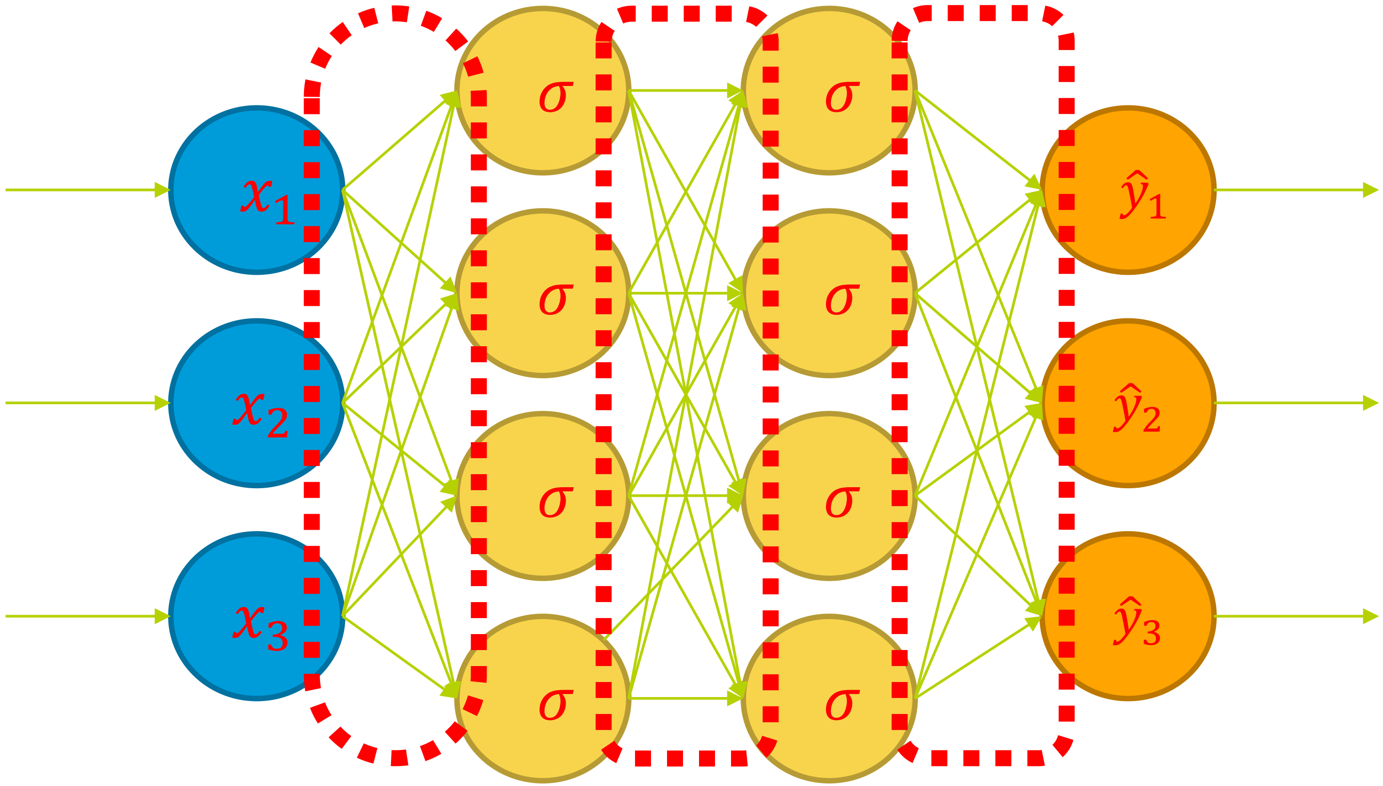
- Why not just use a single neuron? Why do we need a larger network?
- A single neuron (like logistic regression) only permits a linear decision boundary.
- Most real-world problems are considerably more complicated!



