

# Introduction to Neural Nets

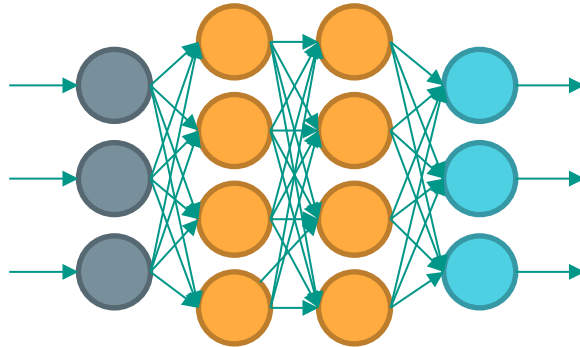
---



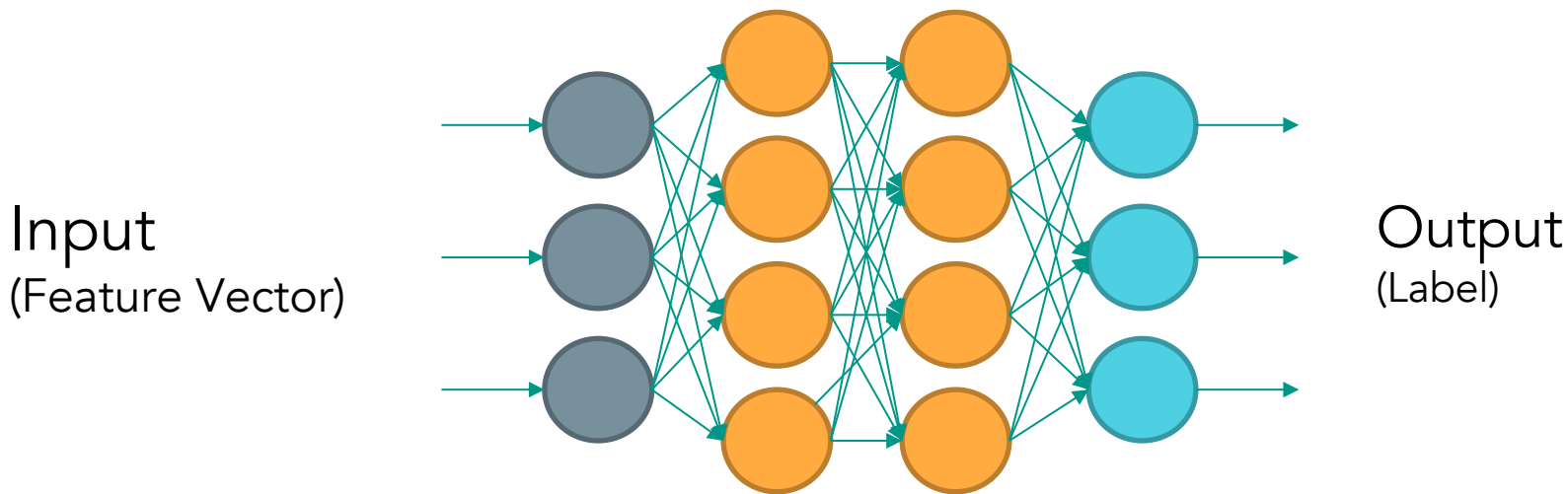
**METIS**

# 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



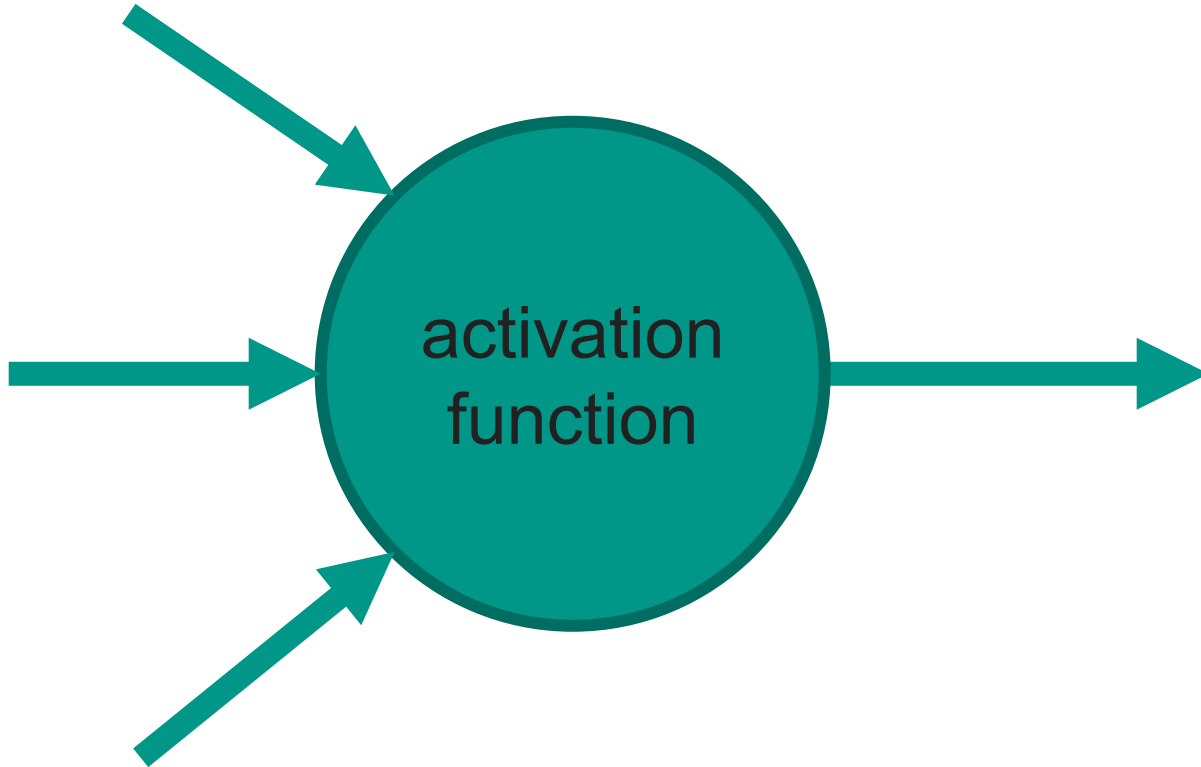
# Neural Net Structure



- 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

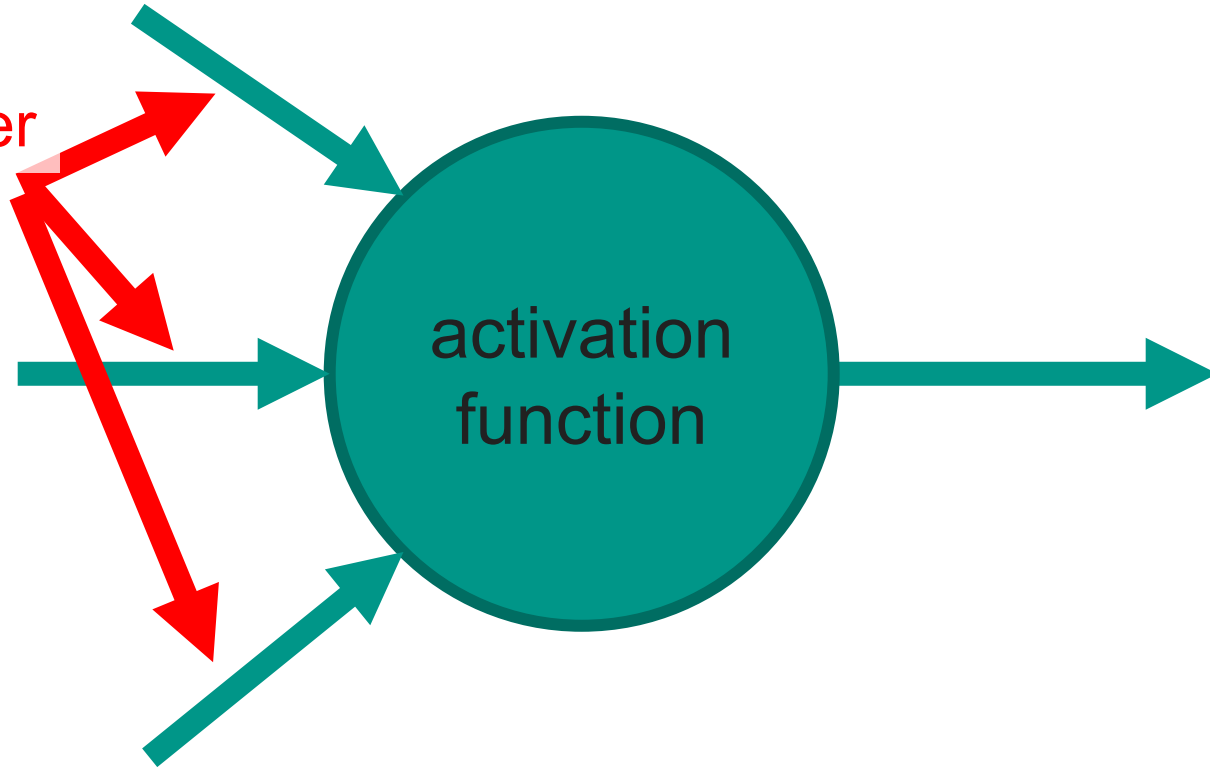


# Basic Neuron Visualization

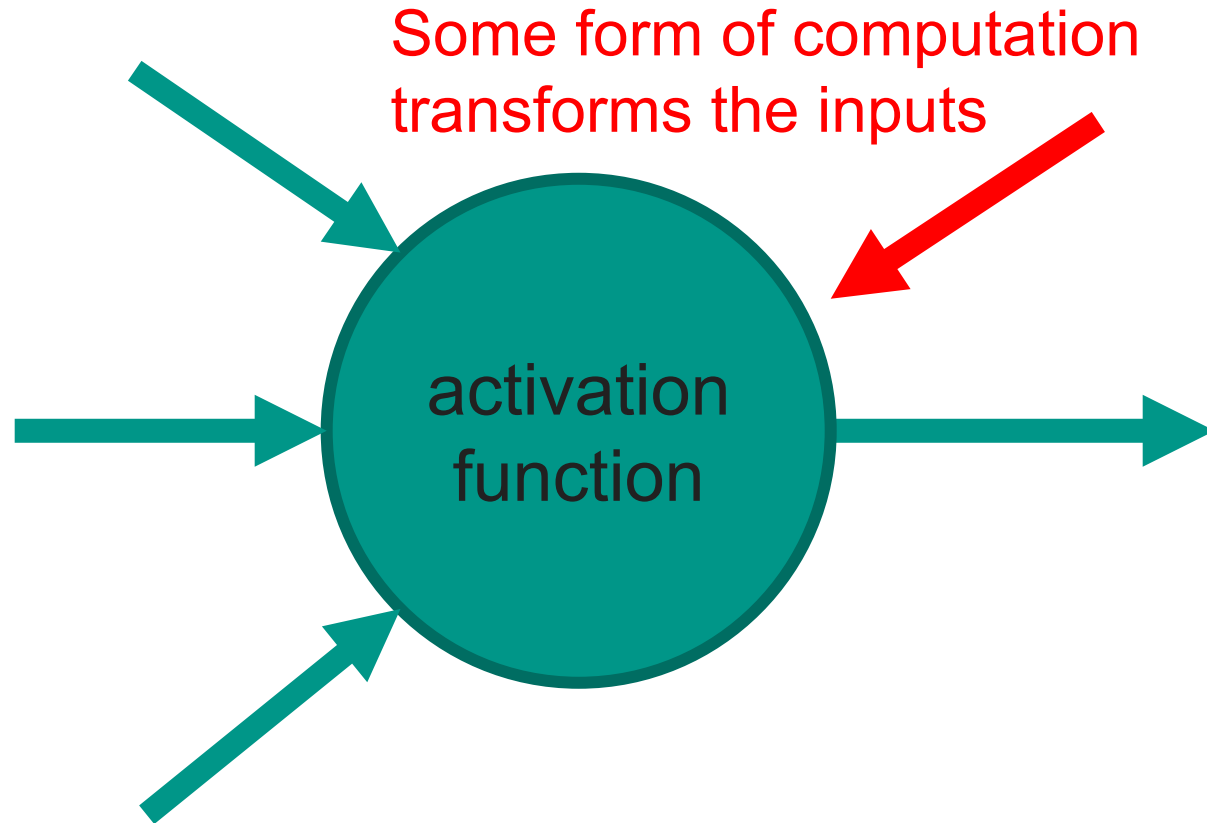


