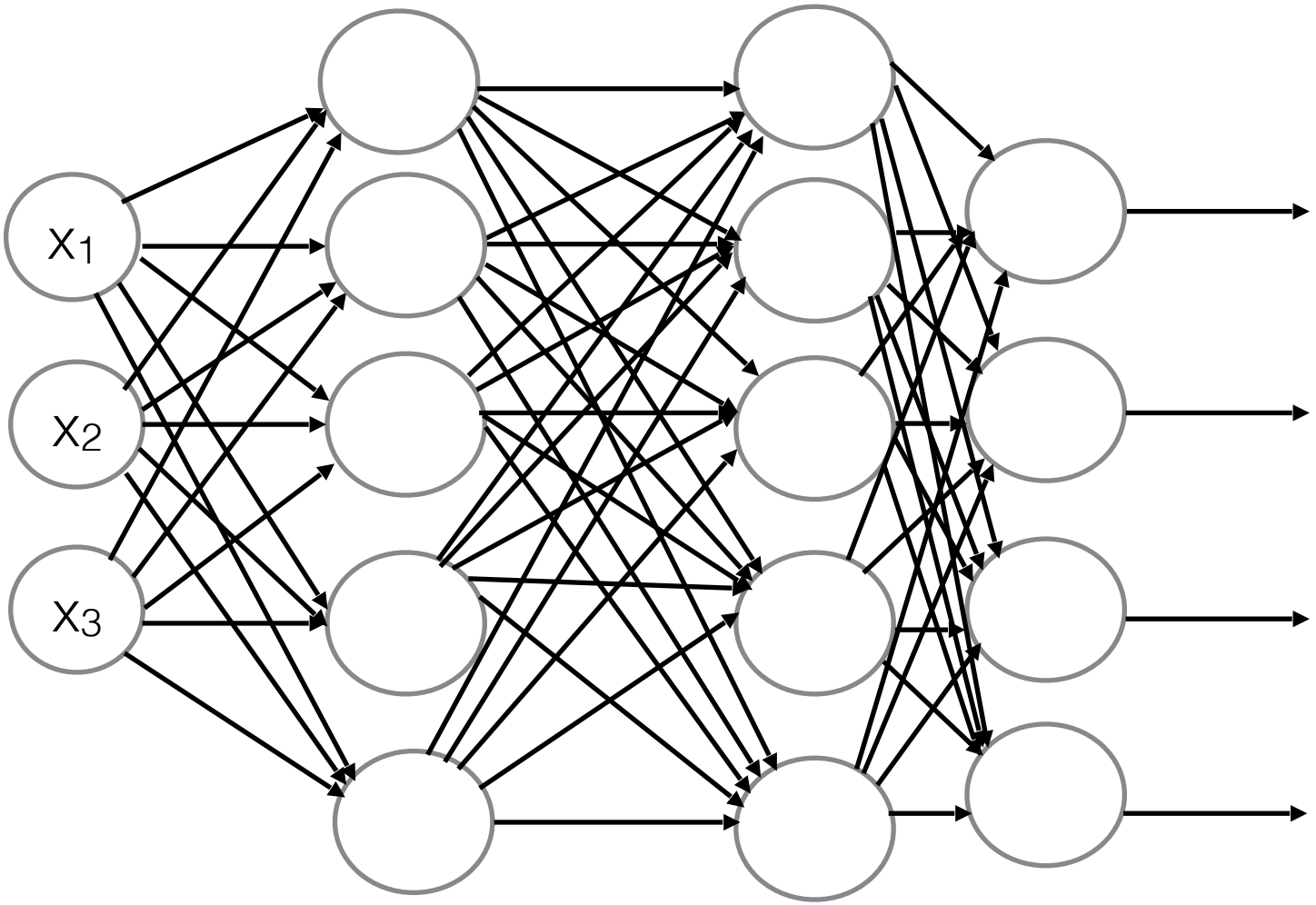
Artificial Neural Networks

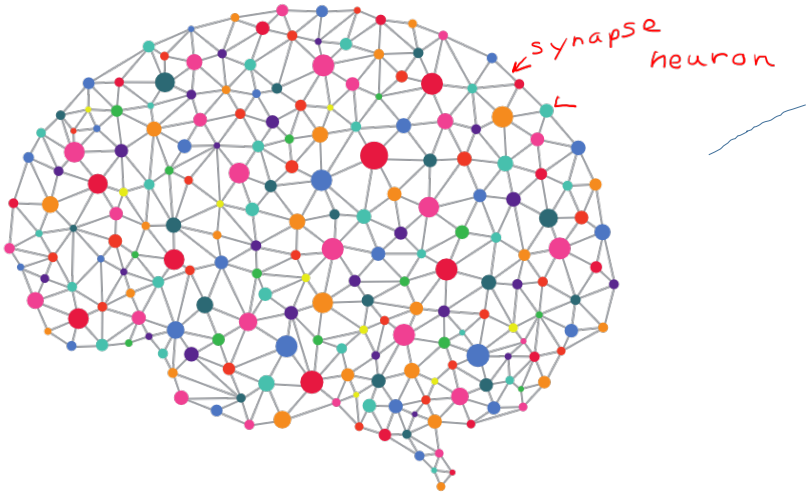


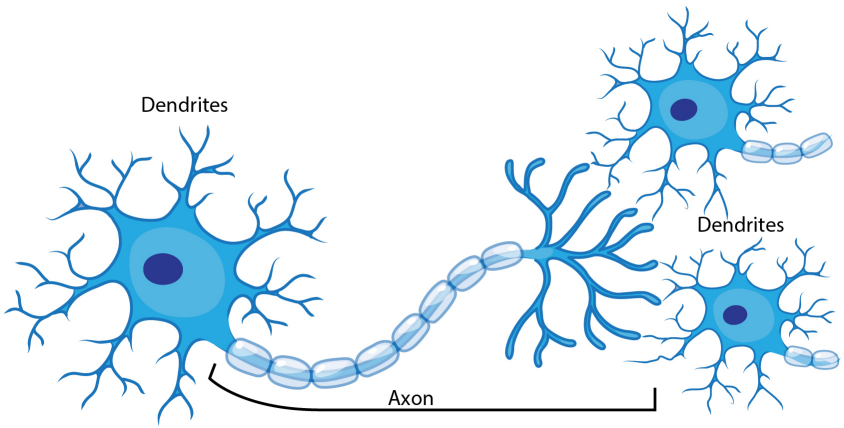
What are they?

Inspired by the Human Brain.

The human brain has about 86 Billion neurons and requires 20% of your body's energy to function.

These neurons are connected to between 100 Trillion to 1 Quadrillion synapses!

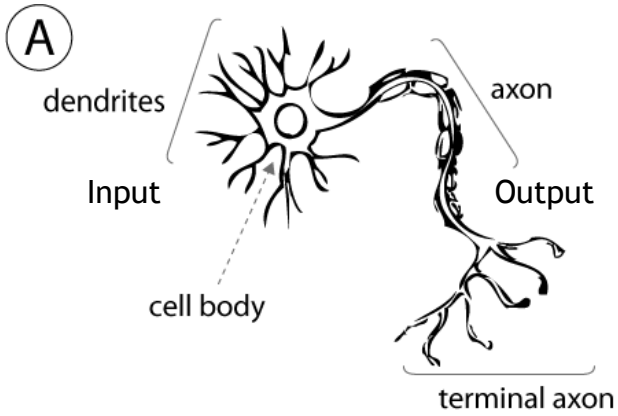


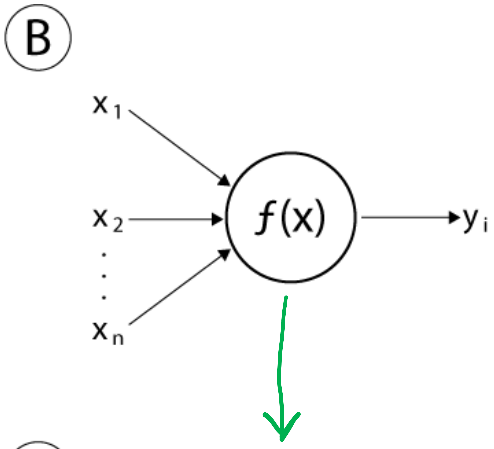


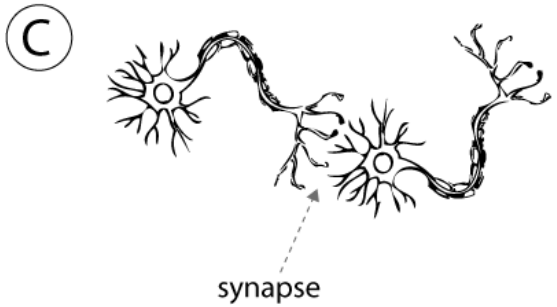
What are they?

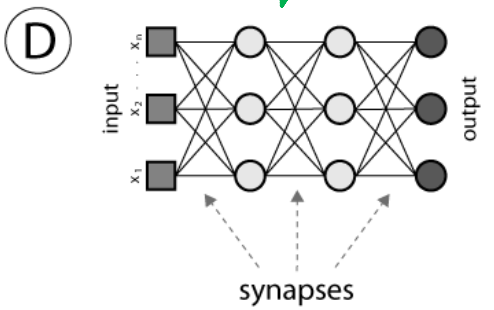
Human

Machine









What are they?

1. Originally developed by Warren McCullough and Walter Pitts (1944)
2. First trainable network, perceptron, proposed by Frank Rosenblatt (1957)
3. Started off as an unsupervised learning tool.
    1. Had problems with computing time and could not compute XOR
    2. Was abandoned in favor of other algorithms
4. Werbos's (1975) backpropagation algorithm
    1. Incorporated supervision and solved XOR
    2. But were still too slow vs. other algorithms e.g., Support Vector Machines
5. Backpropagation was accelerated by GPUs in 2010 and shown to be more efficient and cost effective

GPUS

GPUS handle parallel operations much better (thousands of threads per core) but are not as quick as CPUs. However, the matrix multiplication steps in ANNs can be run in parallel resulting in considerable time + cost savings. The best CPUs handle about 50GB/s while the best GPUs handle 750GB/s memory bandwidth.

