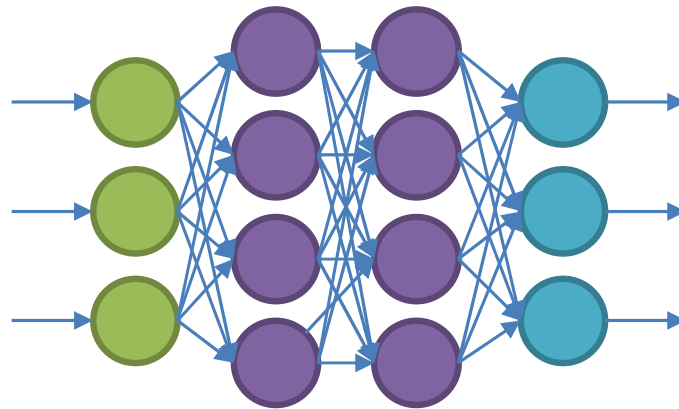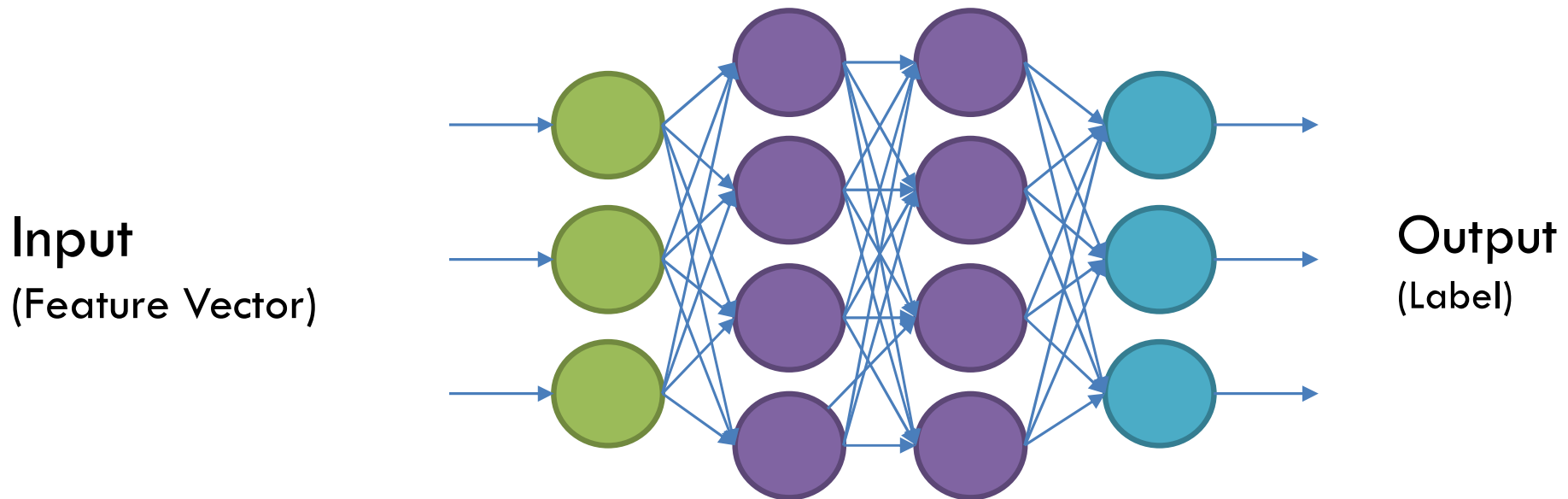# Neural Networks

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

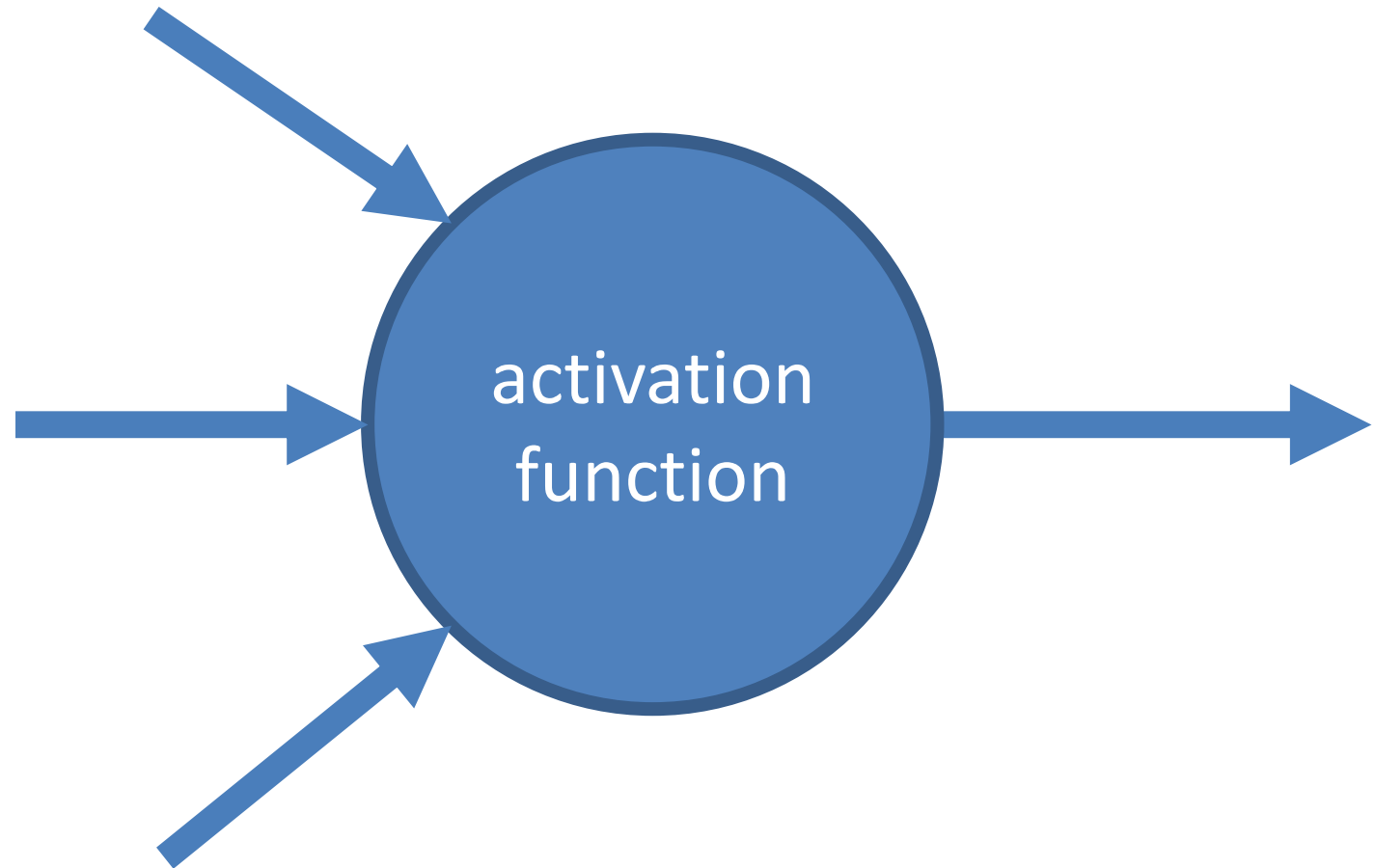- By layering many neurons, can create complex model

Neural Net Structure

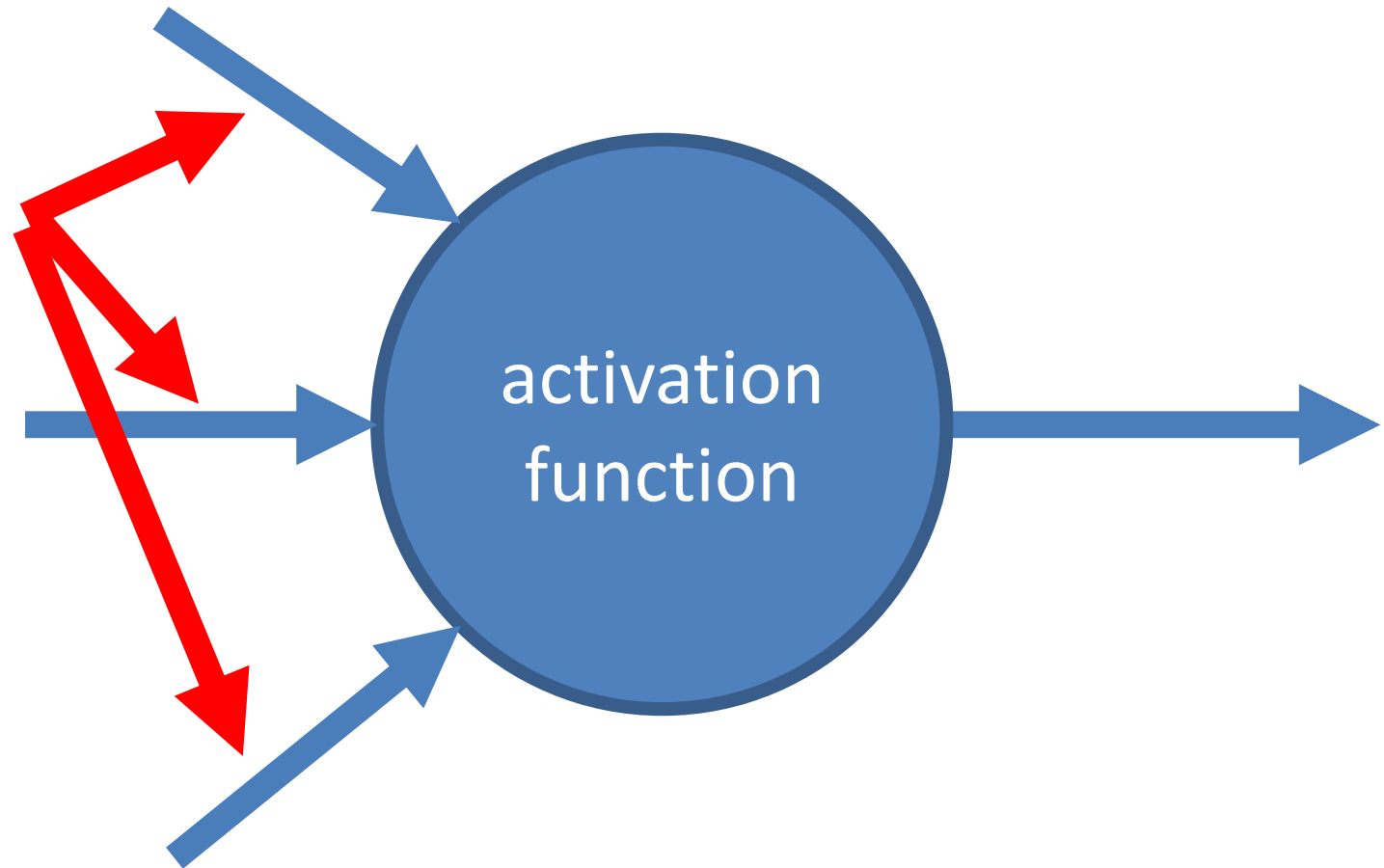

Input
(Feature Vector)