Feedforward Neural Network

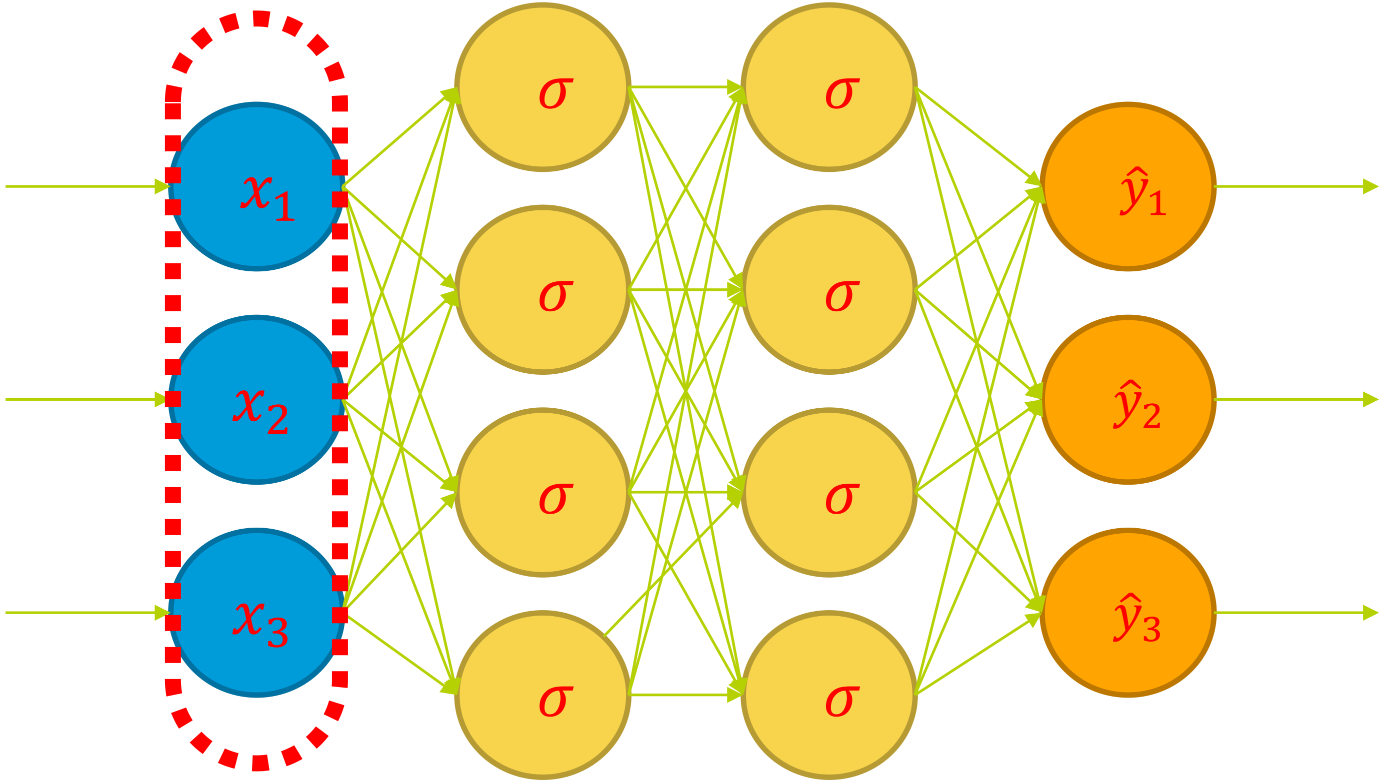


Weights



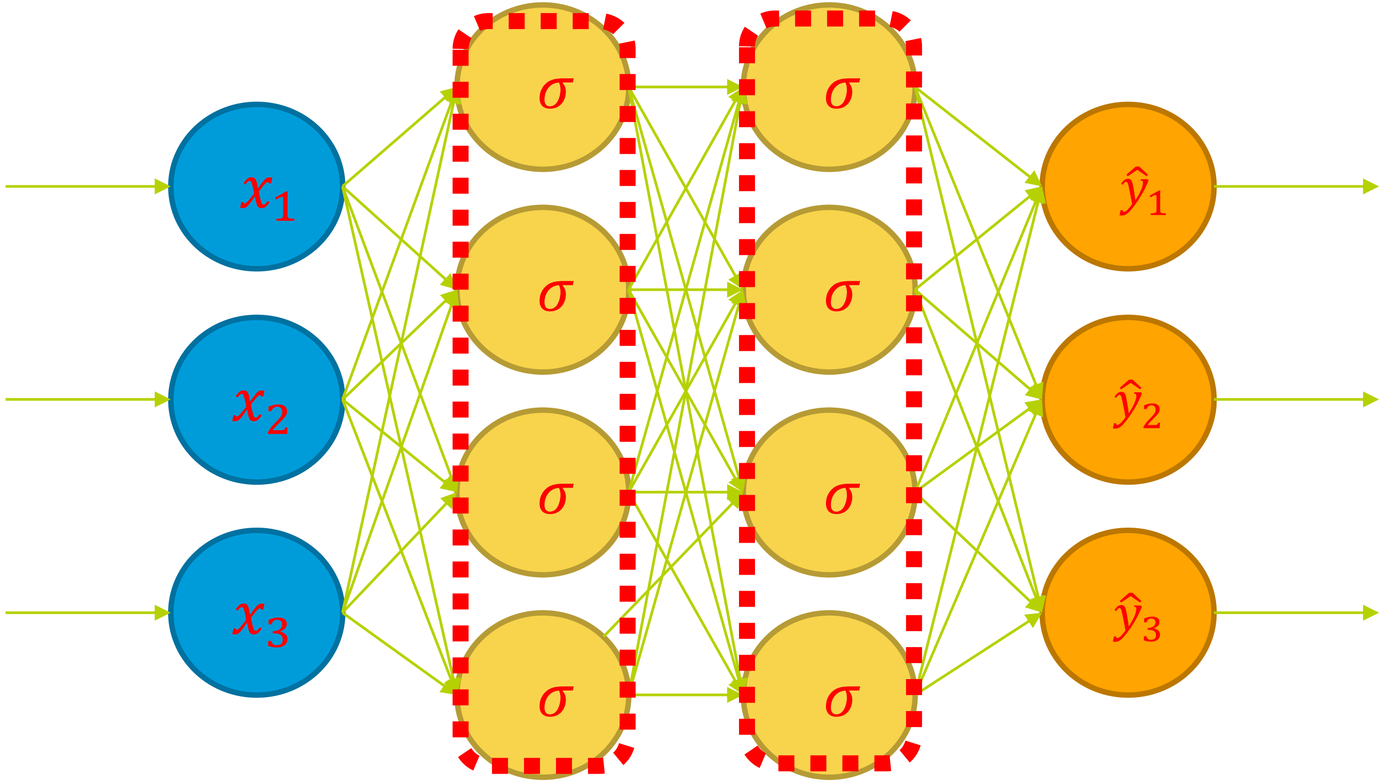
x_1
x_2
x_3
σ
σ
σ
σ
σ
σ
σ
σ
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$