	CPU i9 Xseries	GeForce GTX 1080
Cores	18 (36 threads)	2560
Clock Speed (GHz)	4.4	1.6G
Memory	Shared	8GB
Price ($)	1799	549

nature neuroscience

Deep neural networks uncover what the brain likes to see

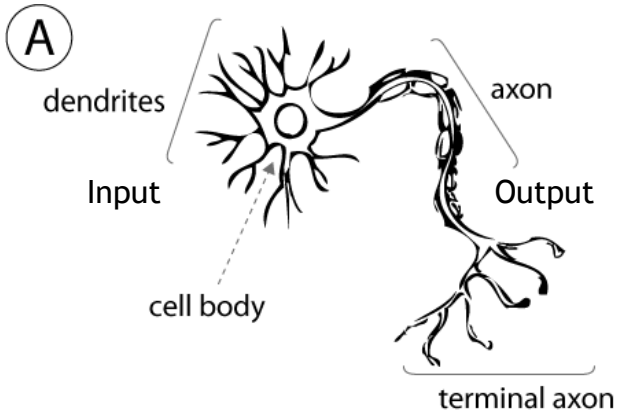
November 4, 2019

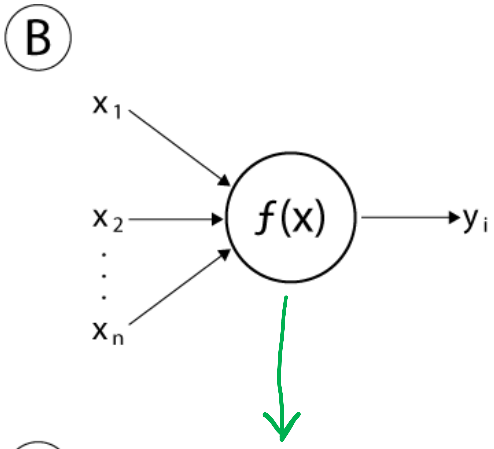
Researchers at Baylor College of Medicine and the University of Tübingen in Germany have developed a novel computational approach to finding stimuli that neurons in the brain 'like.' They built deep artificial neural networks that can accurately predict the neural responses produced by a biological brain to arbitrary visual stimuli.

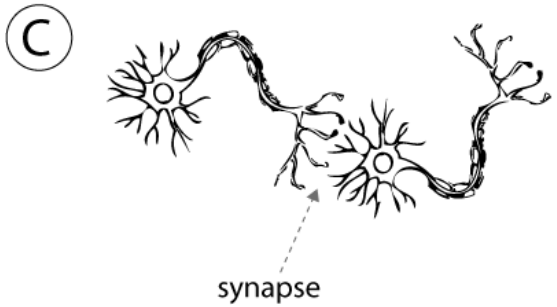
What are they?