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
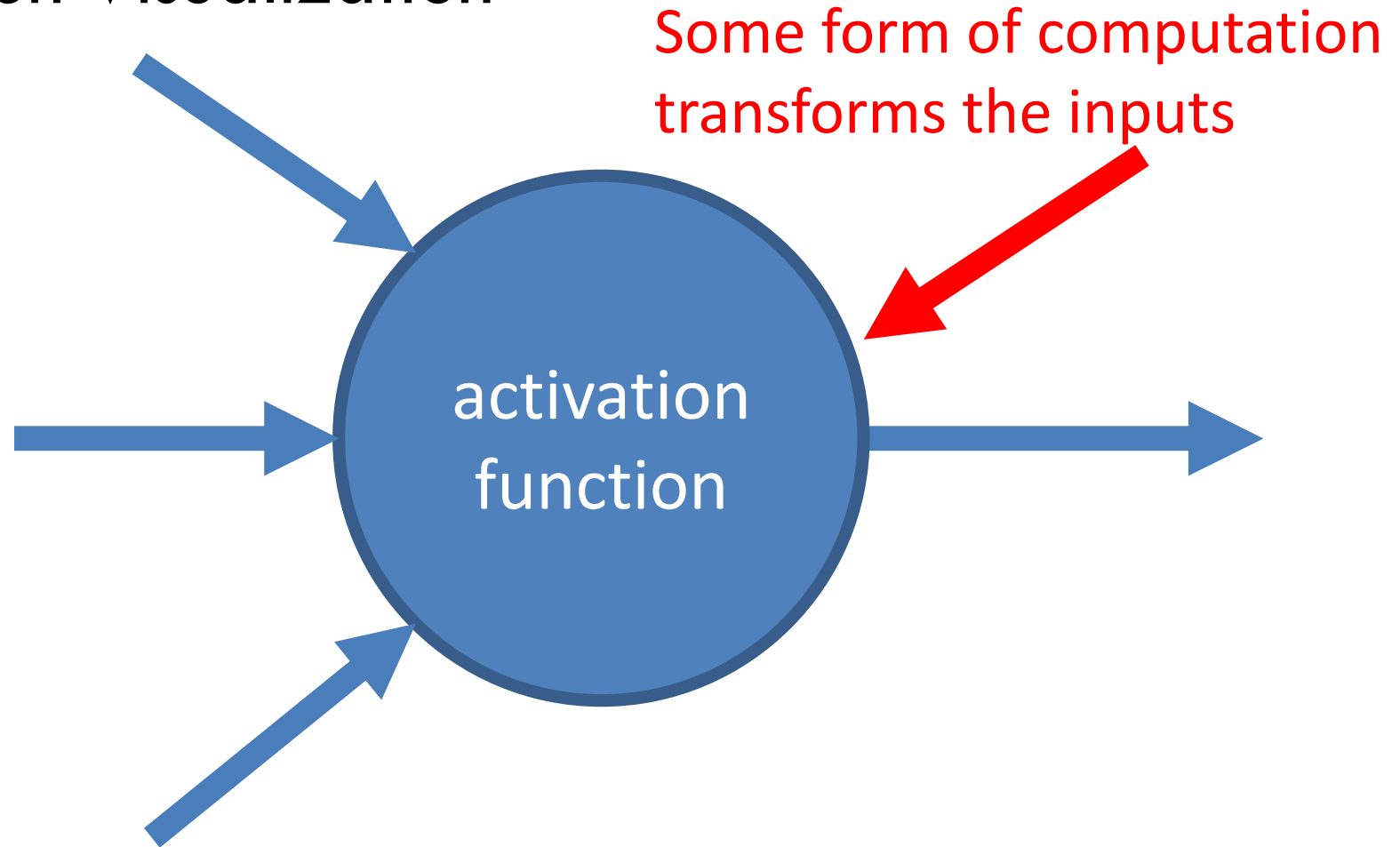
Basic Neuron Visualization