Input Layer

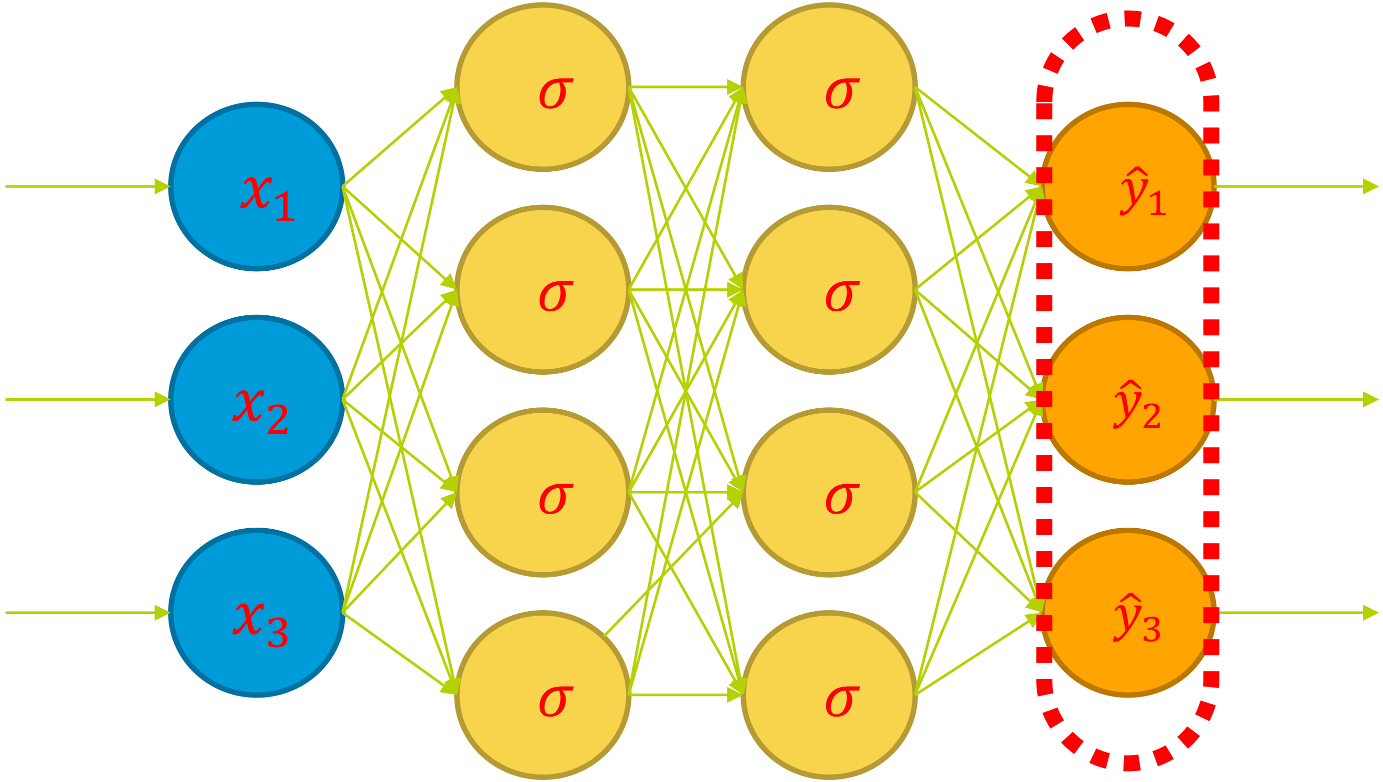


x_1
x_2
x_3
σ
σ
σ
σ
σ
σ
σ
σ
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$