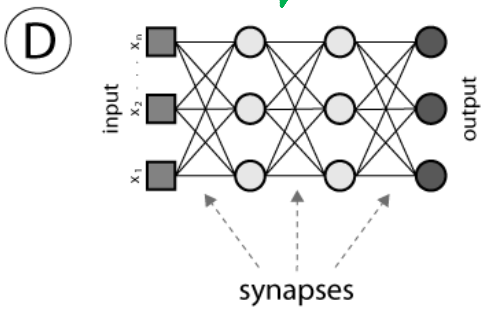
Human

Machine

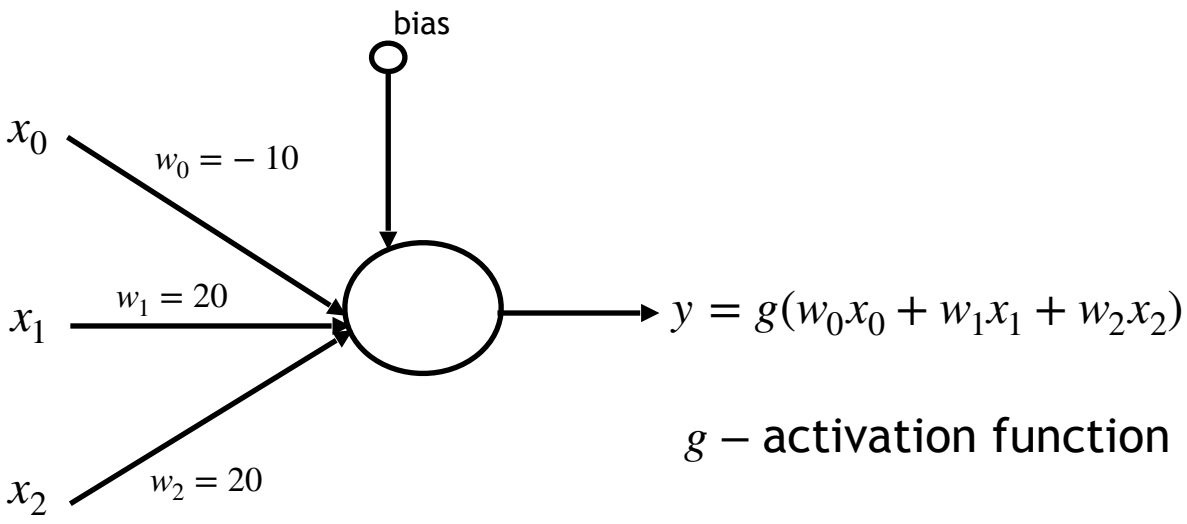




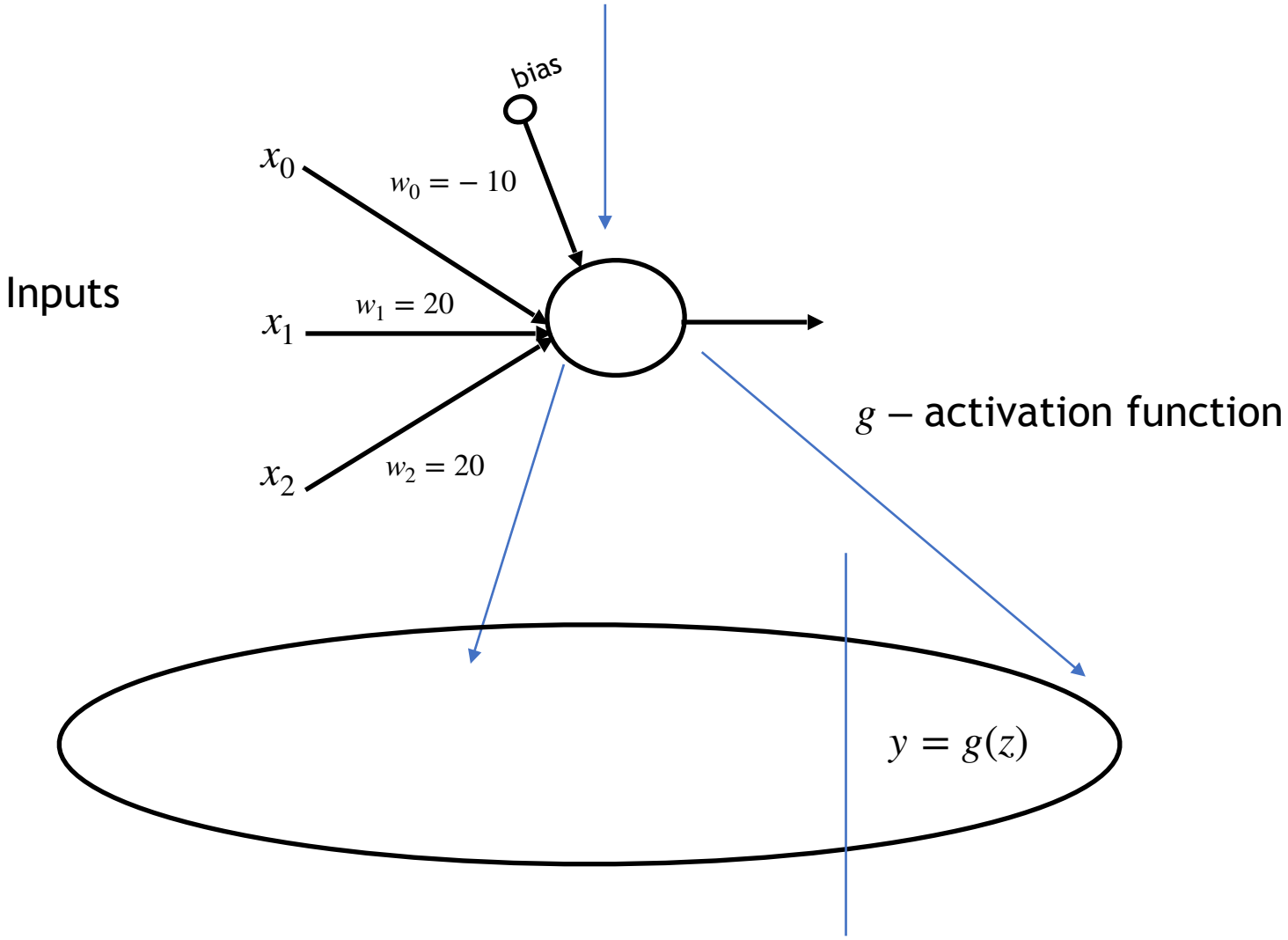




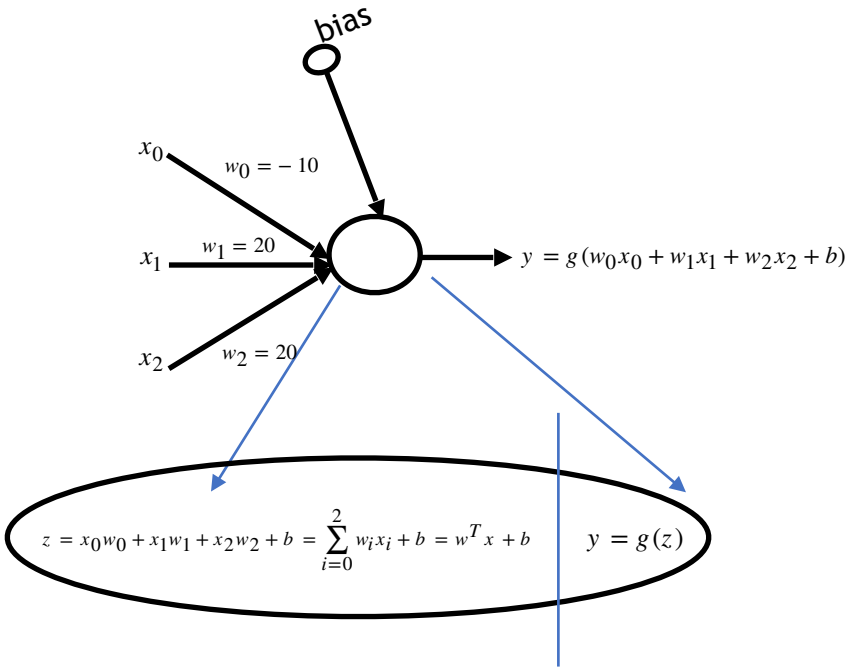
Neuron


bias
x_0
$w_0 = -10$
x_1
$w_1 = 20$
x_2
$w_2 = 20$
$y = g(w_0x_0 + w_1x_1 + w_2x_2)$
g – activation function

Neuron


bias
x_0
$w_0 = -10$
Inputs
x_1
$w_1 = 20$
x_2
$w_2 = 20$
g – activation function
$y = g(z)$