Basic Neuron Visualization

Data from
previous
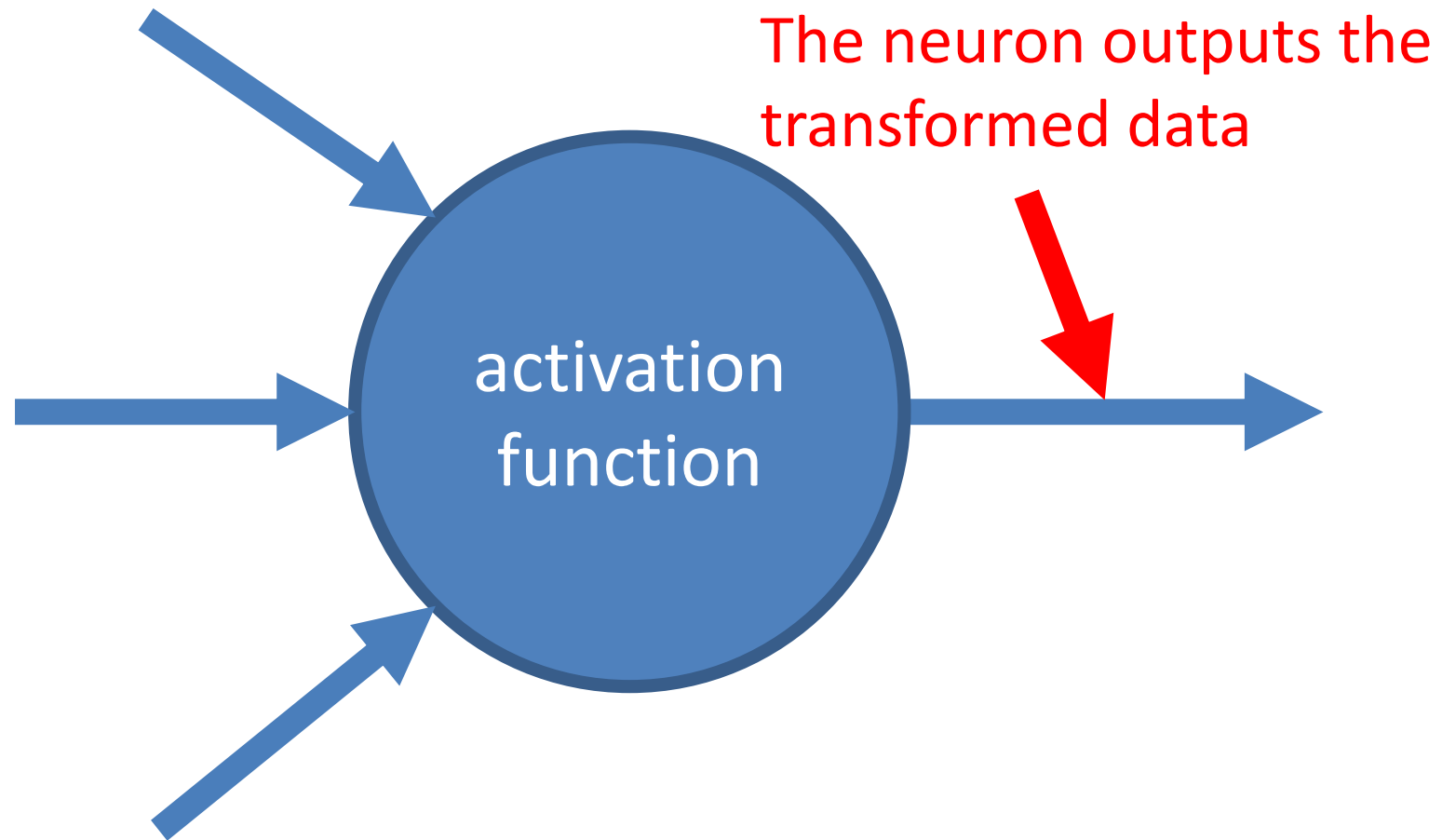layer

activation
function

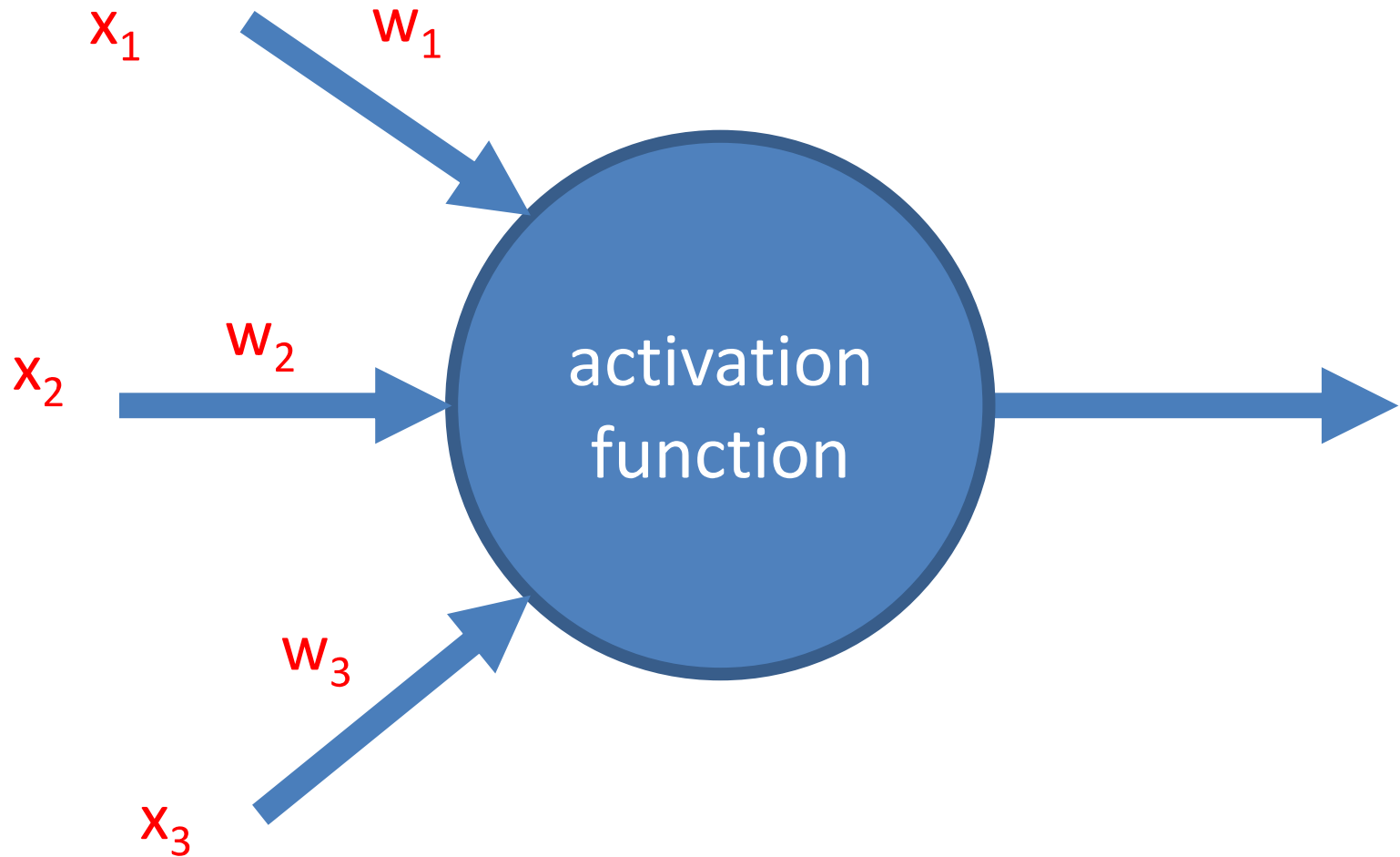# Basic Neuron Visualization

Some form of computation transforms the inputs

activation function

Basic Neuron Visualization

activation function

The neuron outputs the transformed data

Basic Neuron Visualization

Basic Neuron Visualization

Basic Neuron Visualization

$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$

$f(z)$

activation function

x_1 w_1

x_2 w_2

x_3 w_3 b 1
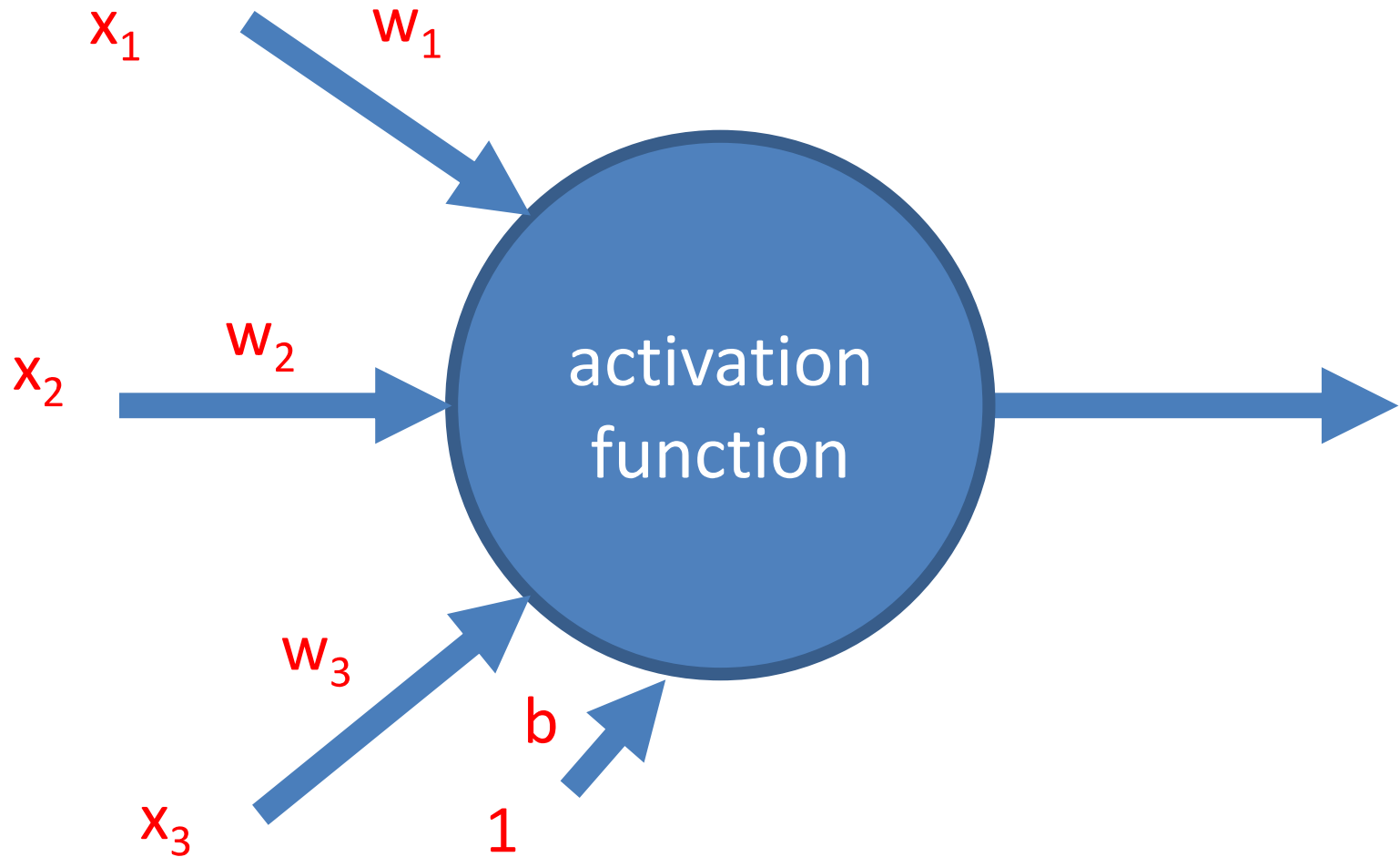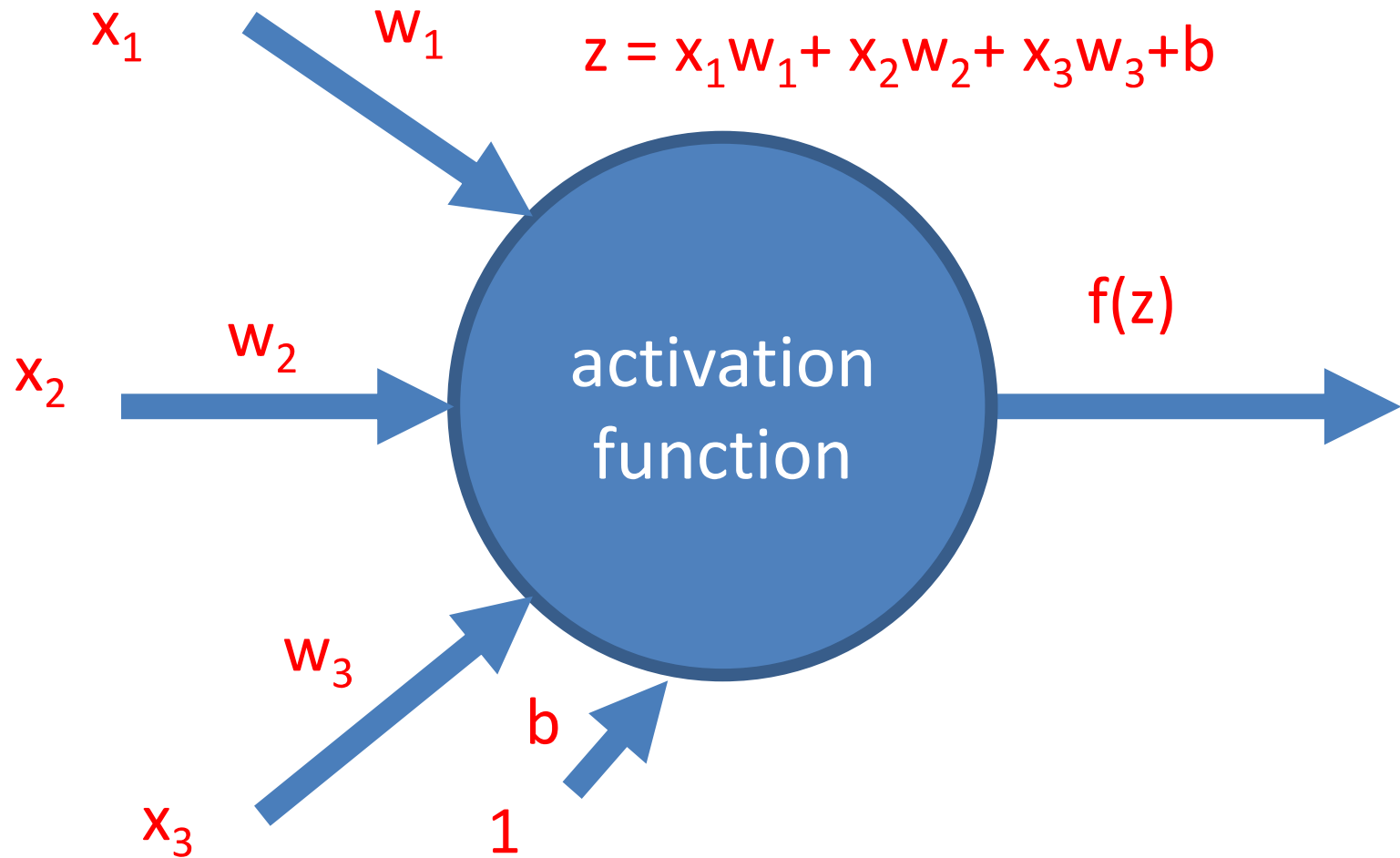
In Vector Notation

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

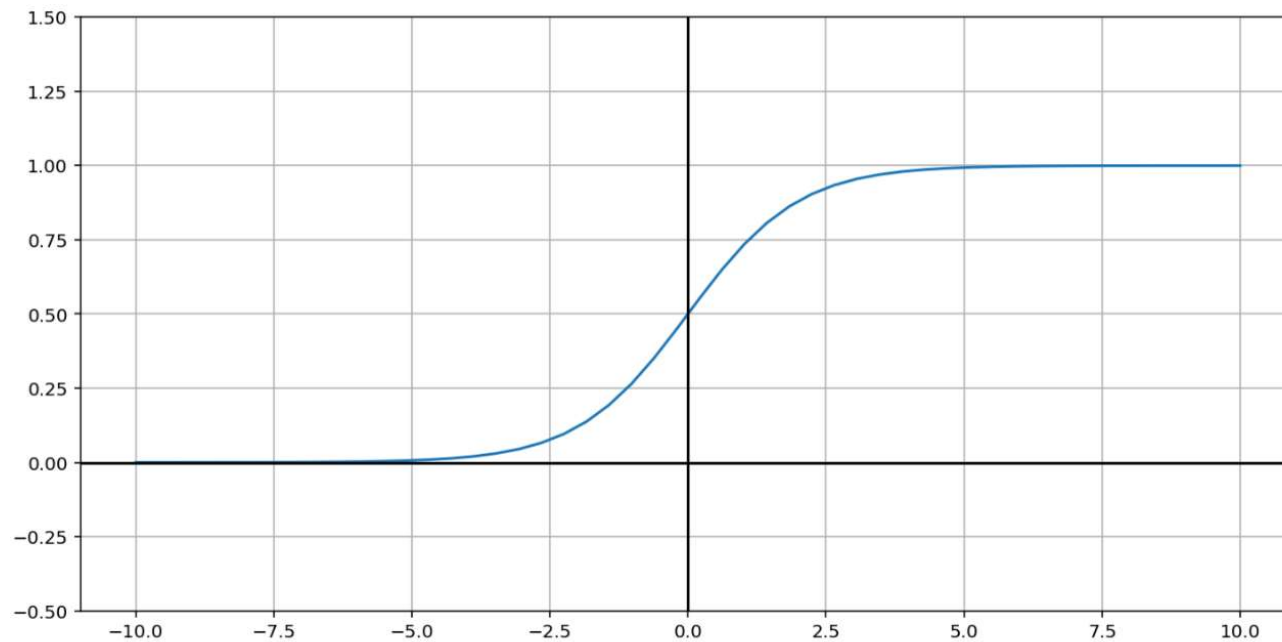$$z = b + \sum_{i=1}^{m} x_i w_i = x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b$$