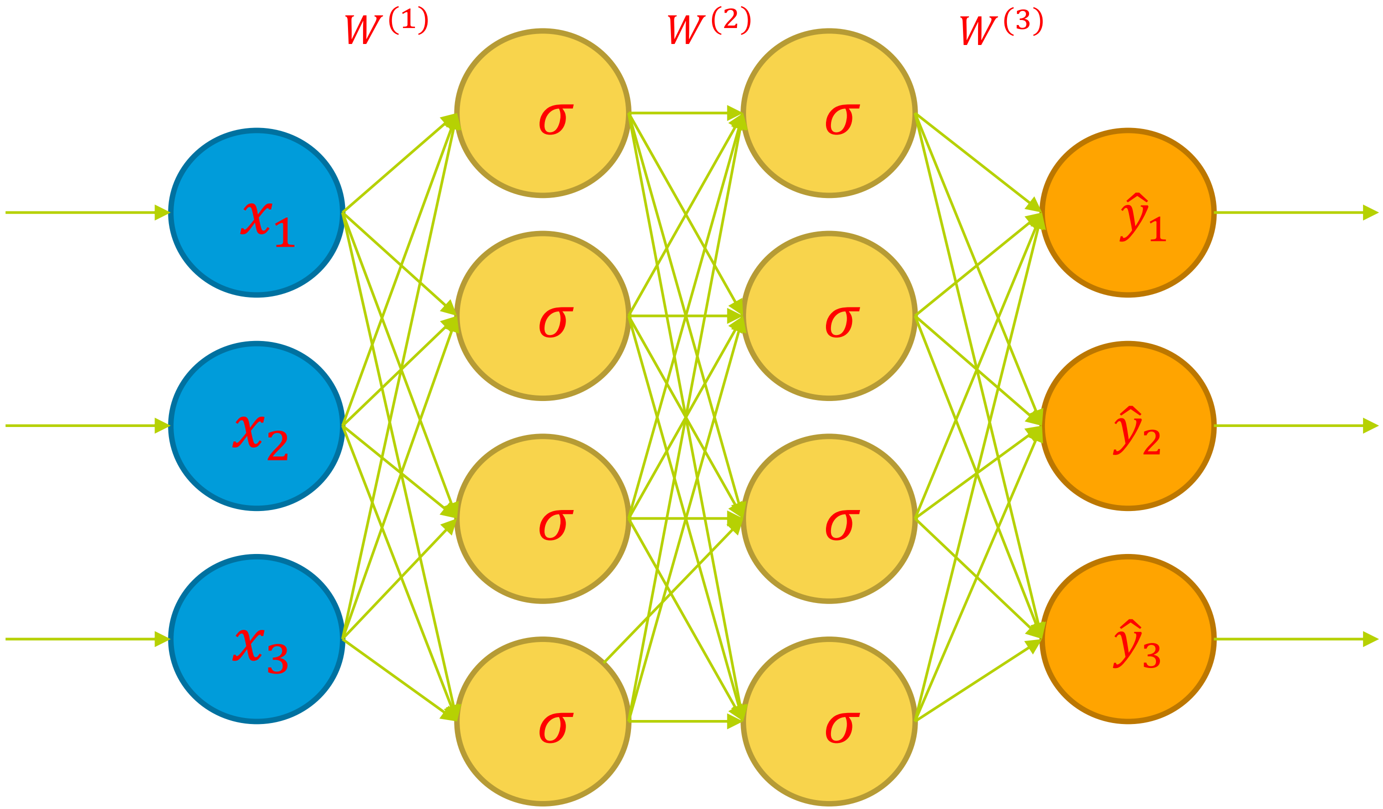
Hidden Layers



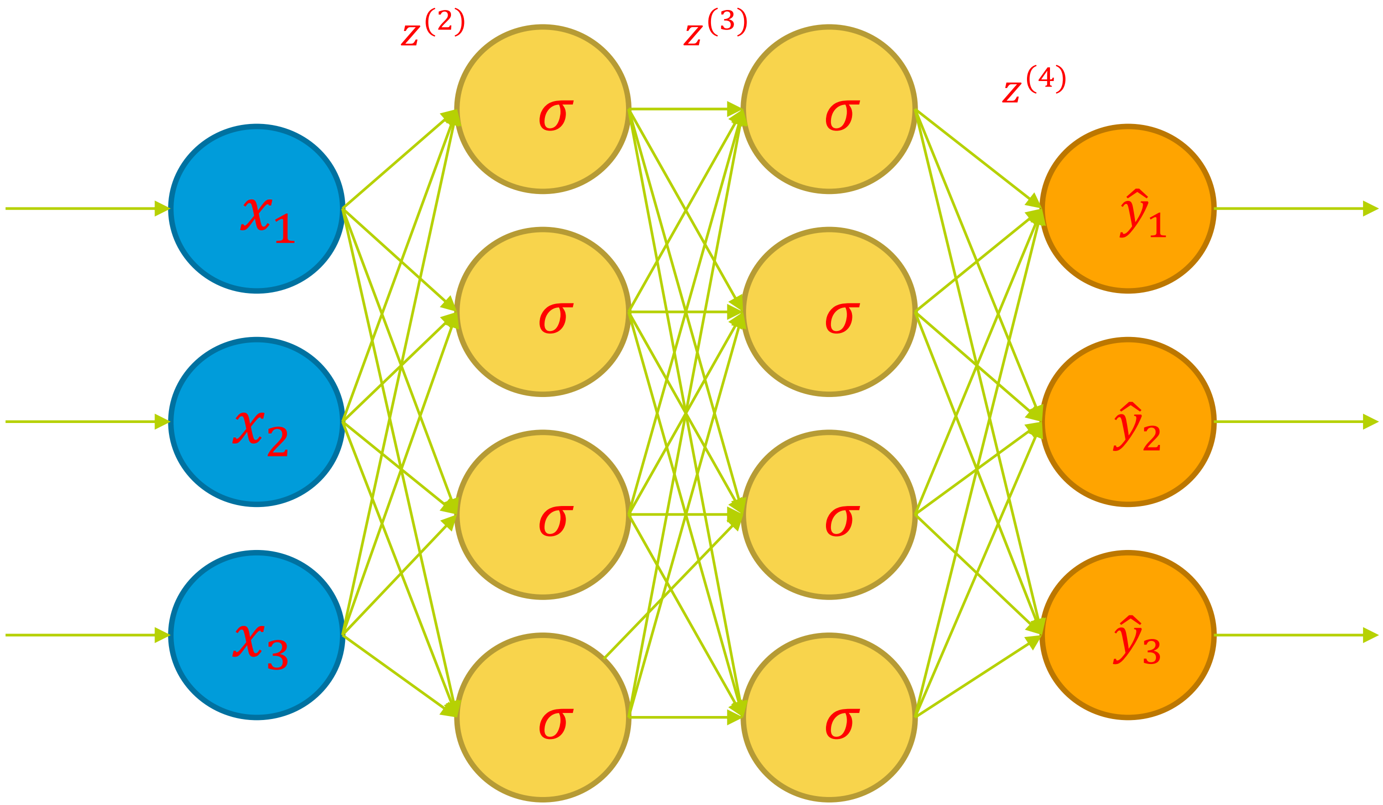