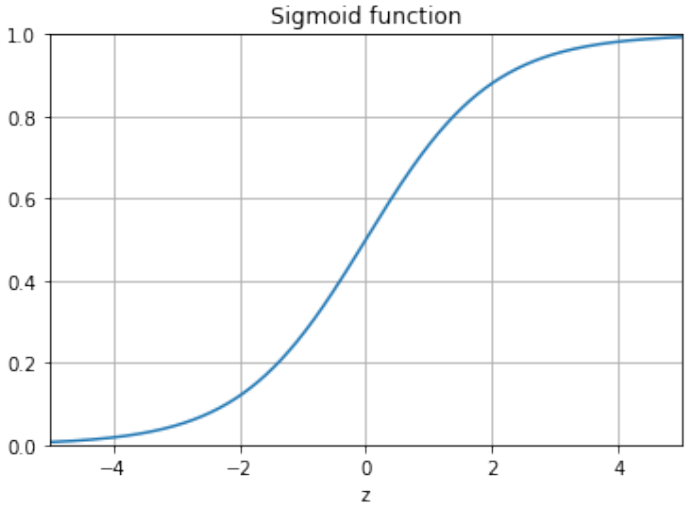
Activation function



g – activation function

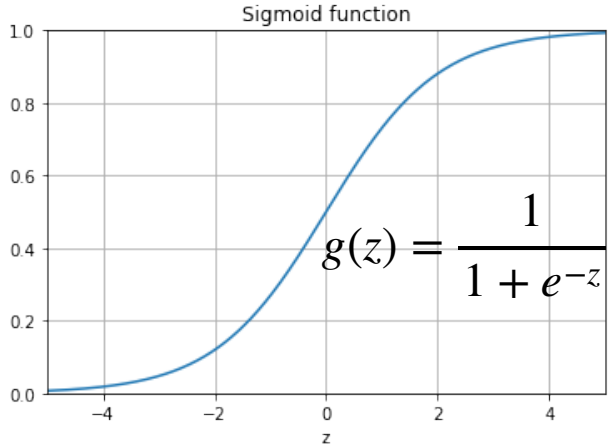
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$z = w_0x_0 + w_1x_1 + w_2x_2 + b = w^Tx + b$$