# Basic Neuron Visualization

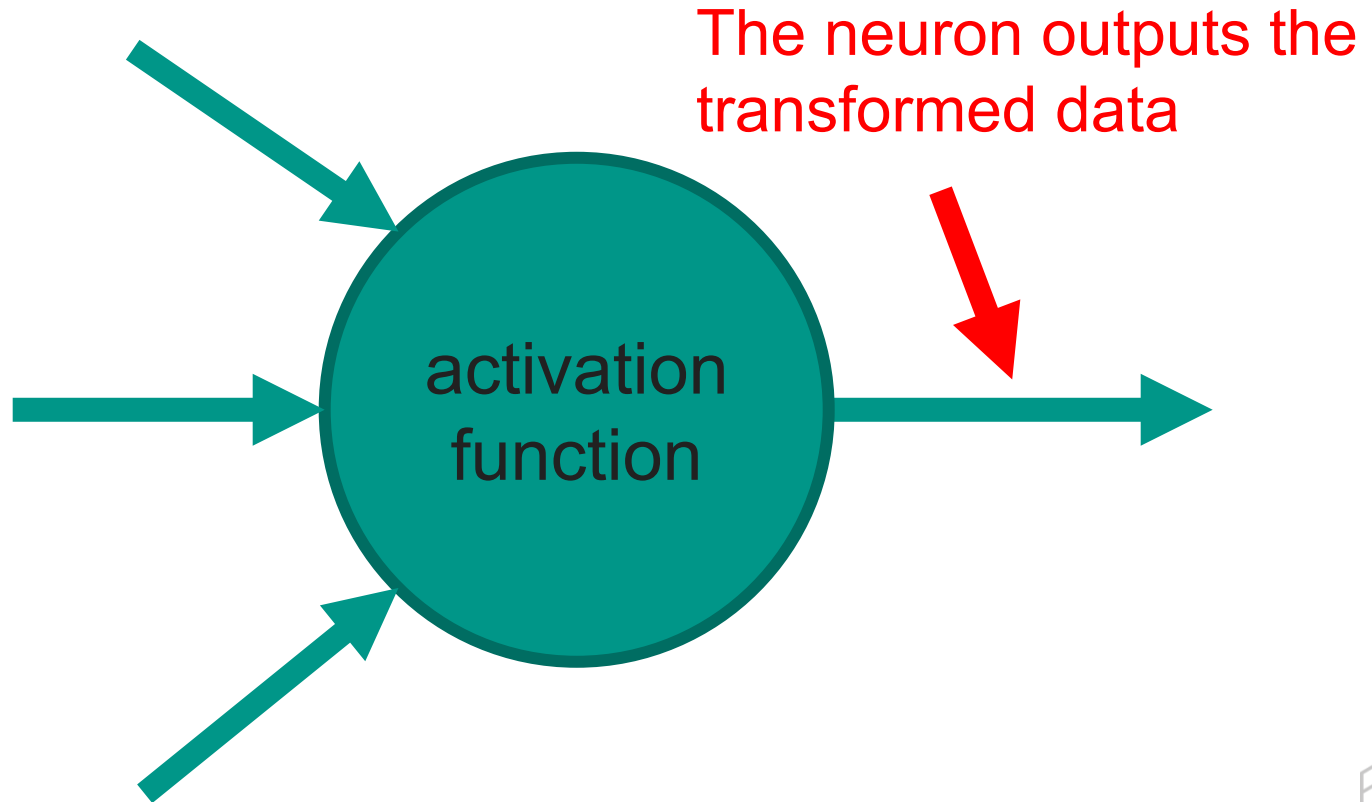
Data from  
previous layer



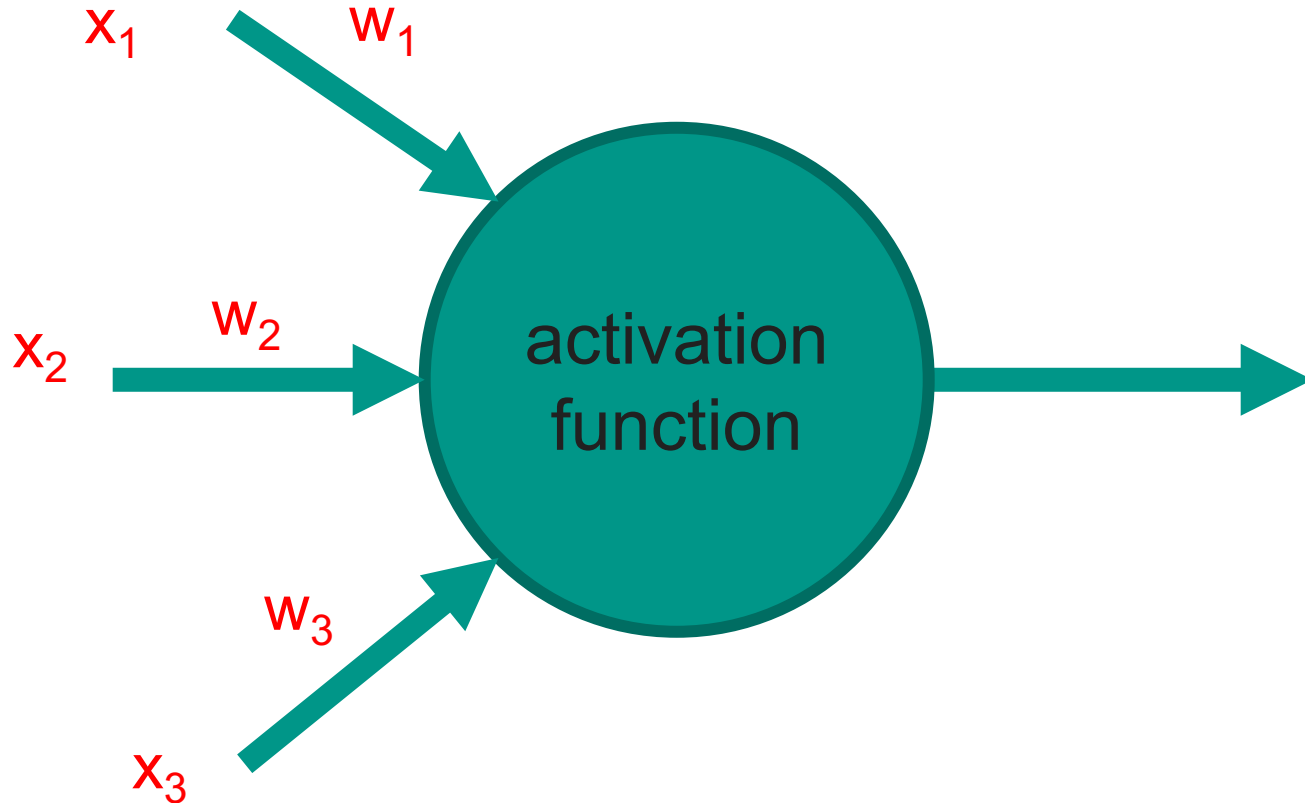
# Basic Neuron Visualization



# Basic Neuron Visualization

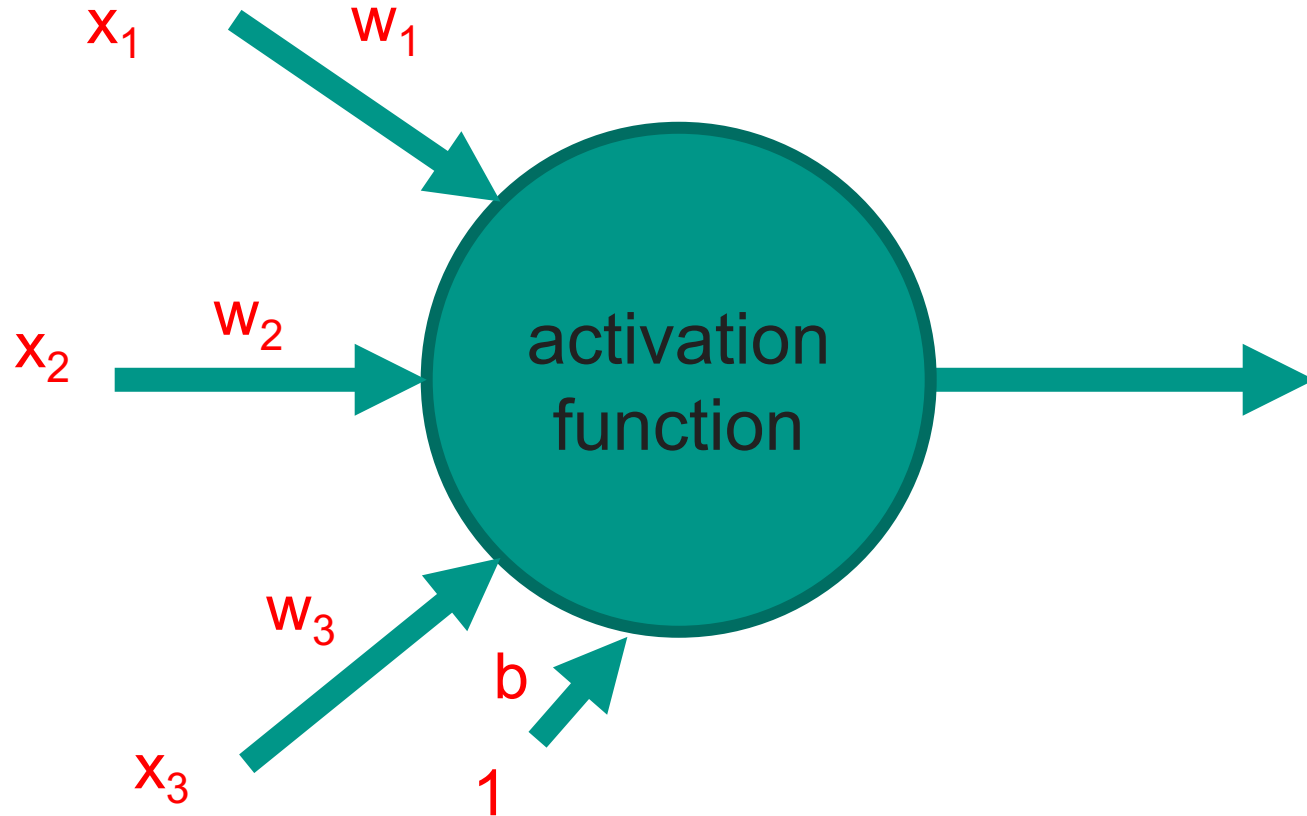


# Basic Neuron Visualization

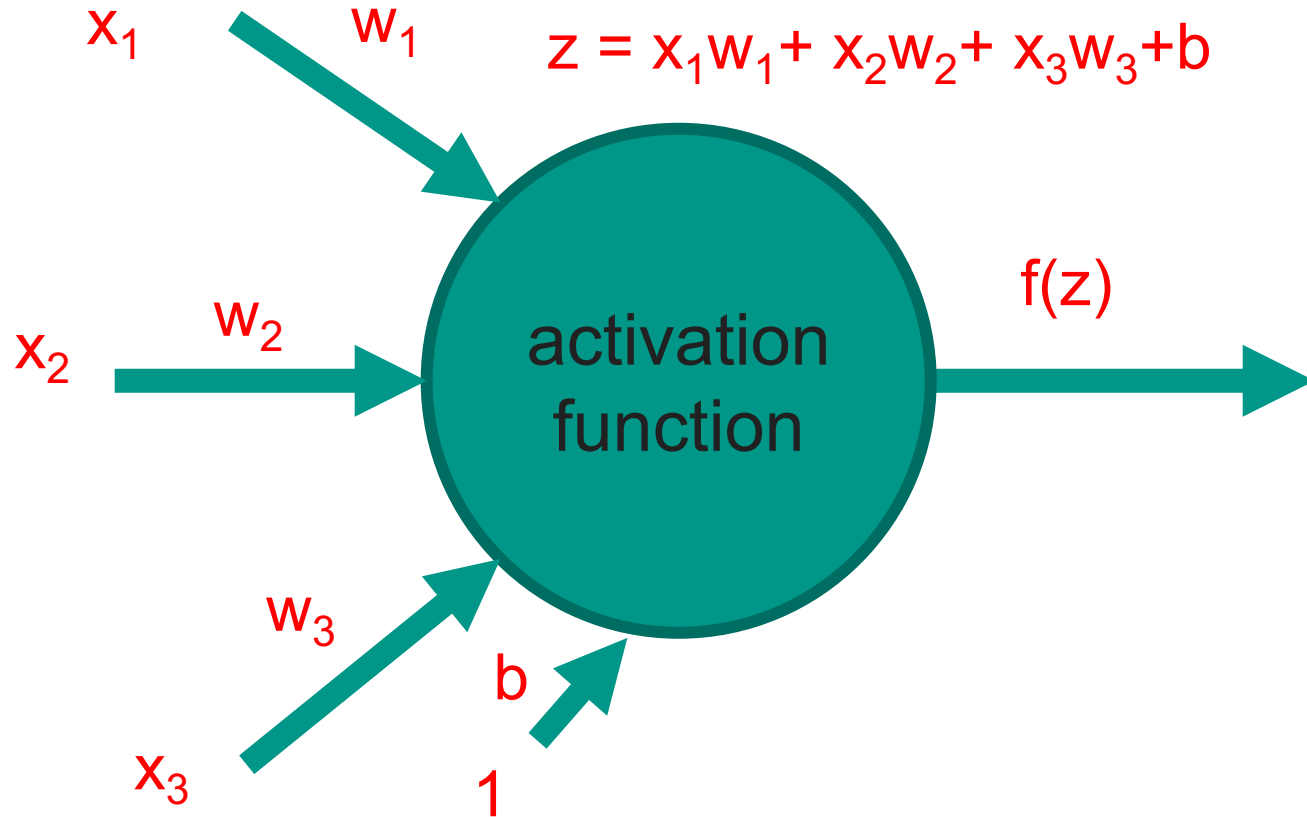




# Basic Neuron Visualization



# 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!

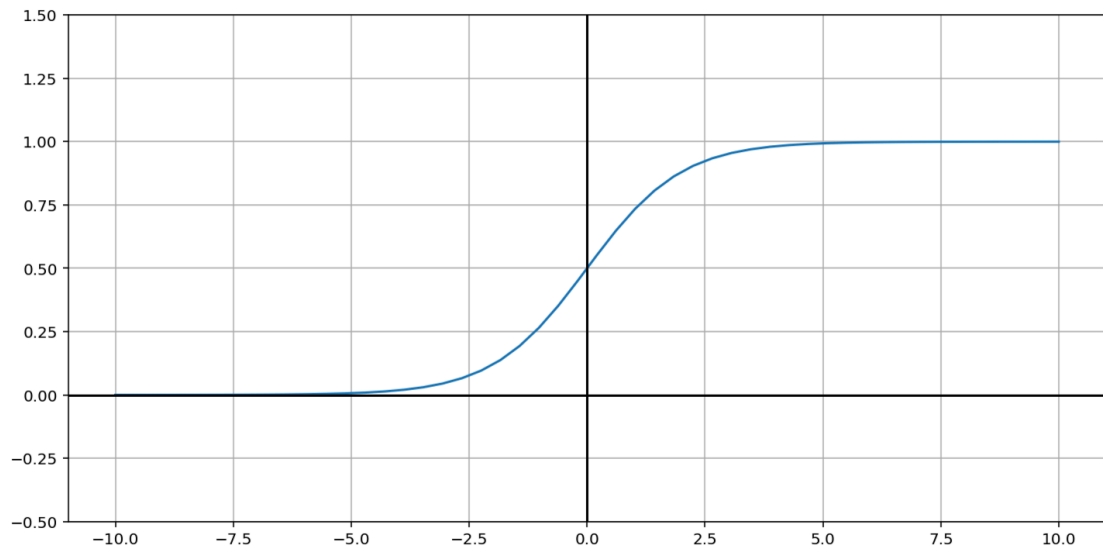