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

Nice Property of Sigmoid Function

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

Quotient rule

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

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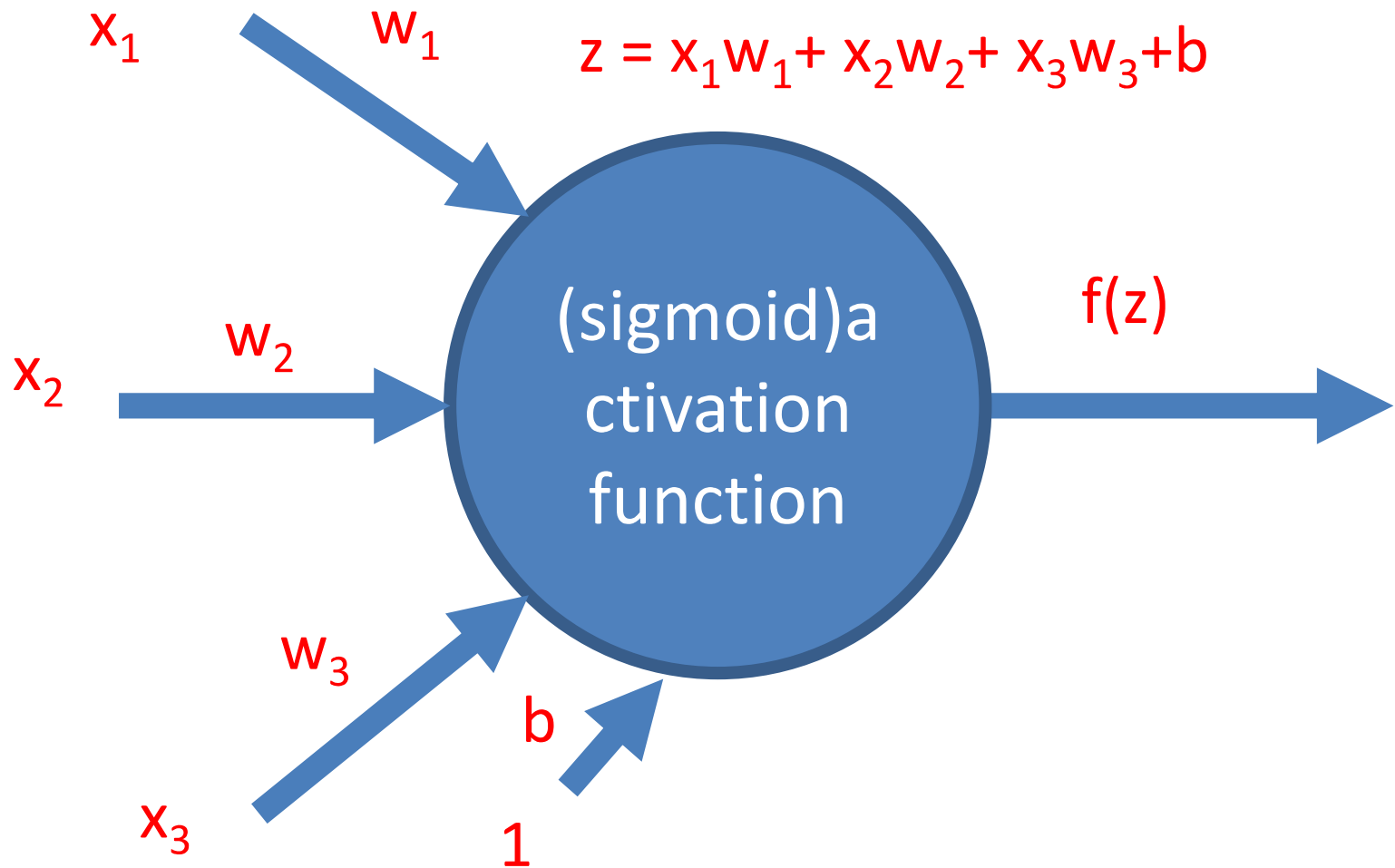
$$= \frac{1 + e^{-z} - 1}{(1 + e^{-z})^2} = \frac{1 + e^{-z}}{(1 + e^{-z})^2} - \frac{1}{(1 + e^{-z})^2}$$

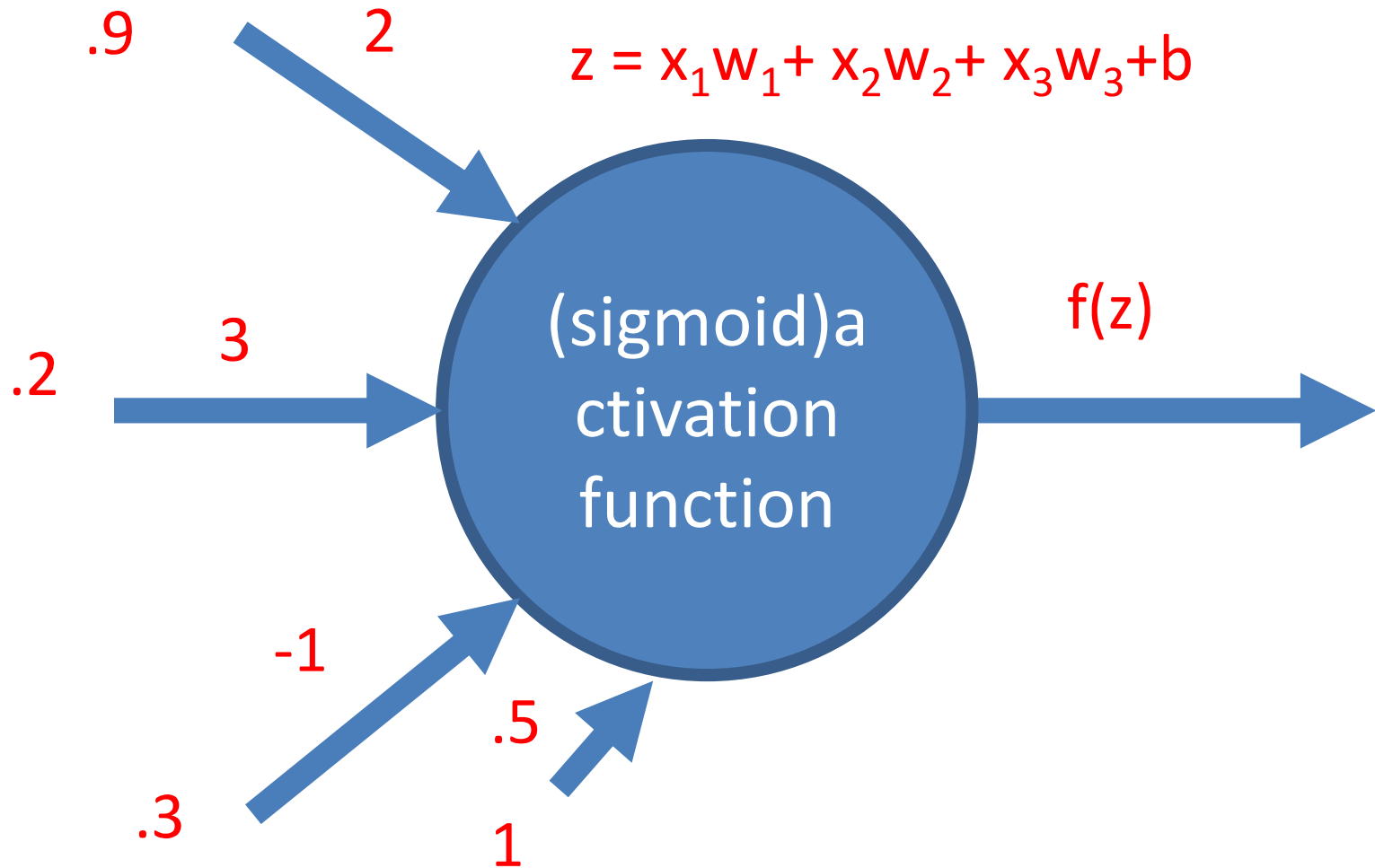$$= \frac{1}{1 + e^{-z}} - \frac{1}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}}\left(1 - \frac{1}{1 + e^{-z}}\right)$$

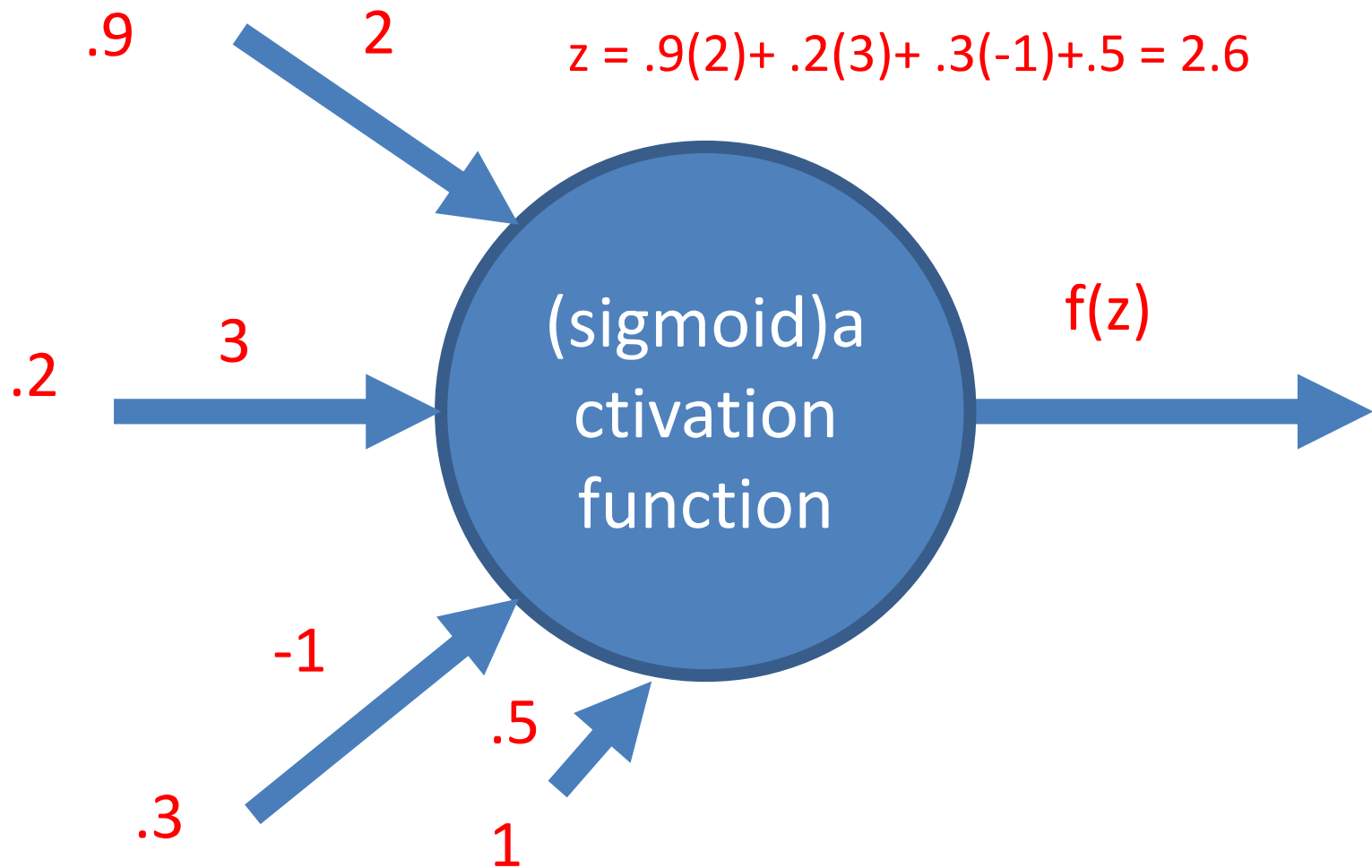$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

This will be helpful!