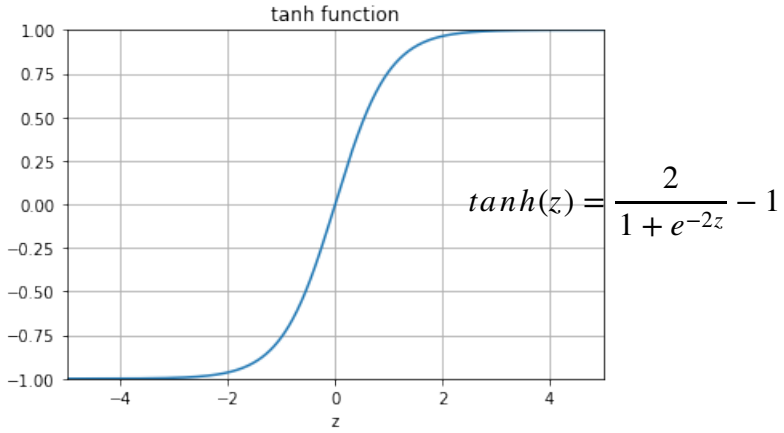


Activation Functions

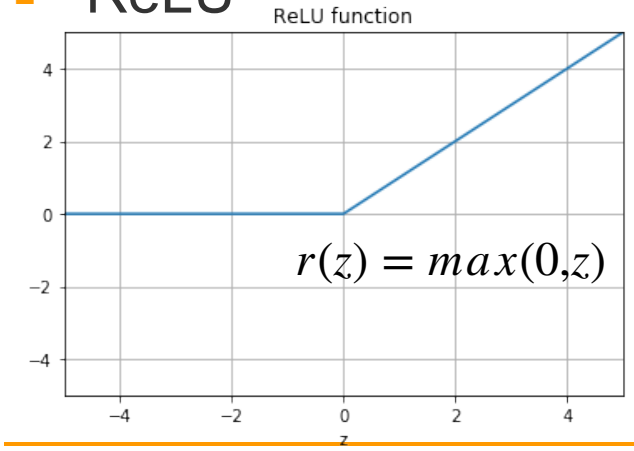
- Sigmoid



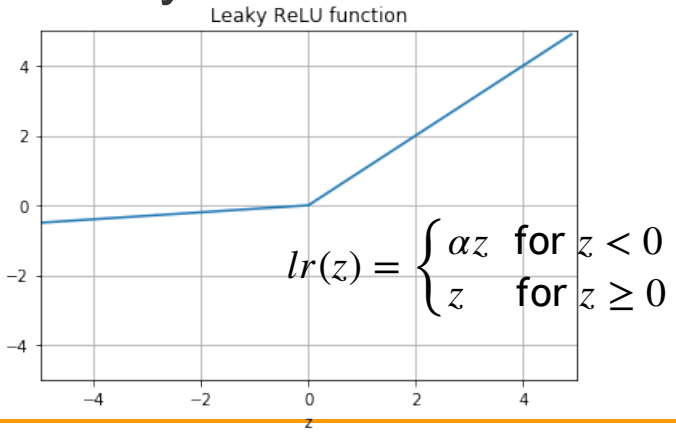
- Hyperbolic tangent



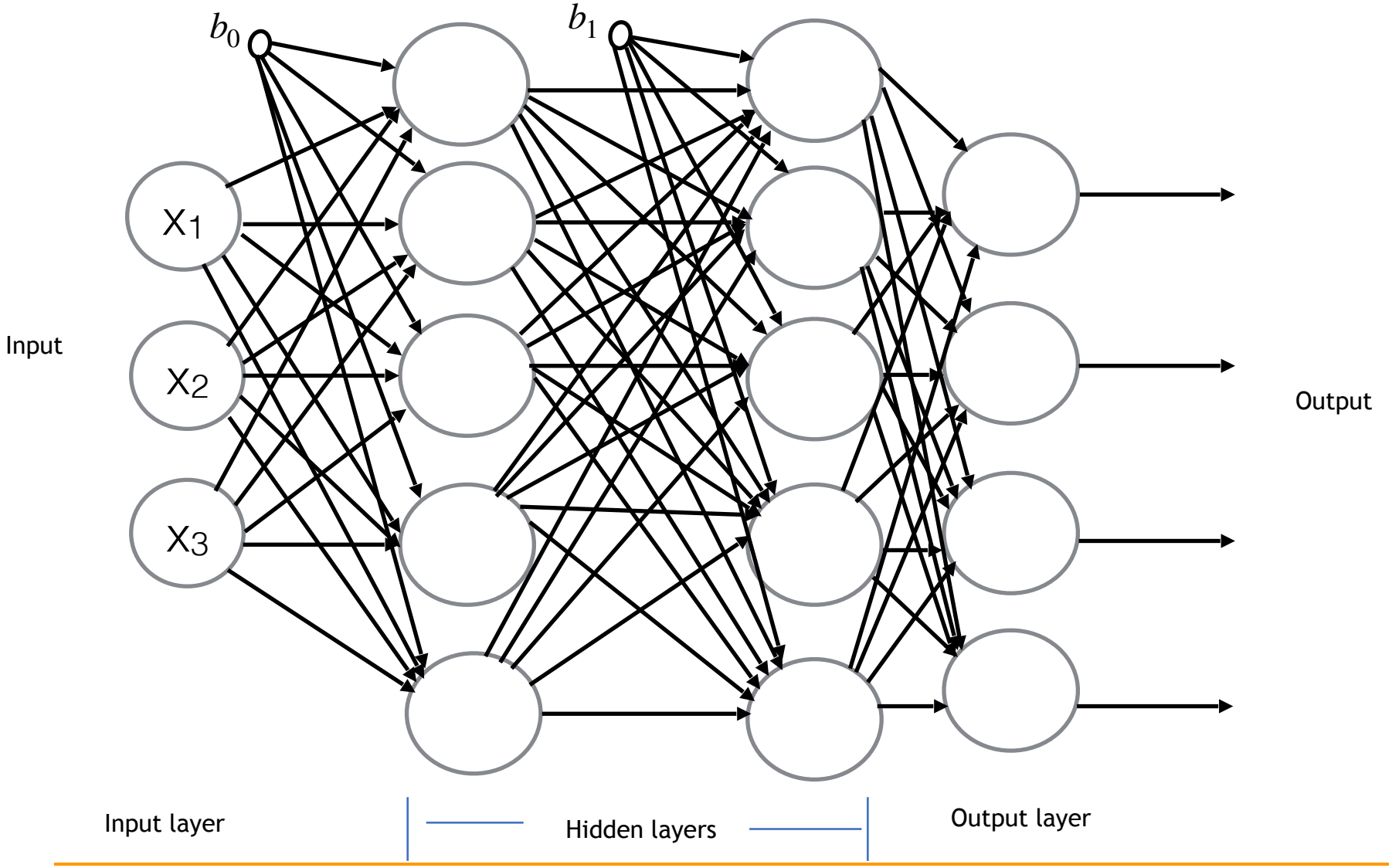
- ReLU



- Leaky ReLU



Neural Network



Neural Network - Logistic Regression

$$\begin{bmatrix} x_{i1} \\ x_{i2} \\ \cdot \\ \cdot \\ x_{ij} \\ \cdot \\ \cdot \\ x_{ip} \end{bmatrix}$$

w_1

w_2

$\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$

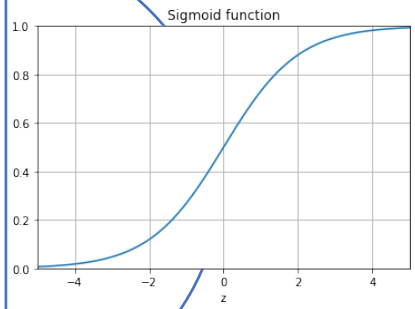
w_j

$\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$

w_p

b

$z_i = w^T x_i + b$



$$\sigma(z_i) = \sigma(w^T x_i + b) = \hat{y}_i = p(x_i)$$

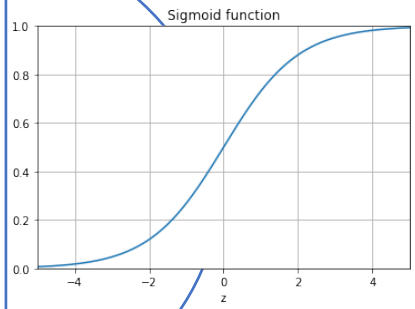
Neural Network - Feedforward propagation