weights  $\Leftrightarrow$  coefficients      inputs  $\Leftrightarrow$  variables

bias term  $\Leftrightarrow$  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

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

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

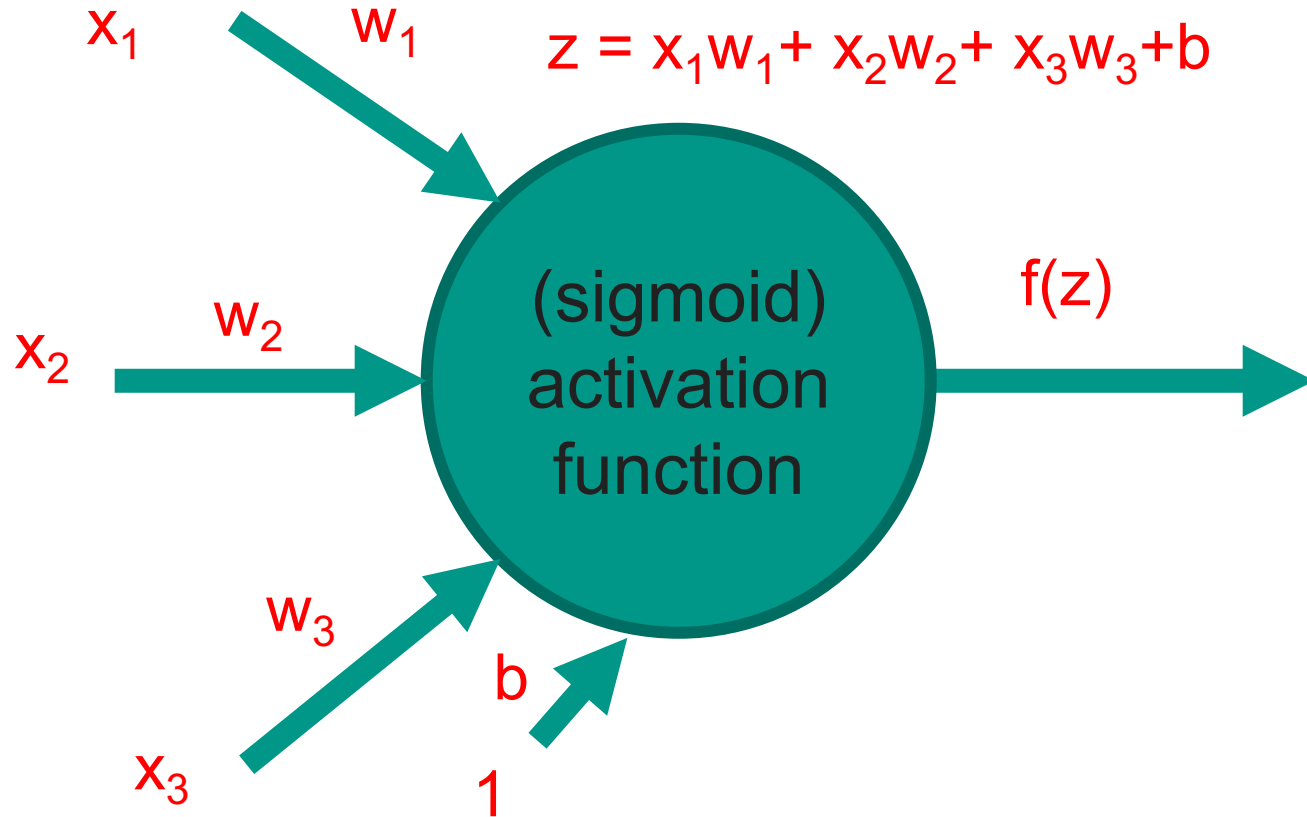
$$= \frac{1 + e^{-z} - 1}{(1 + e^{-z})^2} = \frac{\cancel{1 + e^{-z}}}{(1 + e^{-z})^2} - \frac{1}{(1 + e^{-z})^2}$$