Example Neuron Computation

x_1 w_1
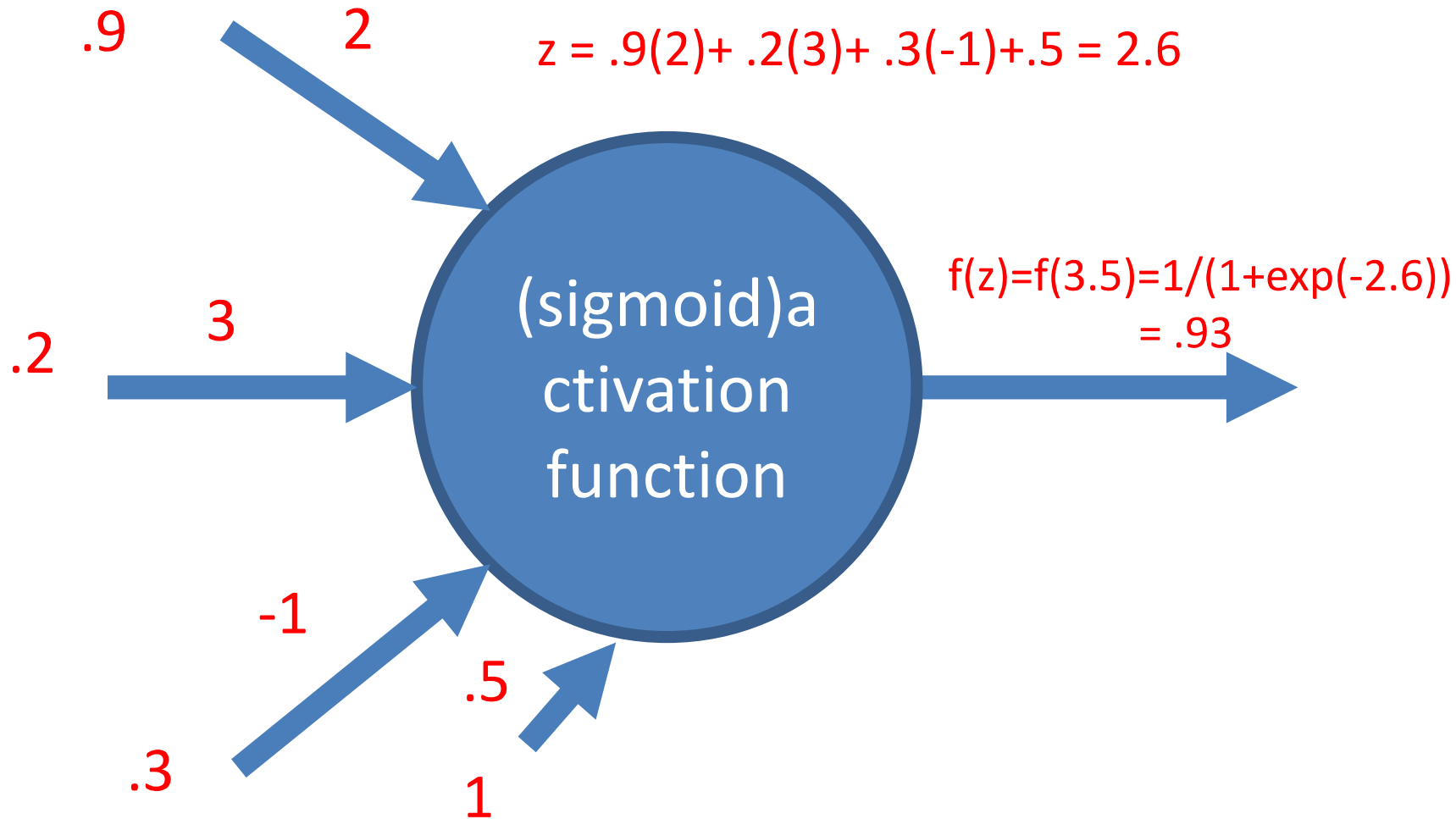
$z = x_1w_1 + x_2w_2 + x_3w_3 + b$

w_2

x_2

(sigmoid)activation function

$f(z)$

w_3

b

x_3

1

Example Neuron Computation

.9   2

$z = x_1w_1 + x_2w_2 + x_3w_3 + b$

.2   3

(sigmoid)activation function

f(z)

-1

.5

.3   1

Example Neuron Computation

.9

2

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

(sigmoid)activation function

f(z)