Output Layer



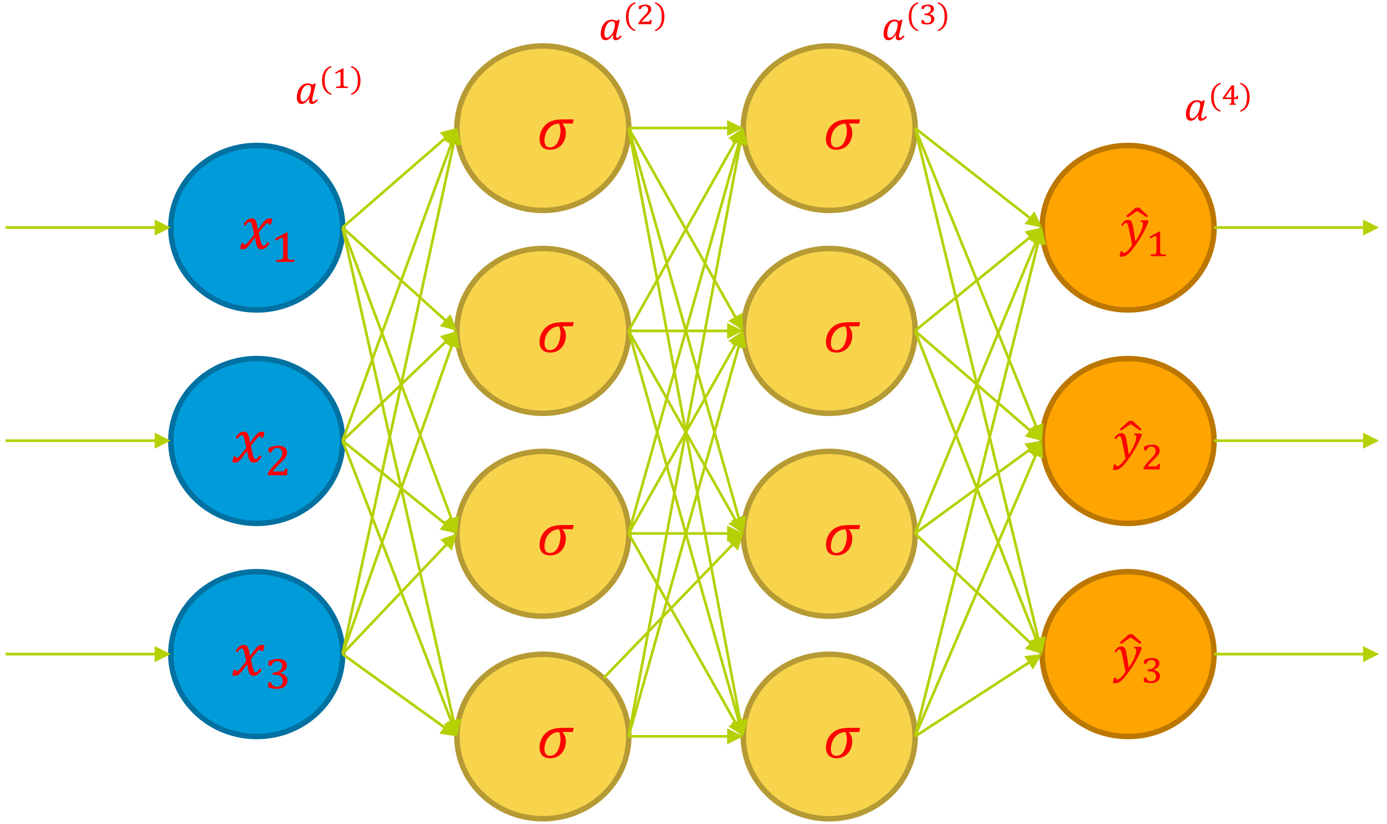
Weights (represented by matrices)