$$\begin{bmatrix} x_{i1} \\ x_{i2} \\ \cdot \\ \cdot \\ x_{ij} \\ \cdot \\ \cdot \\ x_{ip} \end{bmatrix}$$

w_1 w_2 $\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$ w_j $\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$ w_p

b

$$z_i = w^T x_i + b$$



$$\sigma(z_i) = \sigma(w^T x_i + b) = \hat{y}_i = p(x_i)$$

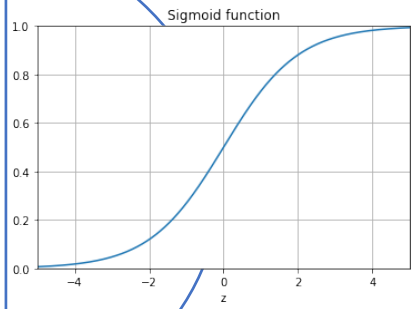
Neural Network - Back propagation

$$\begin{bmatrix} x_{i1} \\ x_{i2} \\ \cdot \\ \cdot \\ x_{ij} \\ \cdot \\ \cdot \\ x_{ip} \end{bmatrix}$$

w_1, w_2, $\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$, w_j, $\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$, w_p

b

$z_i = w^T x_i + b$



$$\sigma(z_i) = \sigma(w^T x_i + b) = \hat{y}_i = p(x_i)$$

Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)$

$$\frac{\partial J}{\partial w} = \frac{1}{m}[\hat{Y} - Y]X$$

$$\frac{\partial J}{\partial b} = \frac{1}{m}[\hat{Y} - Y]$$

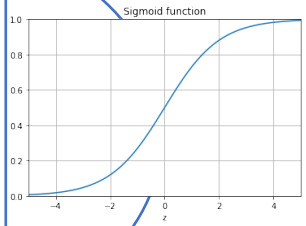
Optimization - Gradient Descent

$$\begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ij} \\ \vdots \\ x_{ip} \end{bmatrix}$$