3

.2

-1

.5

.3

1

Example Neuron Computation

.9

2

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

.2

3

$$f(z) = f(3.5) = 1/(1 + \exp(-2.6)) = .93$$

(sigmoid)activation function

-1

.5

.3

1

Example Neuron Computation

.9

2

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

(sigmoid)activation function

$f(z) = f(3.5) = 1/(1 + \exp(-2.6)) = .93$

3

.2

-1

.5

.3

1

Neuron would output the value .93
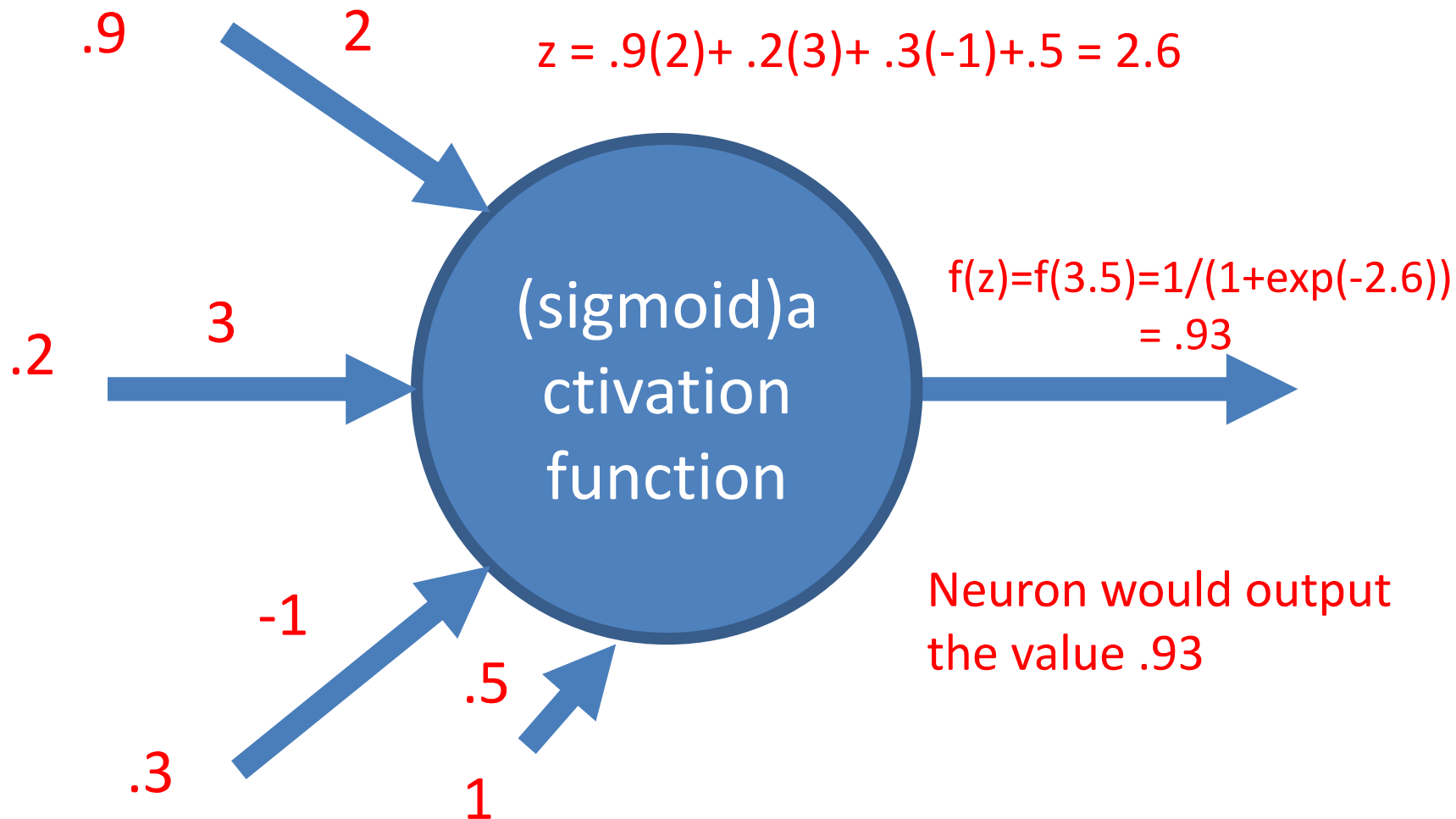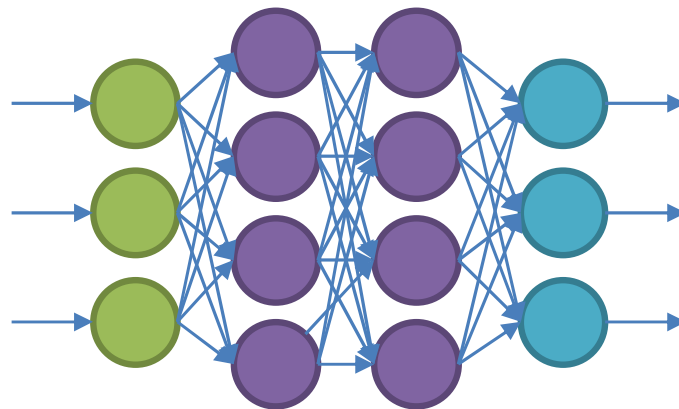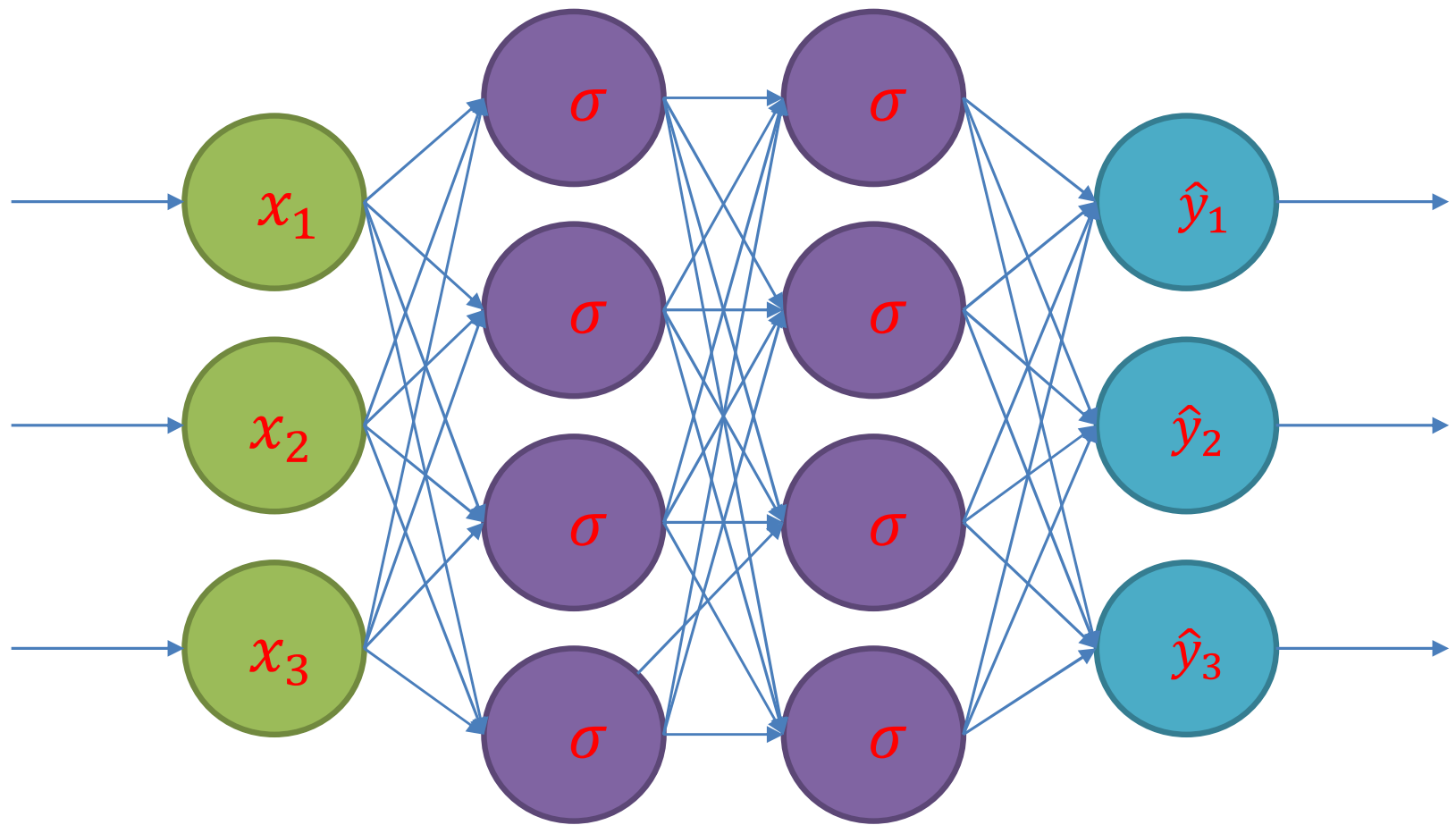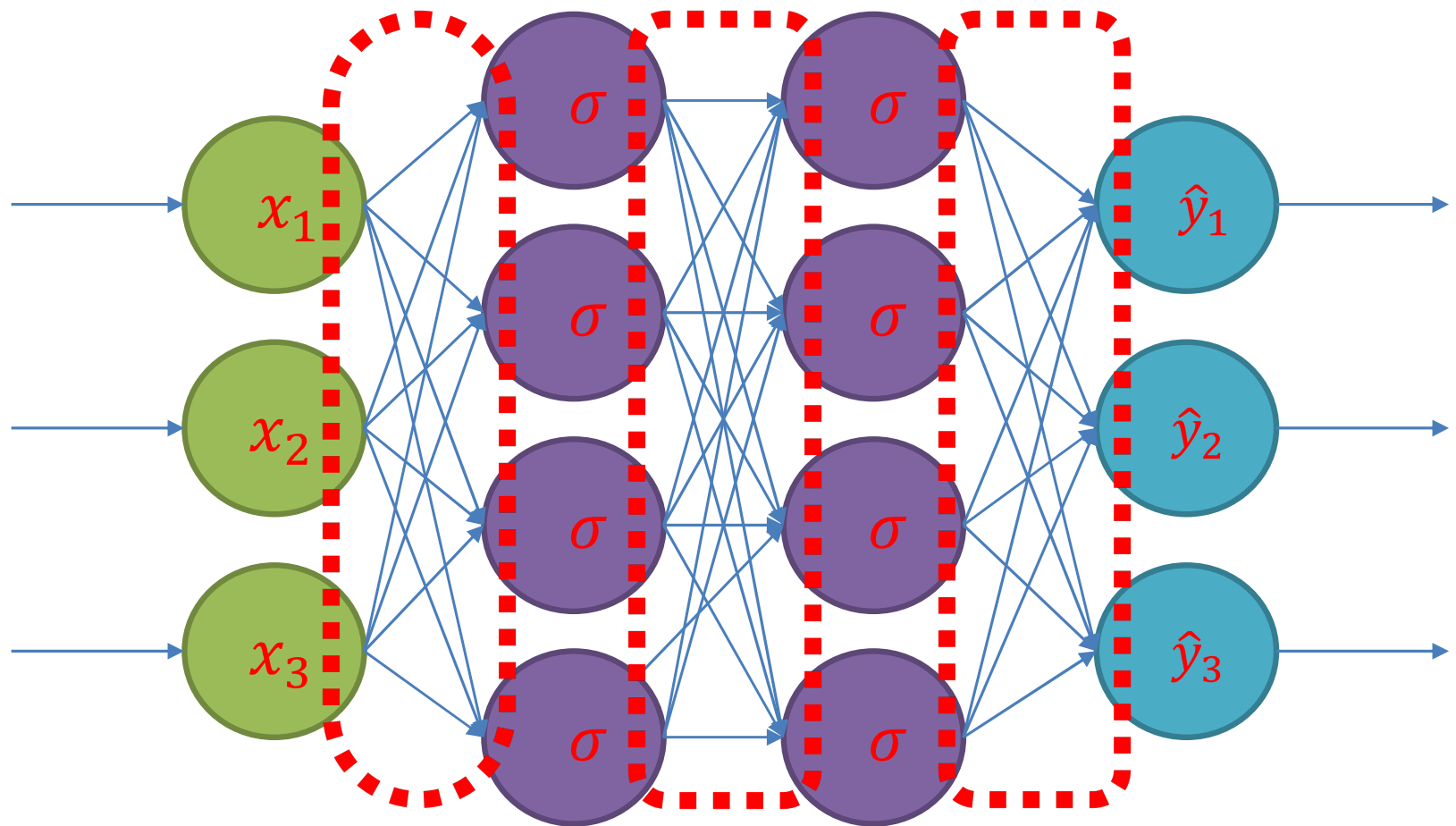
Why Neural Nets?