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

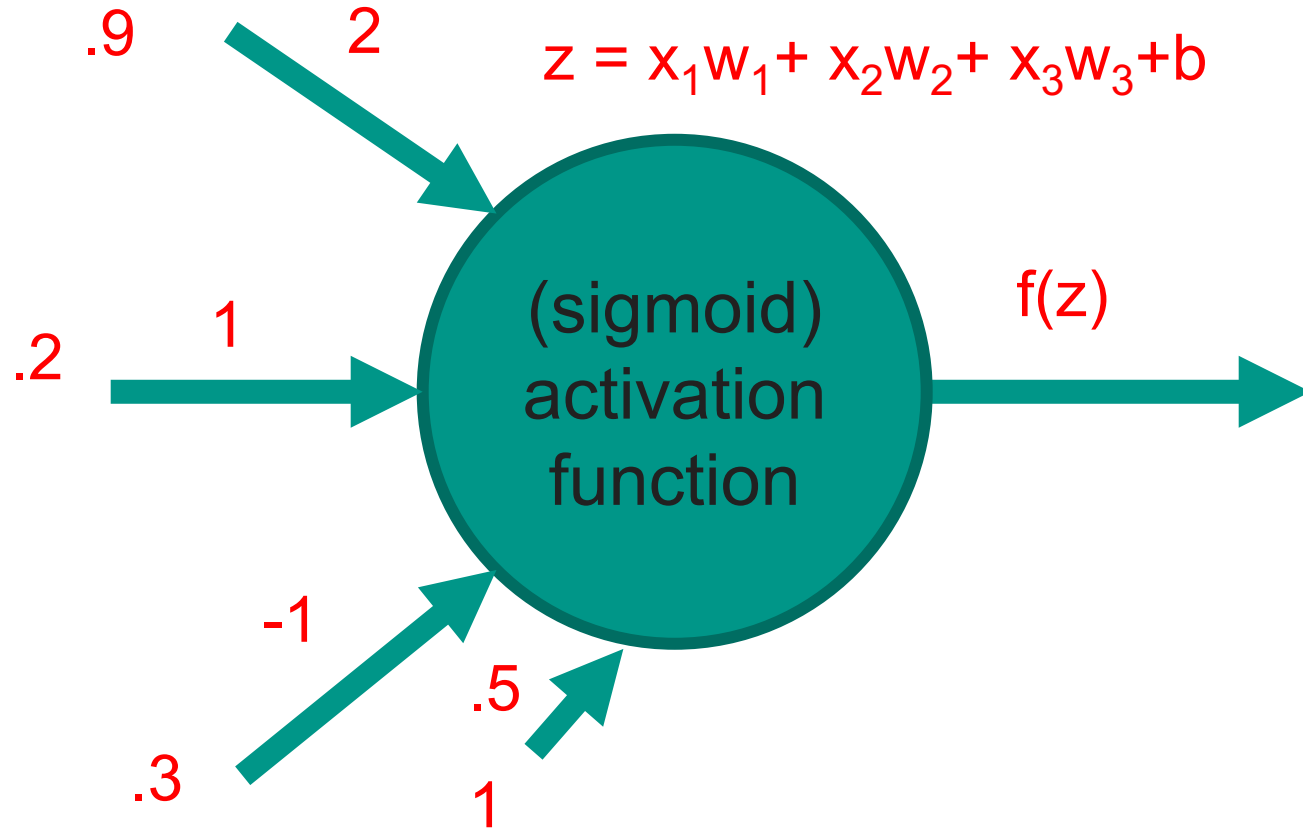
$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \quad \text{This will be helpful!}$$