Net Input (sum of weighted inputs, before activation function)



Activations (output of neurons to next layer)



Matrix representation of computation

x

$(x = a^{(1)})$

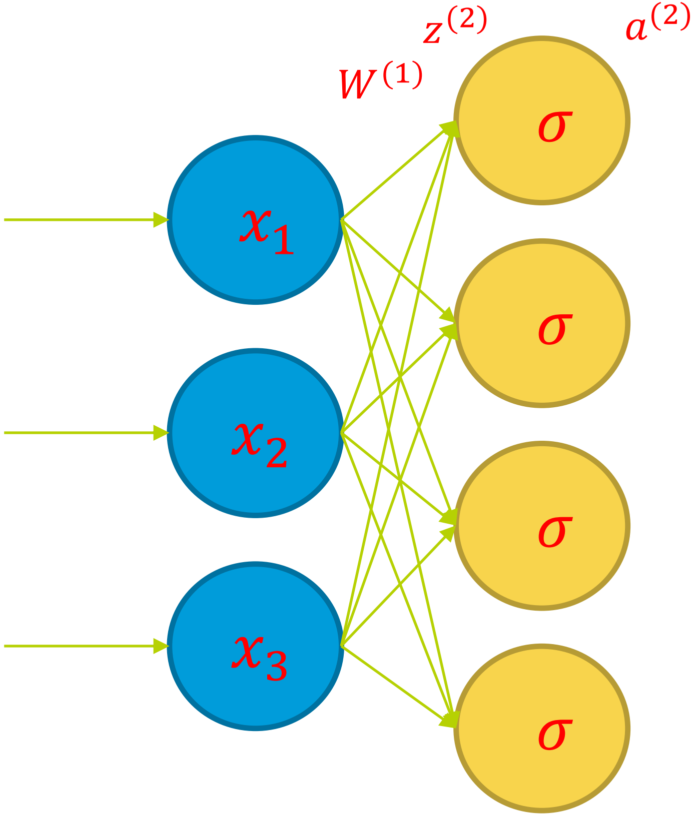
$z^{(2)} = xW^{(1)}$

$a^{(2)} = \sigma(z^{(2)})$

$W^{(1)}$ is a 3x4 matrix

$z^{(2)}$ is a 4-vector

$a^{(2)}$ is a 4-vector



Tensor flow playground

https://playground.tensorflow.org/

Continuing the Computation

For a single training instance (data point)

Input: vector x (a row vector of length 3)

Output: vector $\hat{y}$ (a row vector of length 3)

$$z^{(2)} = xW^{(1)} \qquad a^{(2)} = \sigma(z^{(2)})$$

$$z^{(3)} = a^{(2)}W^{(2)} \qquad a^{(3)} = \sigma(z^{(3)})$$

$$z^{(4)} = a^{(3)}W^{(3)} \qquad \hat{y} = softmax(z^{(4)})$$

Multiple data points

In practice, we do these computation for many data points at the same time, by "stacking" the rows into a matrix.

But the equations look the same!

Input: matrix x (an nx3 matrix) (each row a single instance)

Output: vector $\hat{y}$ (an nx3 matrix) (each row a single prediction)

$$z^{(2)} = xW^{(1)} \qquad a^{(2)} = \sigma(z^{(2)})$$

$$z^{(3)} = a^{(2)}W^{(2)} \qquad a^{(3)} = \sigma(z^{(3)})$$

$$z^{(4)} = a^{(3)}W^{(3)} \qquad \hat{y} = softmax(z^{(4)})$$

Now we know how feedforward NNs do Computations.

Next, we will learn how to adjust the weights to learn from data.