- 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.
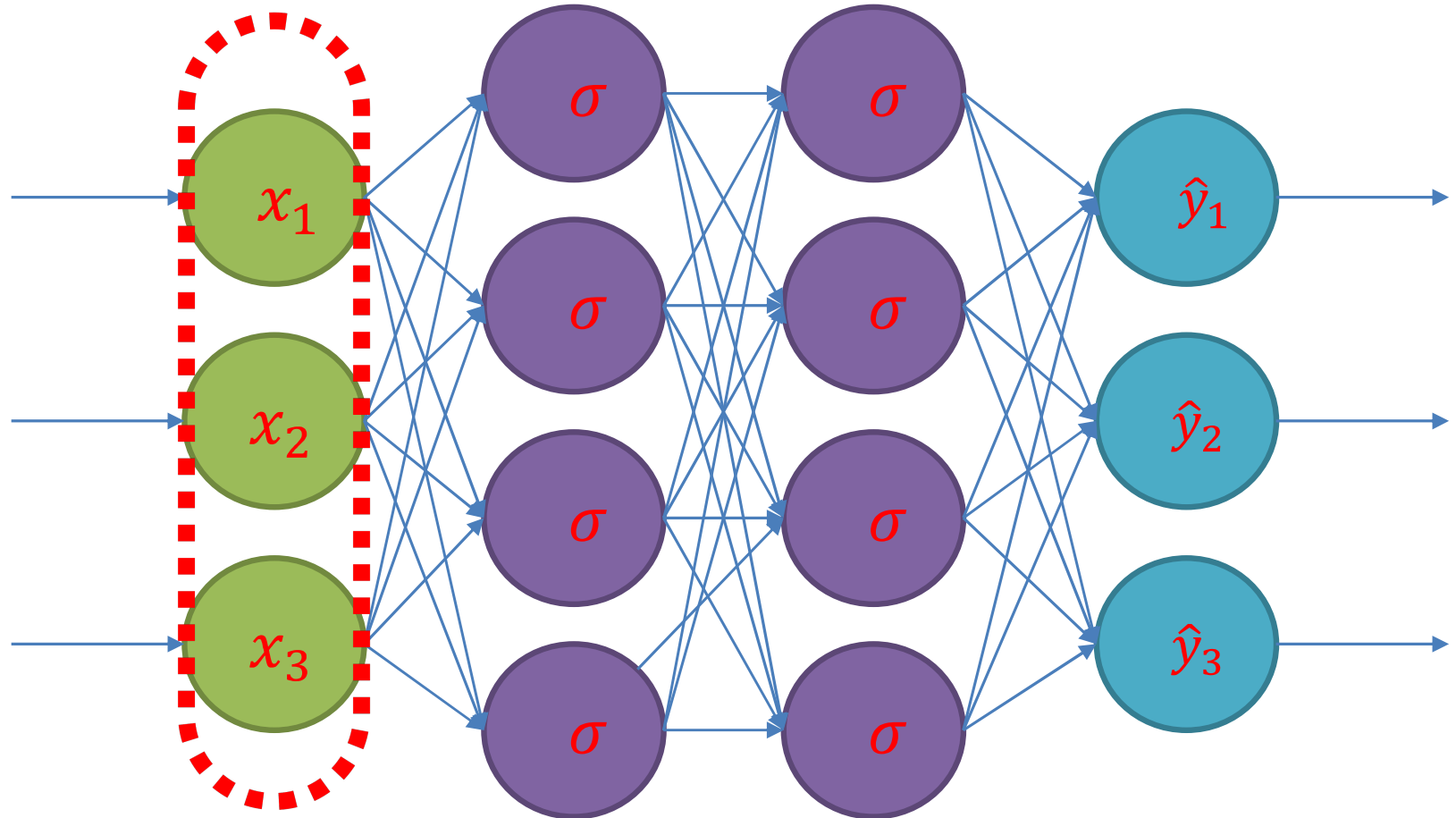- Most real-world problems are considerably more complicated!
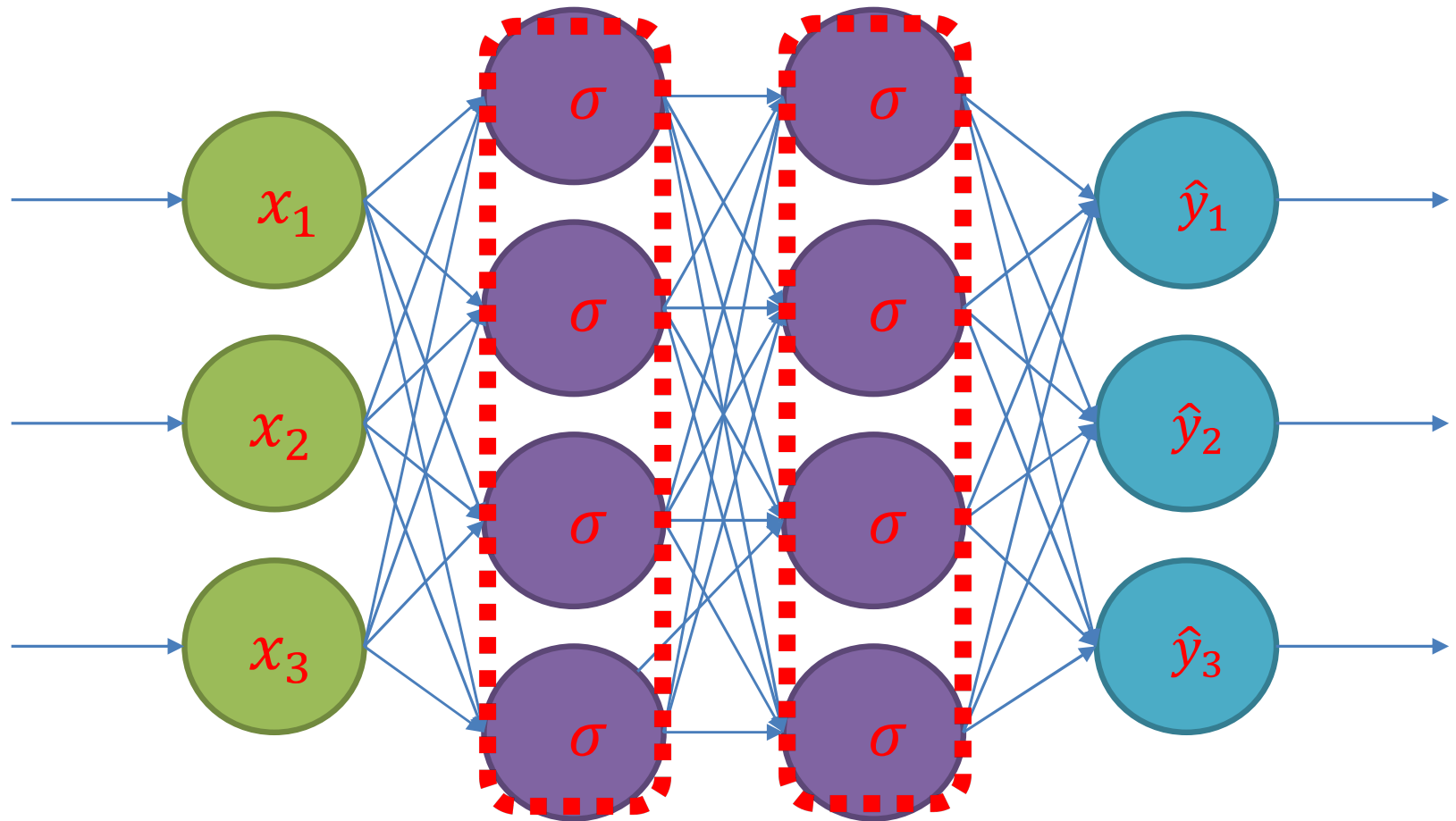
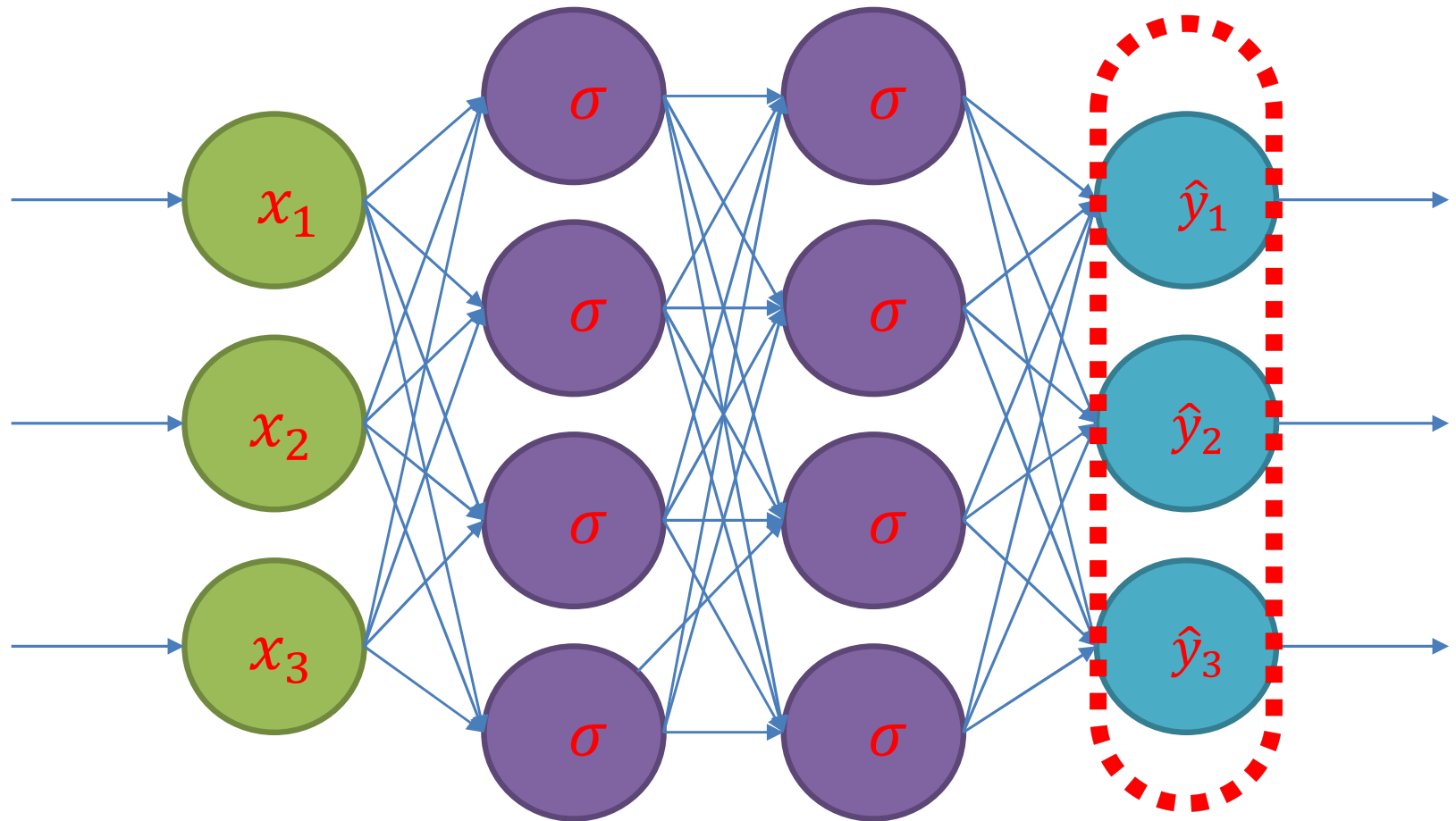# Feedforward Neural Network
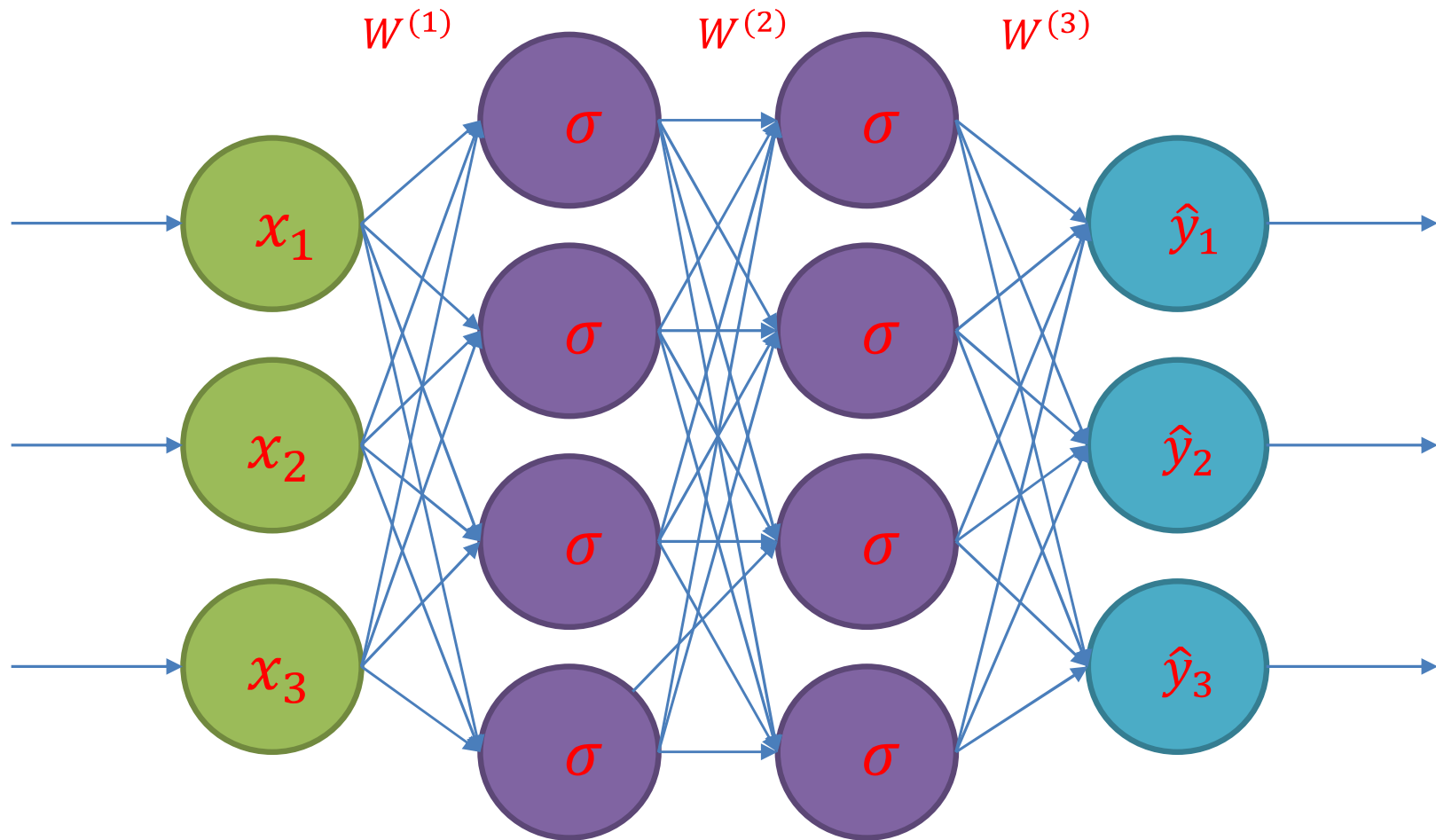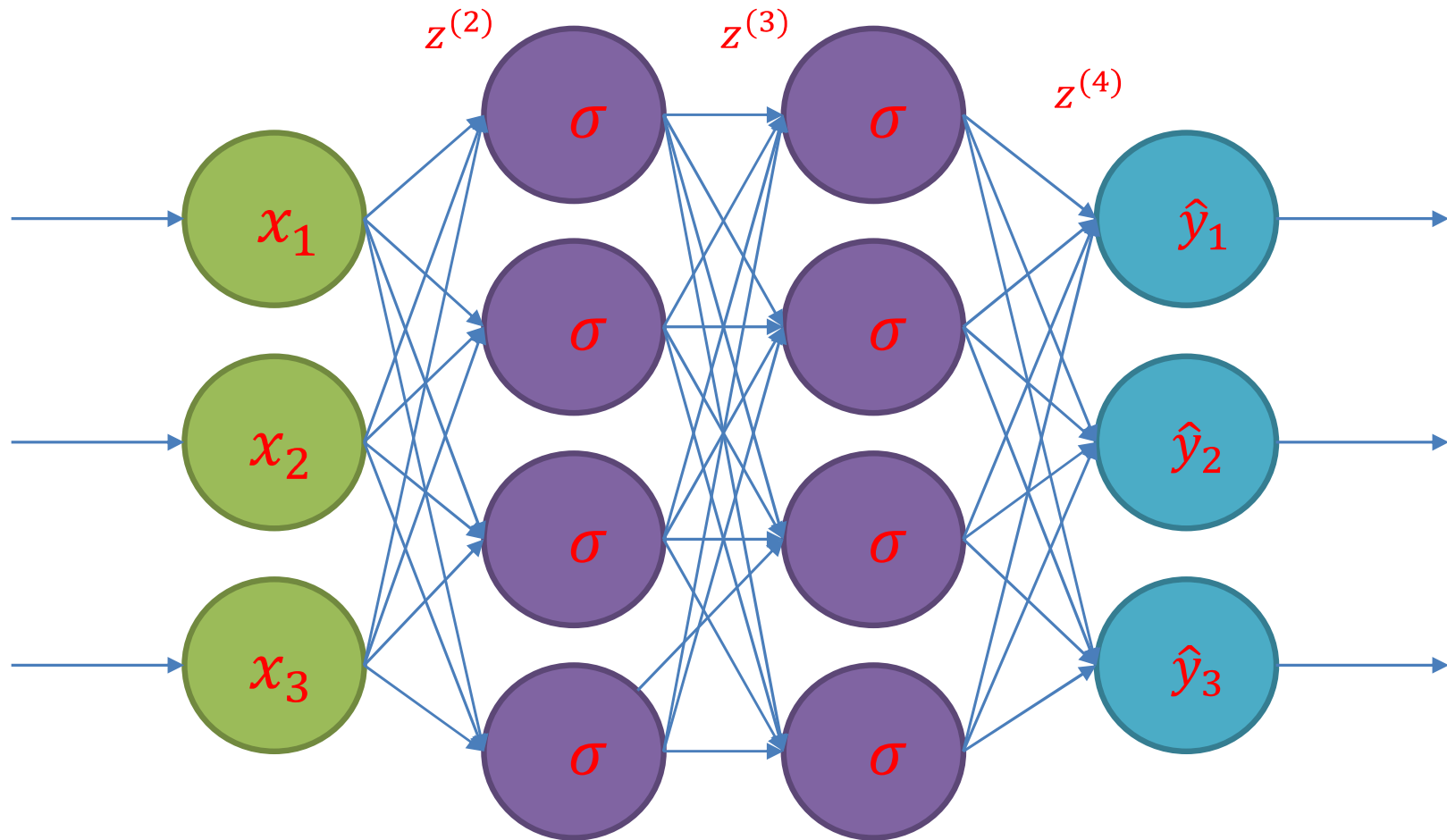
Weights