# Example Neuron Computation

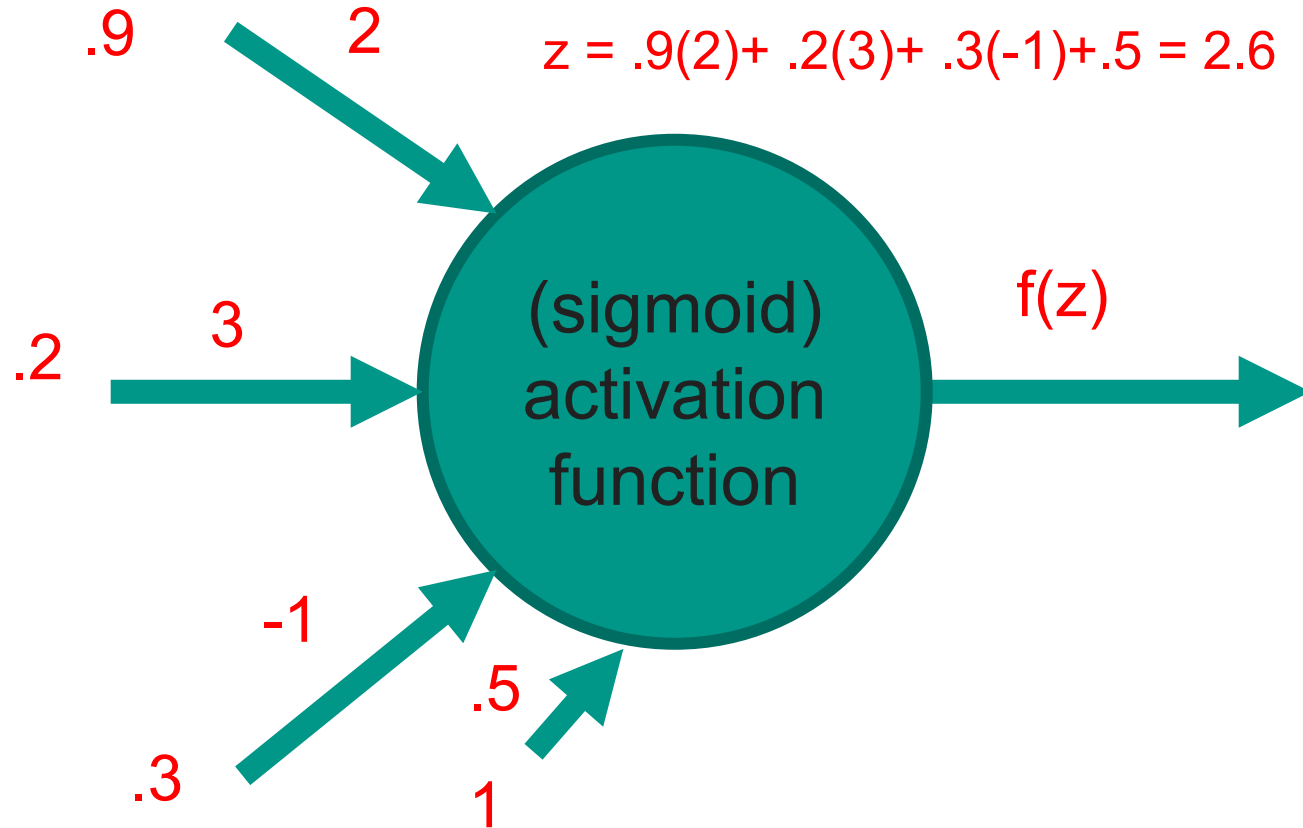


# Example Neuron Computation

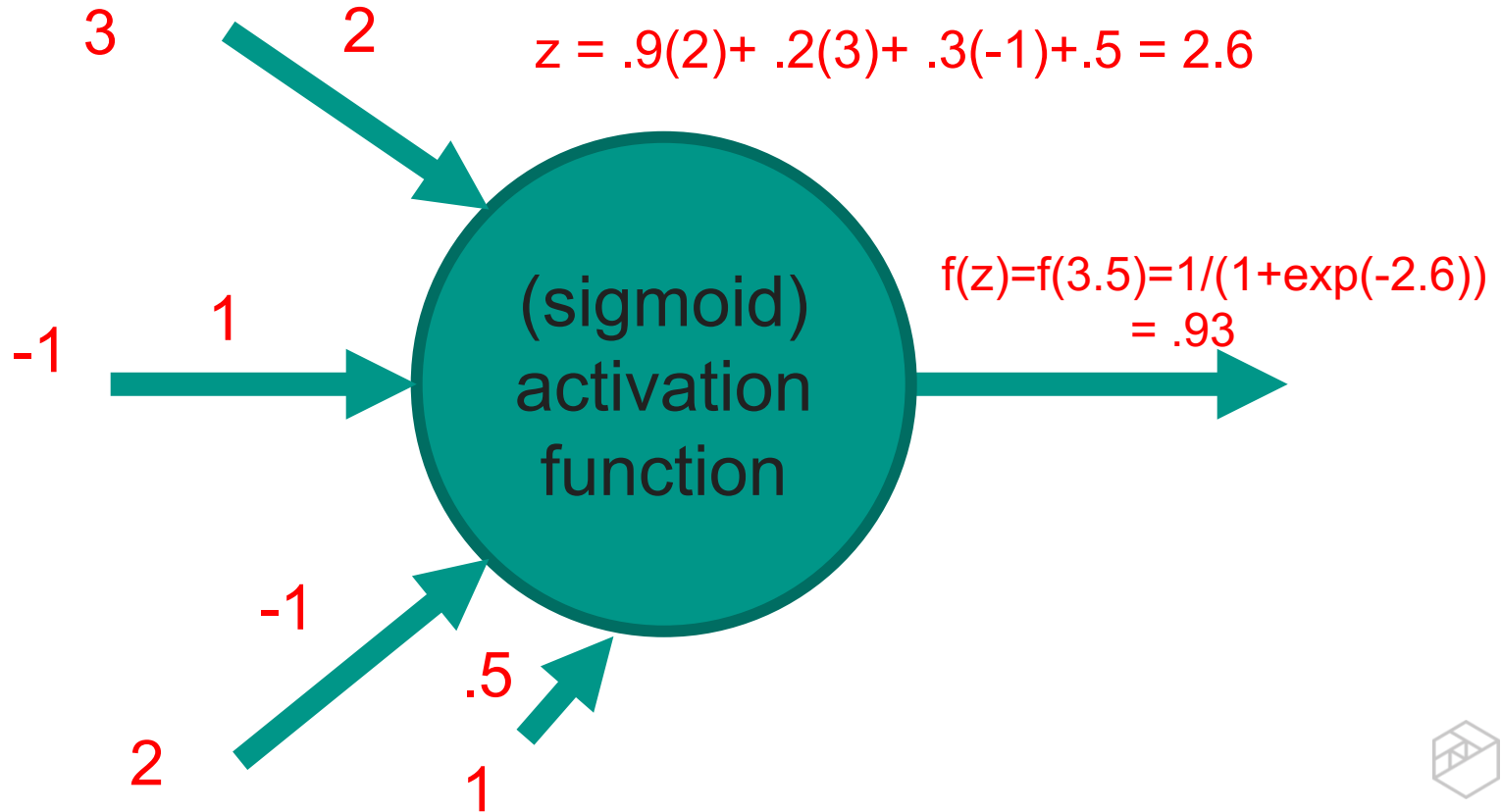




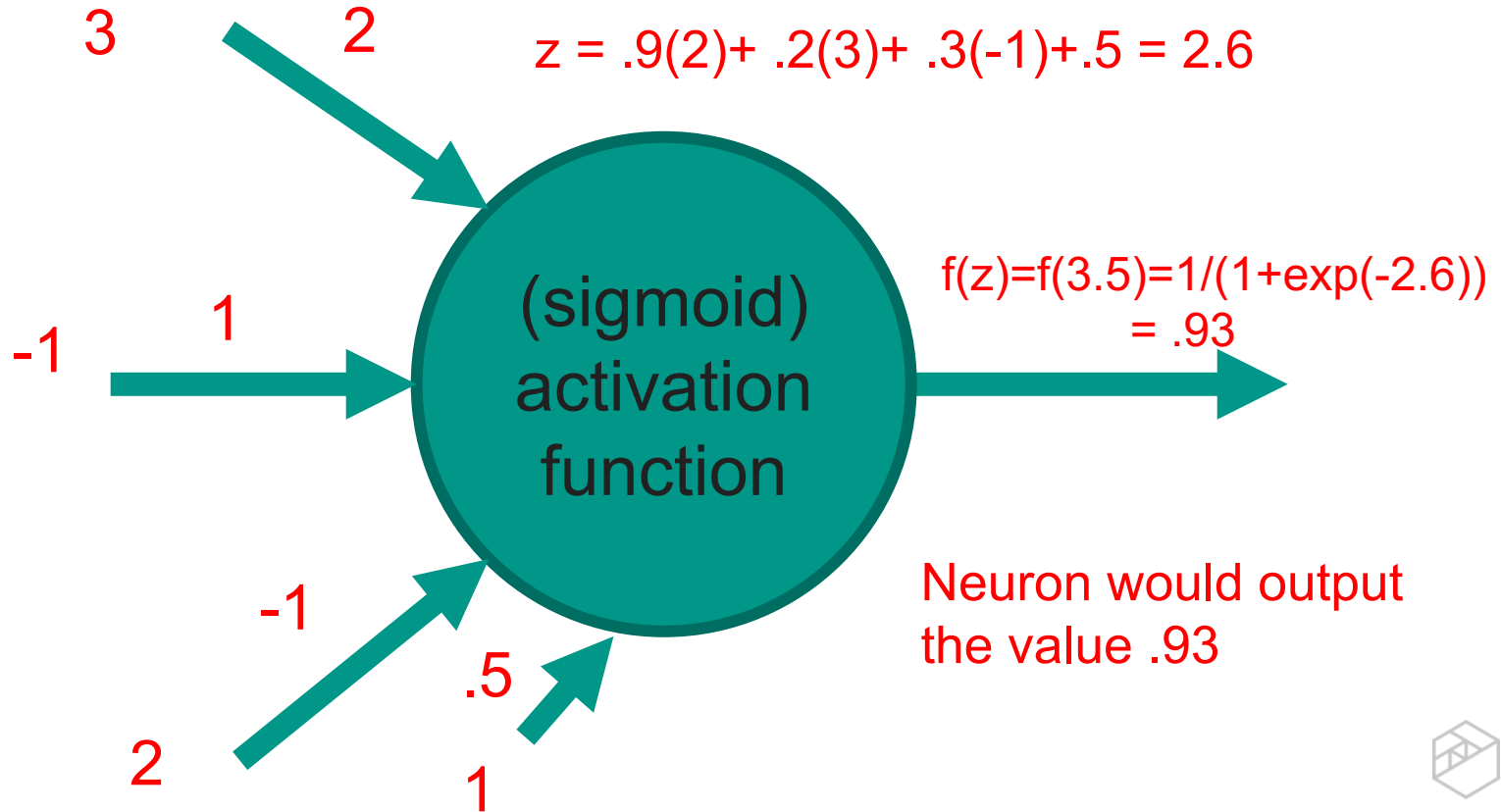
# Example Neuron Computation



# Example Neuron Computation

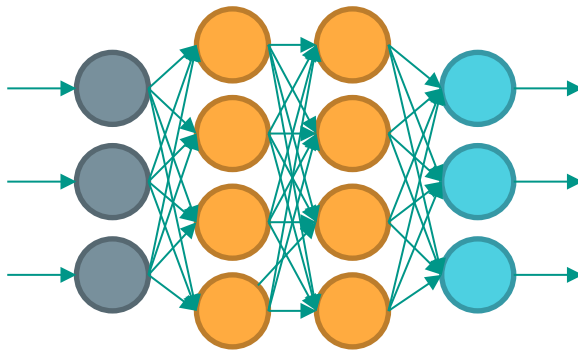


# Example Neuron Computation



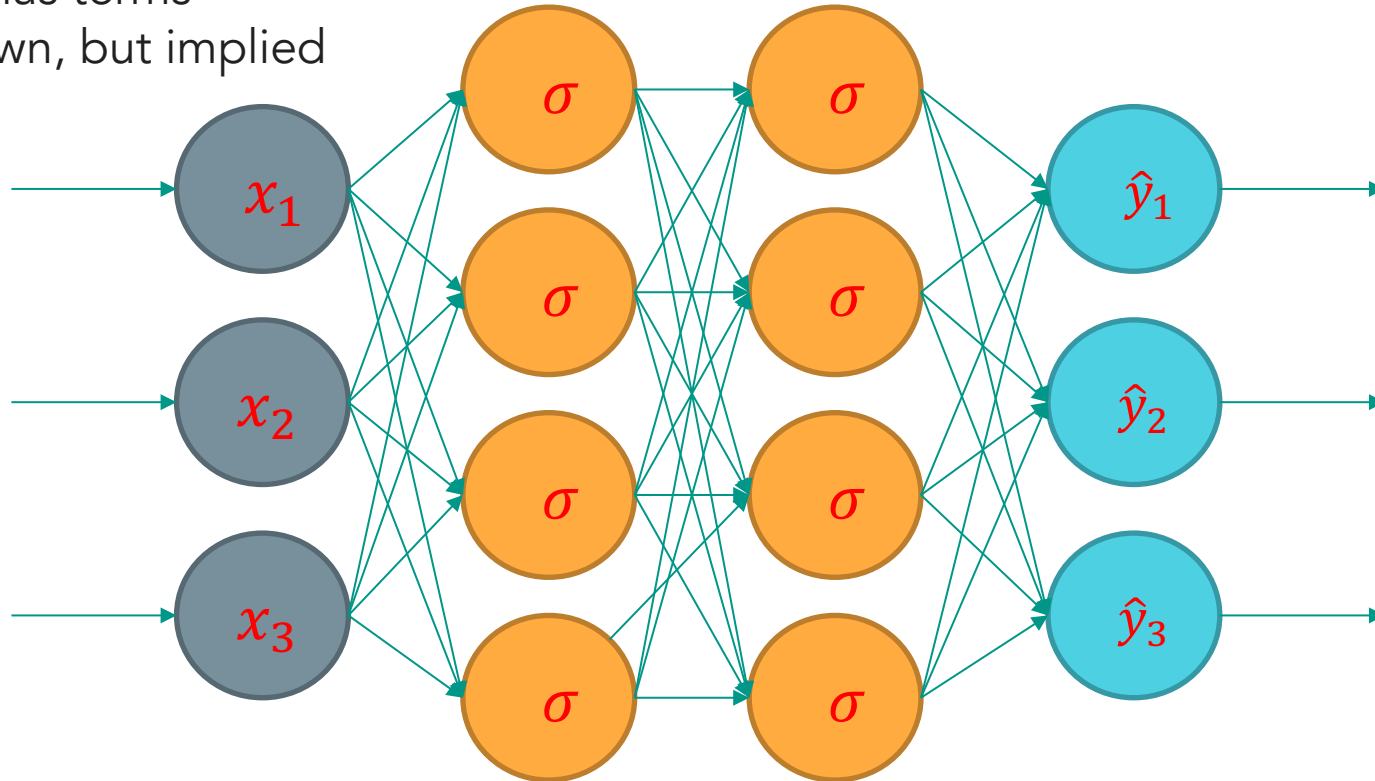
# Why Neural Nets?

- 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!

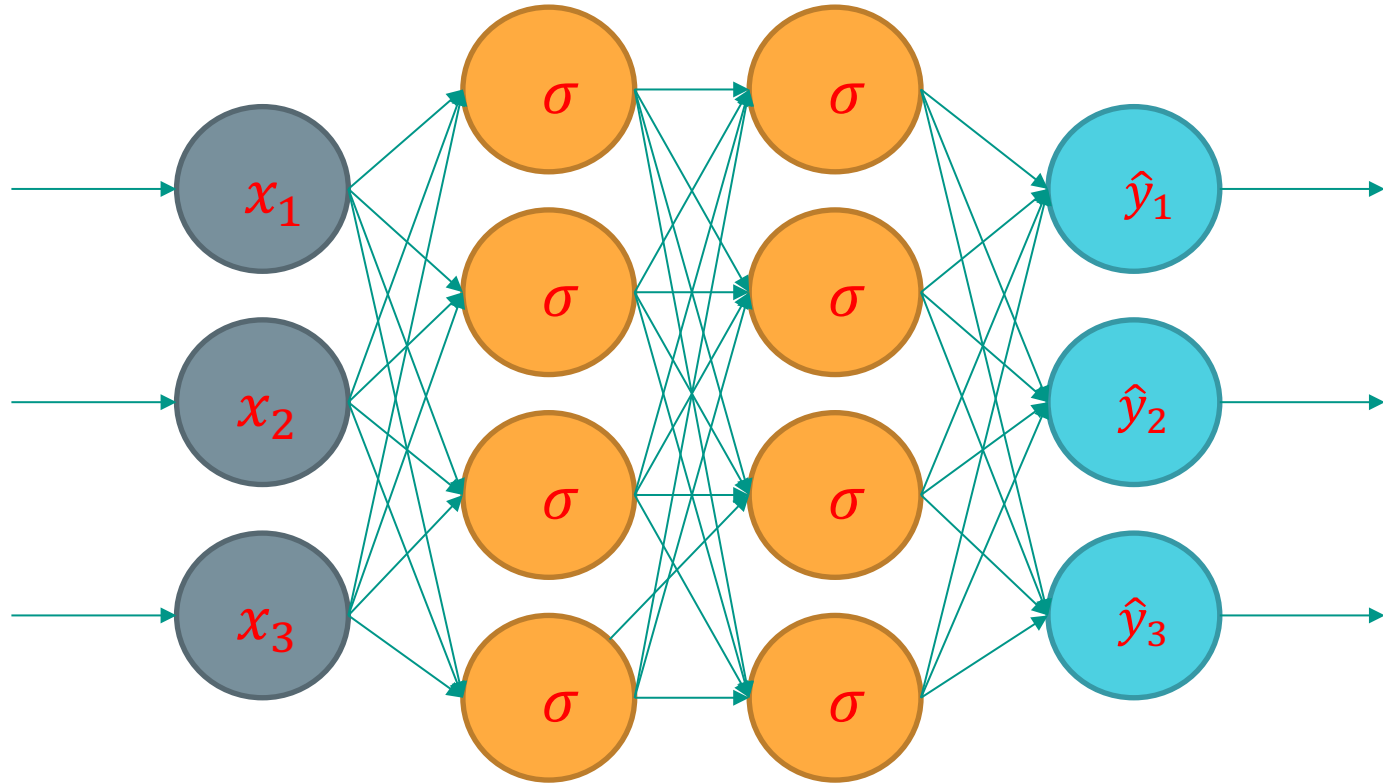


# Feedforward Neural Network

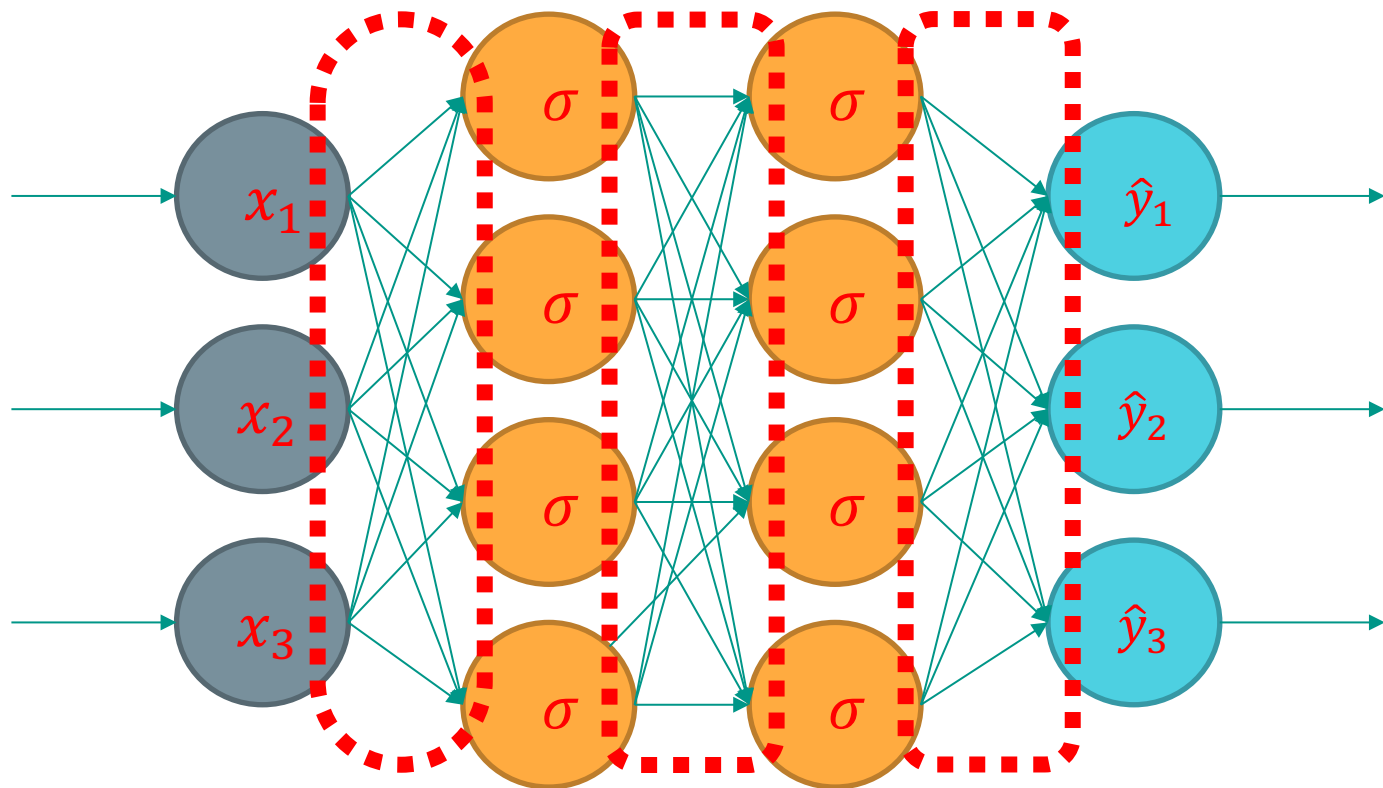
Note: bias terms  
not drawn, but implied