Input Layer

Hidden Layers

Output Layer

Weights (represented by matrices)

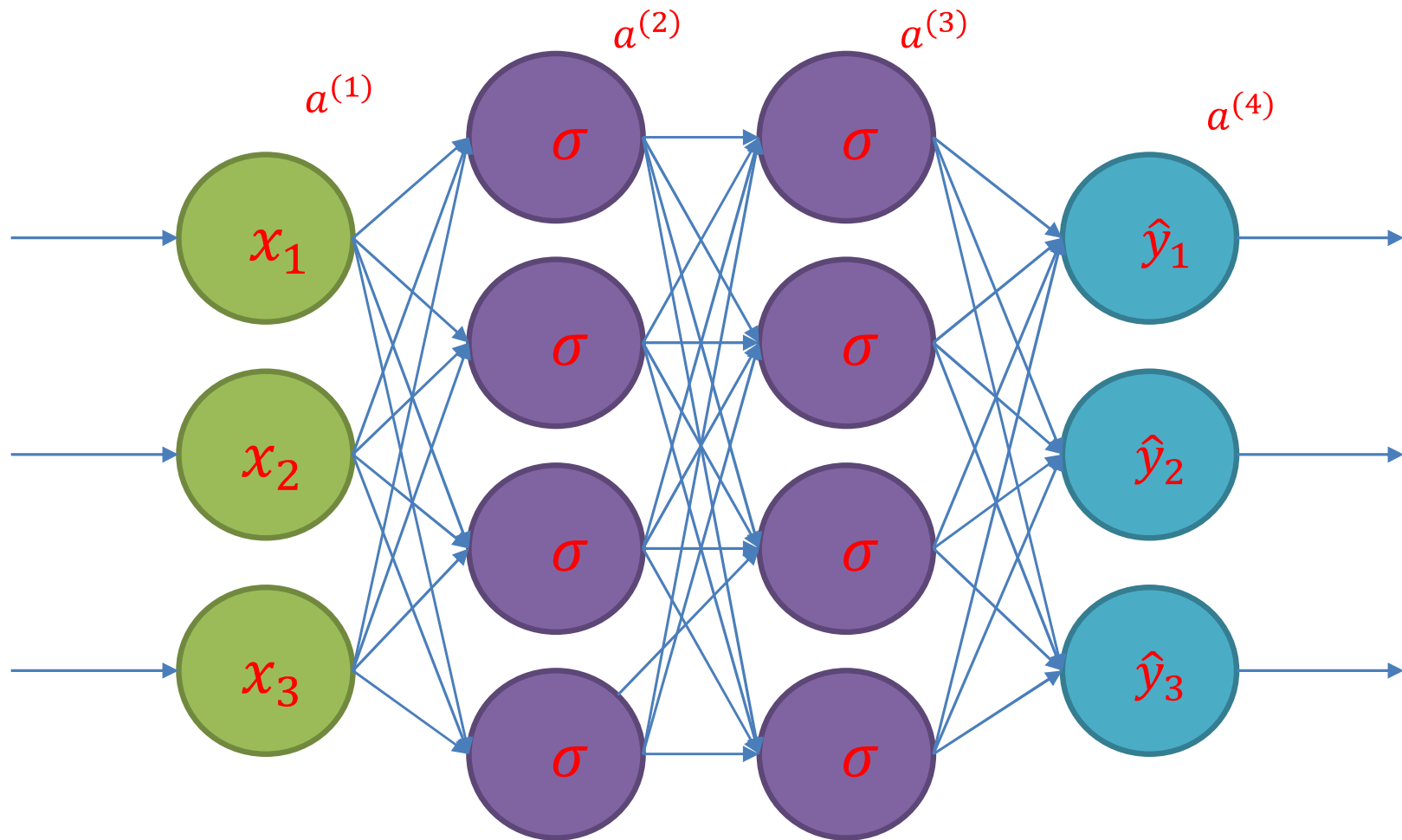# Net Input (sum of weighted inputs, before activation function)

Activations (output of neurons to next layer)

Matrix representation of computation
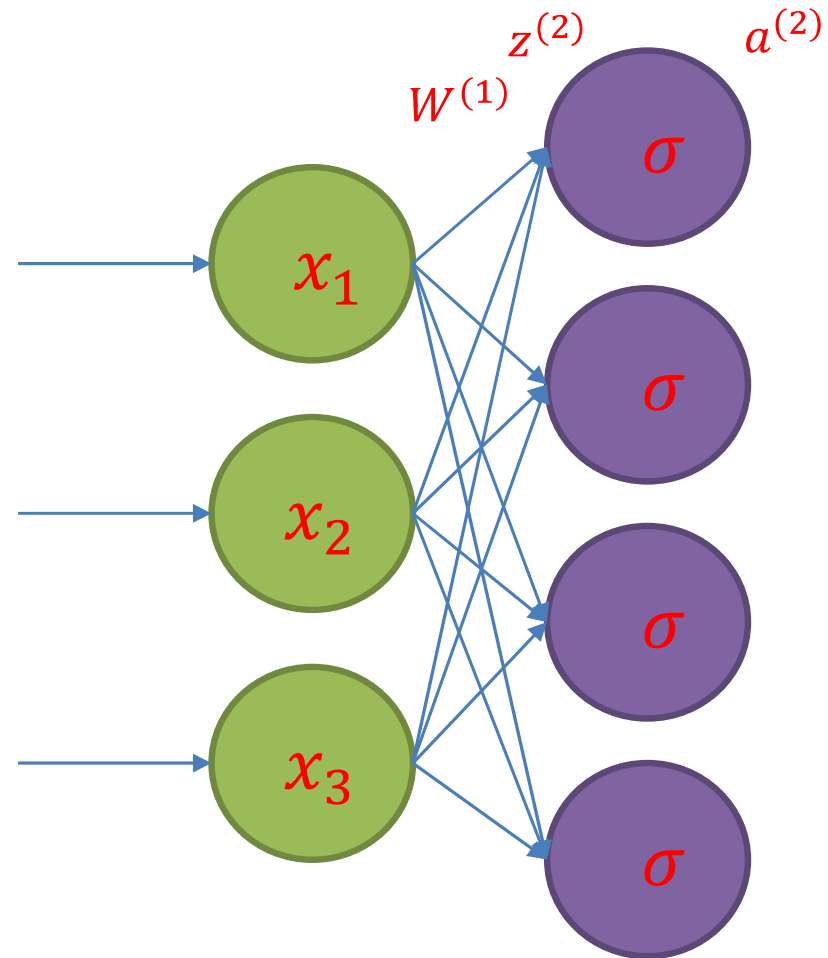
x

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

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

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

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

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

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

$W^{(1)}$

$z^{(2)}$

$a^{(2)}$

x_1

x_2

x_3

σ

σ

σ

σ

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)

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

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

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

$$softmax : \mathbb{R}^K \mapsto (0,1)^K$$

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1,\ldots,K \text{ and } \mathbf{z} = (z_1,\ldots,z_K) \in \mathbb{R}^K$$

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)

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

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

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

Now we know how feedforward NNs do Computations.

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