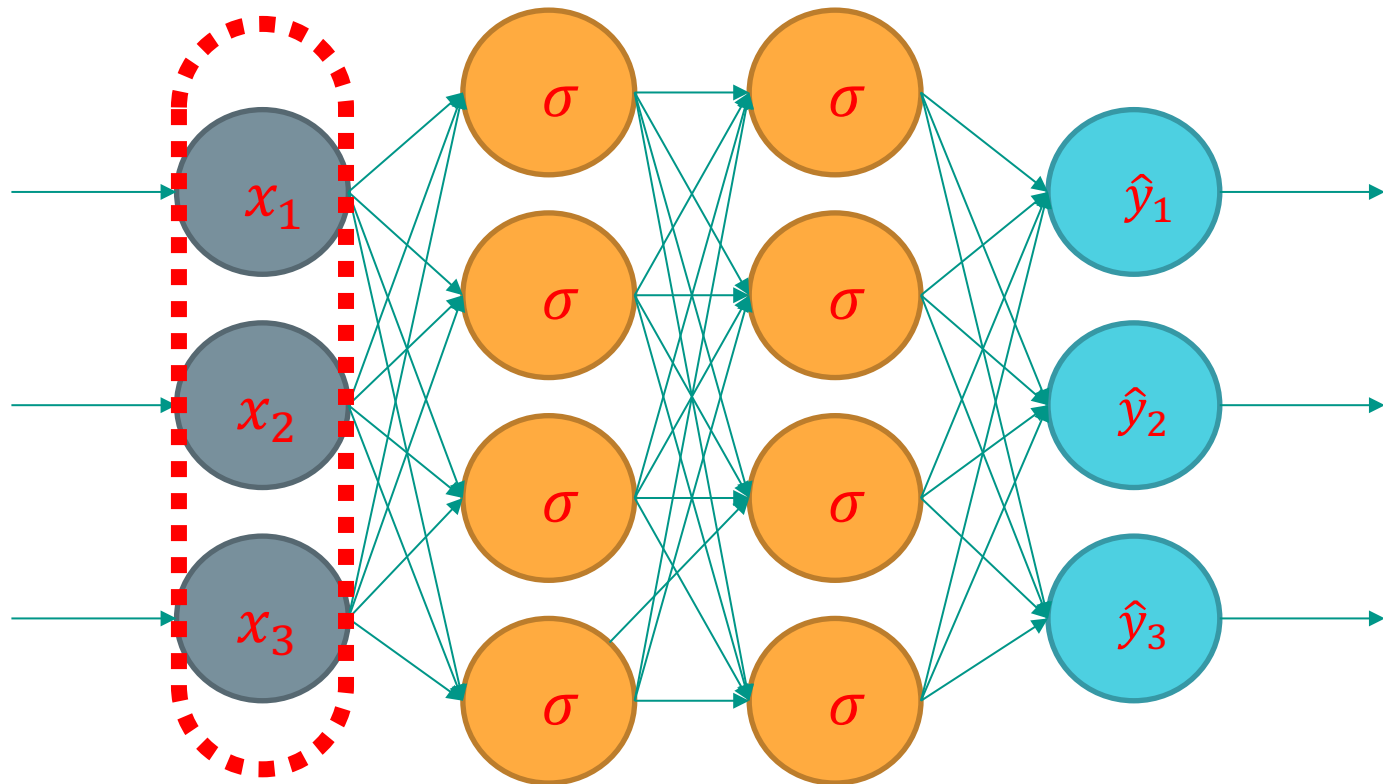
# Feedforward Neural Network



# Weights

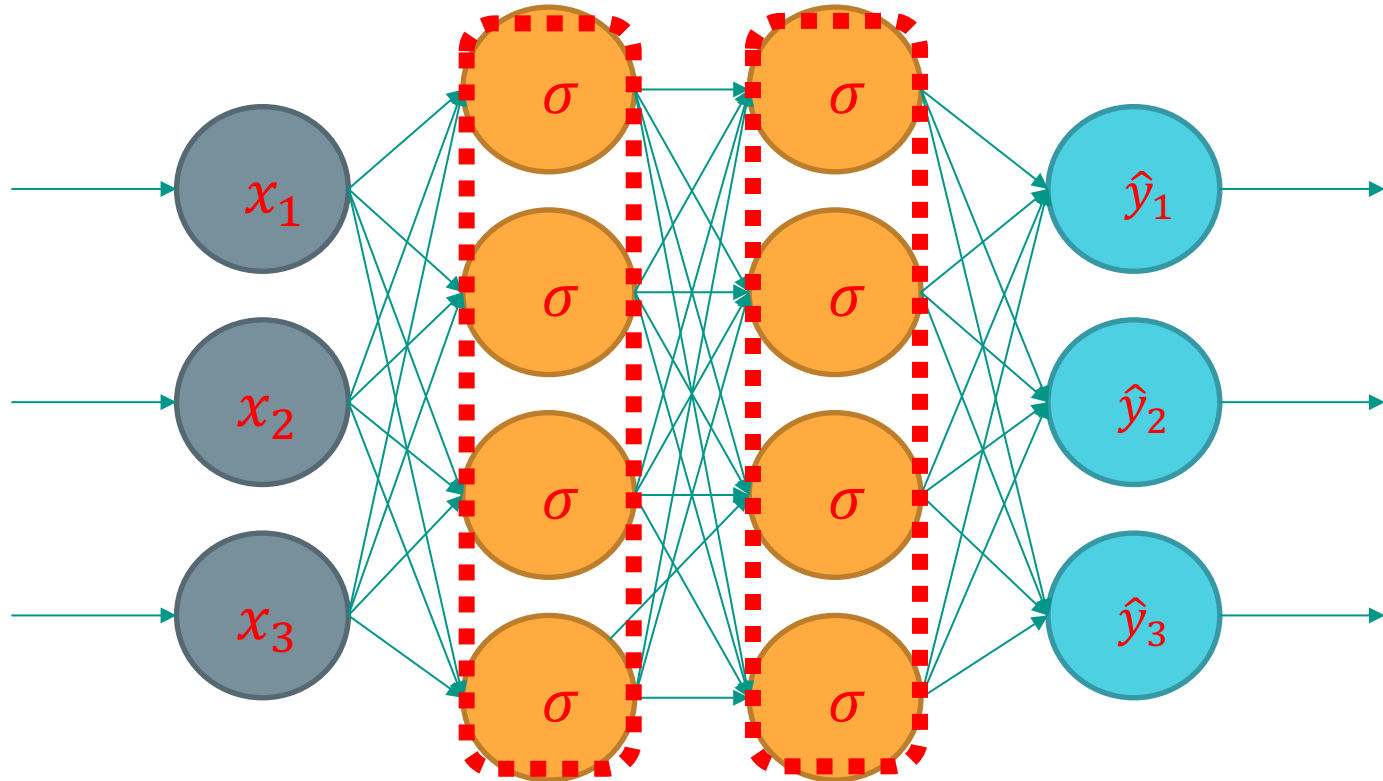


# Input Layer

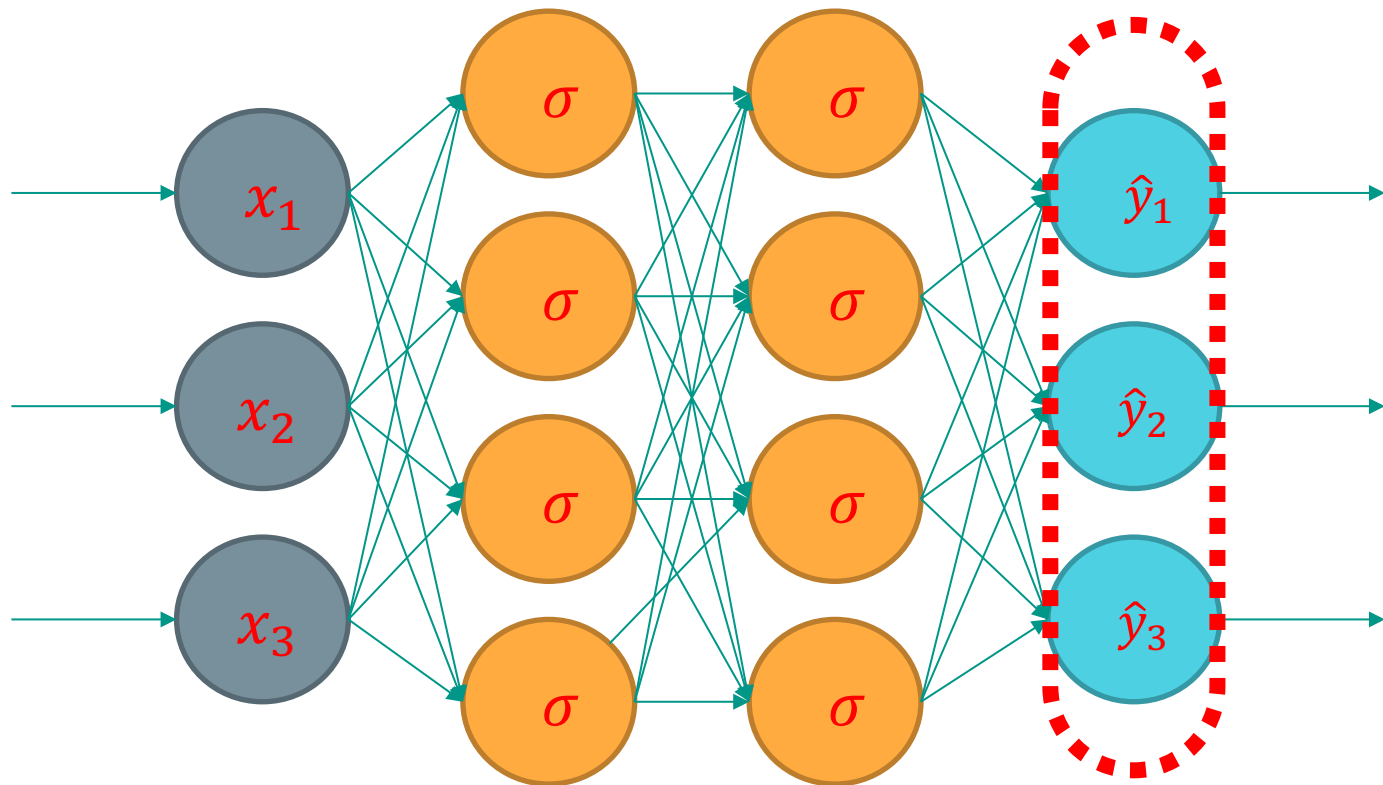




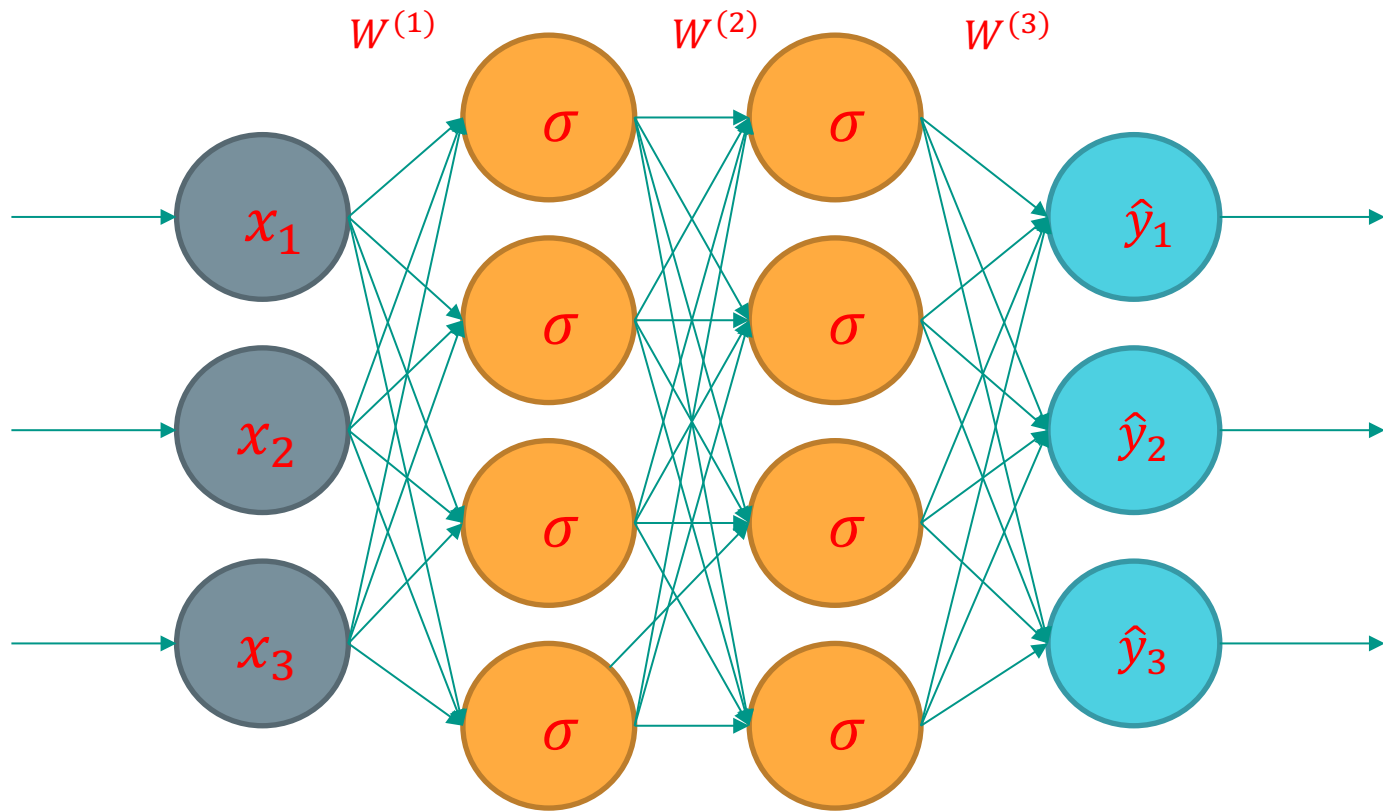
# Hidden Layers



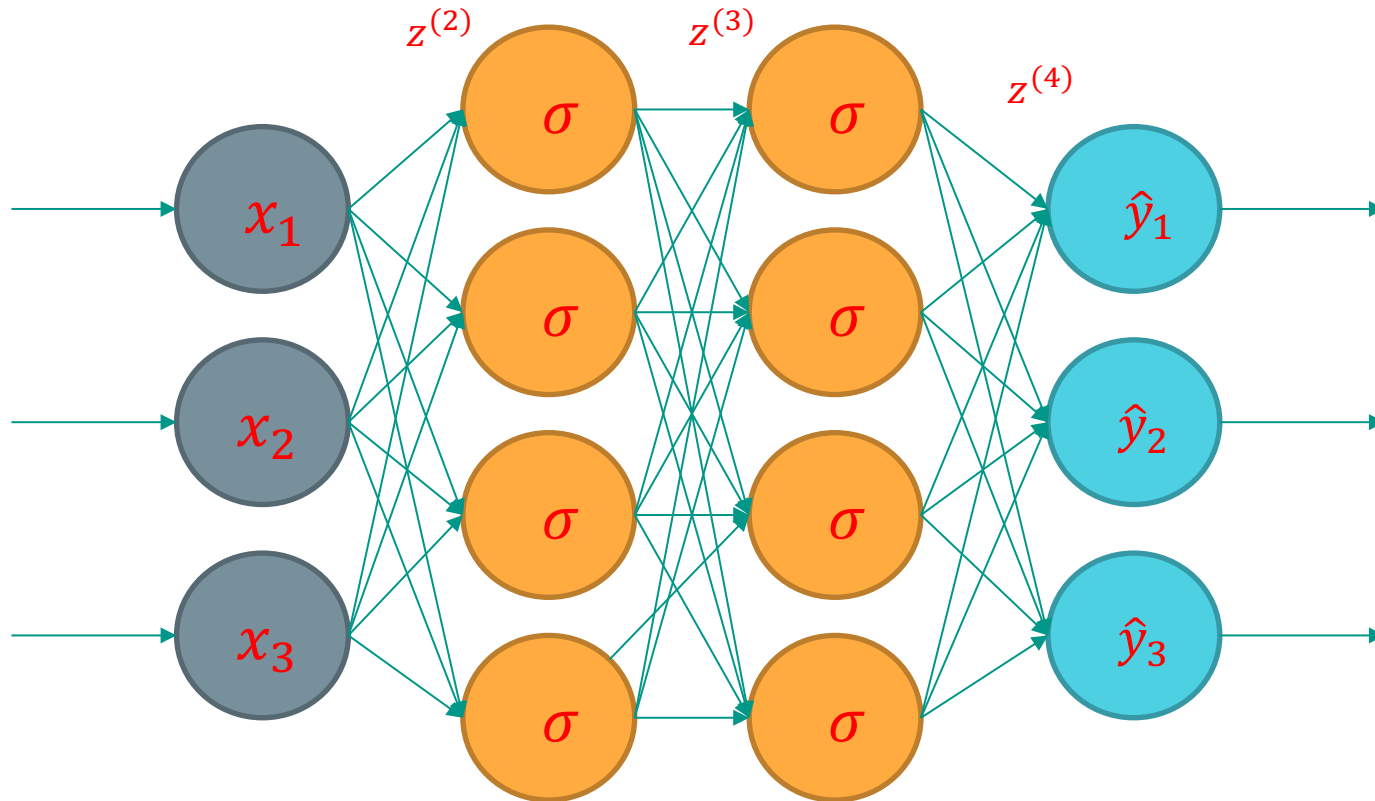
# Output Layer



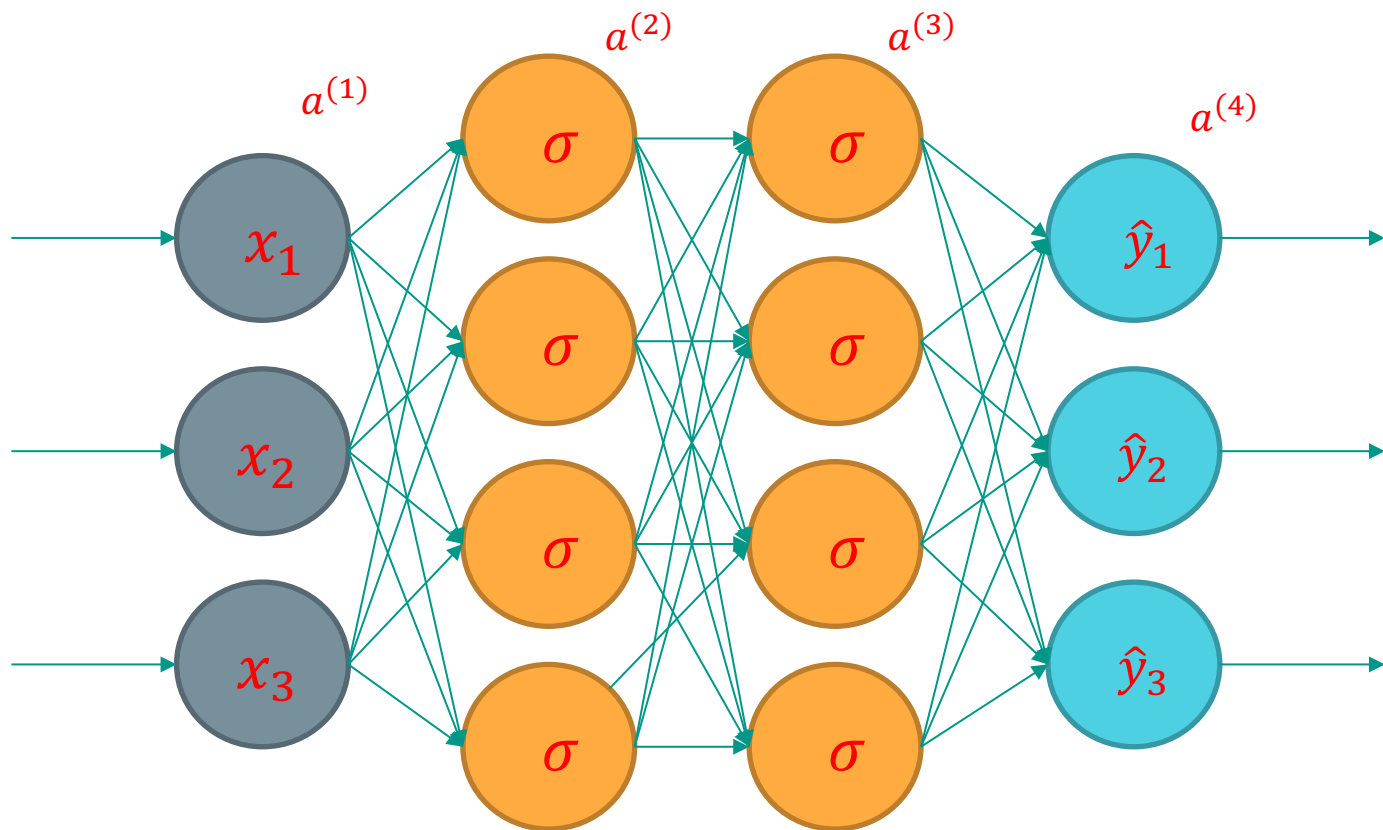
# Weights (represented by matrices)



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



# Activations (output of neurons to next layer)



# Matrix representation of computation

For a single data point (instance)

$$x = [x_1, x_2, x_3]$$

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

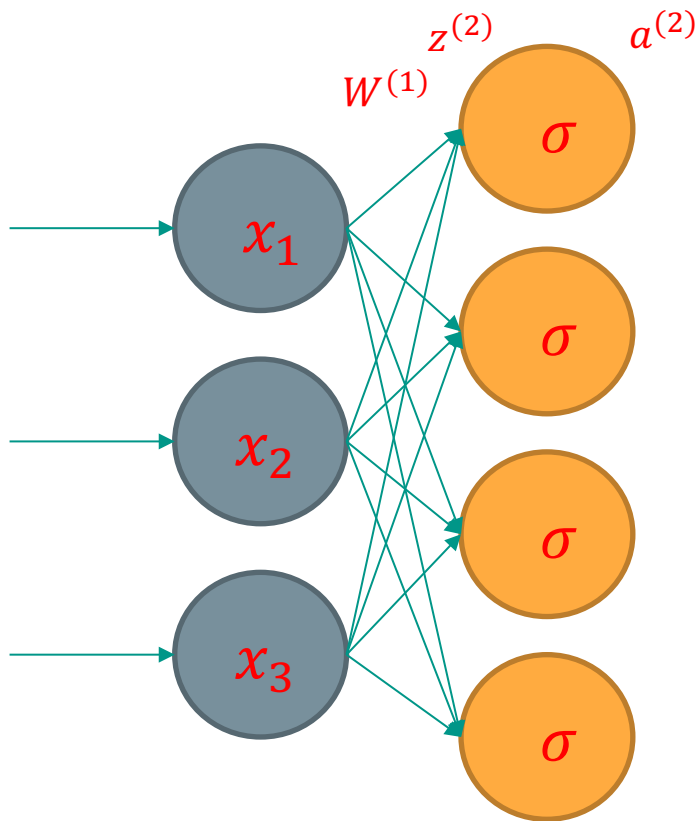
$$z^{(2)} = xW^{(1)}$$

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

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

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

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



# 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)

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

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

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

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

$$z^{(4)} = a^{(3)}W^{(3)}$$

$$\hat{y} = \textit{softmax}(z^{(4)})$$



# 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  $n \times 3$  matrix) (each row a single instance)

Output: vector  $\hat{y}$  (an  $n \times 3$  matrix) (each row a single prediction)

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

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

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

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

$$z^{(4)} = a^{(3)}W^{(3)}$$

$$\hat{y} = \textit{softmax}(z^{(4)})$$





Now we know how feedforward NNs do Computations.

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





**METIS**