w_1, w_2, w_j, w_p

b

$z_i = w^T x_i + b$



$\sigma(z_i) = \sigma(w^T x_i + b) = \hat{y}_i = p(x_i)$

$w = 0$

Repeat{

$\hat{y} = \sigma(w^T x + b)$ # feed forward

$\frac{\partial J}{\partial w} = \frac{1}{m}[\hat{y} - y]X$

$\frac{\partial J}{\partial b} = \frac{1}{m}[\hat{Y} - Y]$ # back prop

$w = w - \alpha \frac{\partial J}{\partial w}$

$b = b - \alpha \frac{\partial J}{\partial b}$ *#update*

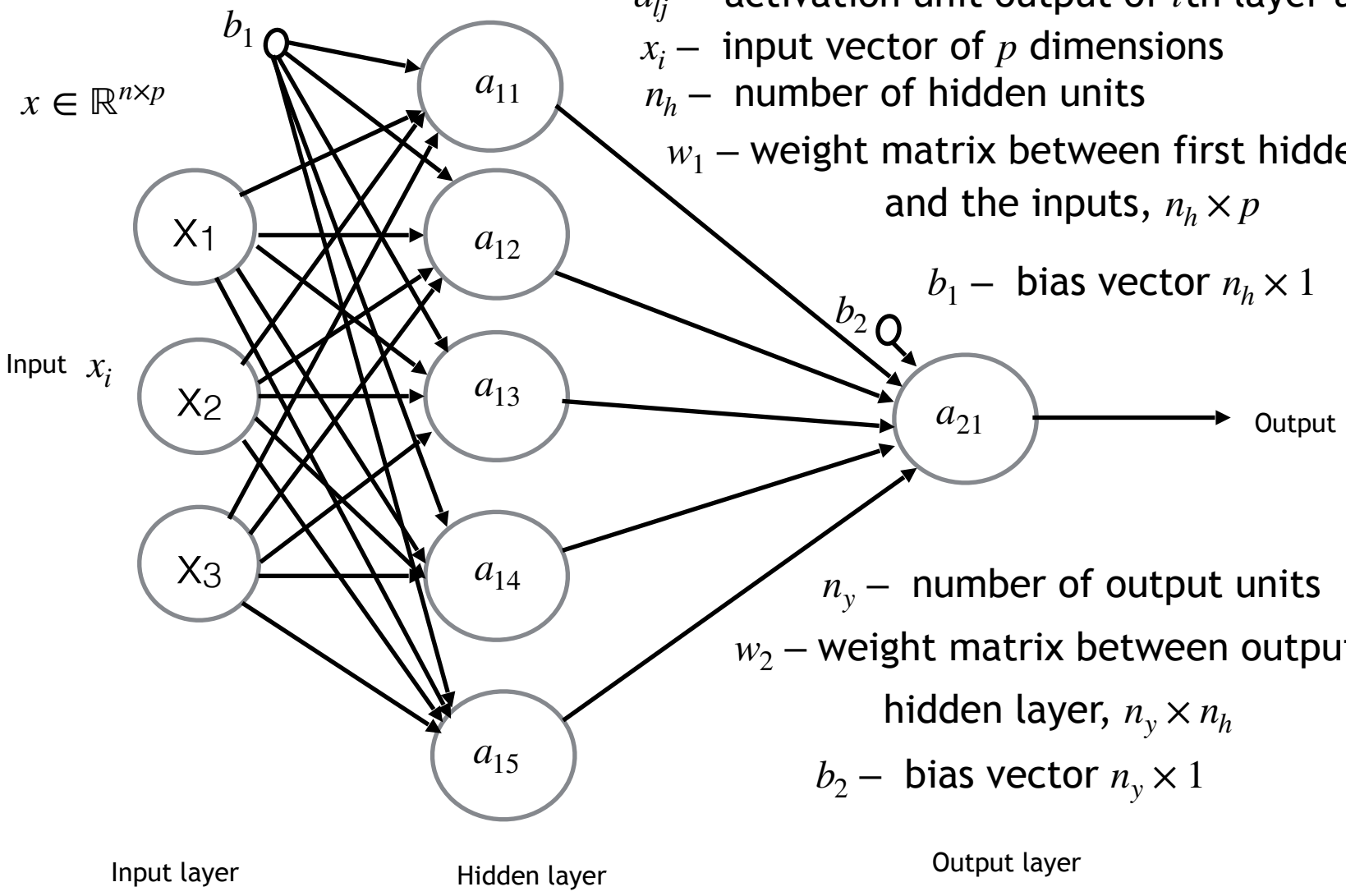
}

Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)$

$\frac{\partial J}{\partial w} = \frac{1}{m}[\hat{Y} - Y]X$

$\frac{\partial J}{\partial b} = \frac{1}{m}[\hat{Y} - Y]$

Neural Network with one hidden layer



a_{lj} – activation unit output of lth layer and jth unit

x_i – input vector of p dimensions

n_h – number of hidden units

w_1 – weight matrix between first hidden layer and the inputs, $n_h \times p$

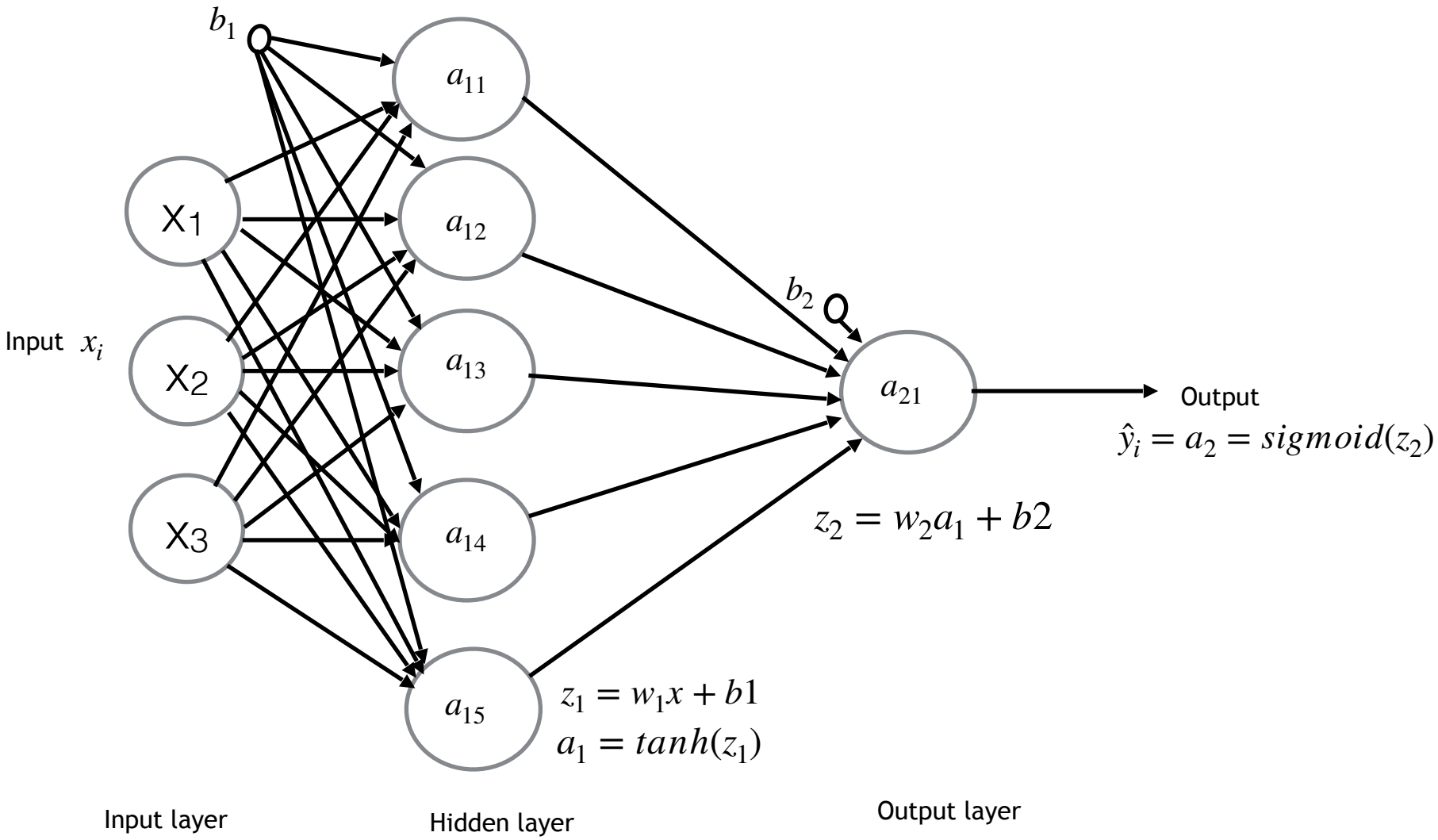
b_1 – bias vector $n_h \times 1$

n_y – number of output units

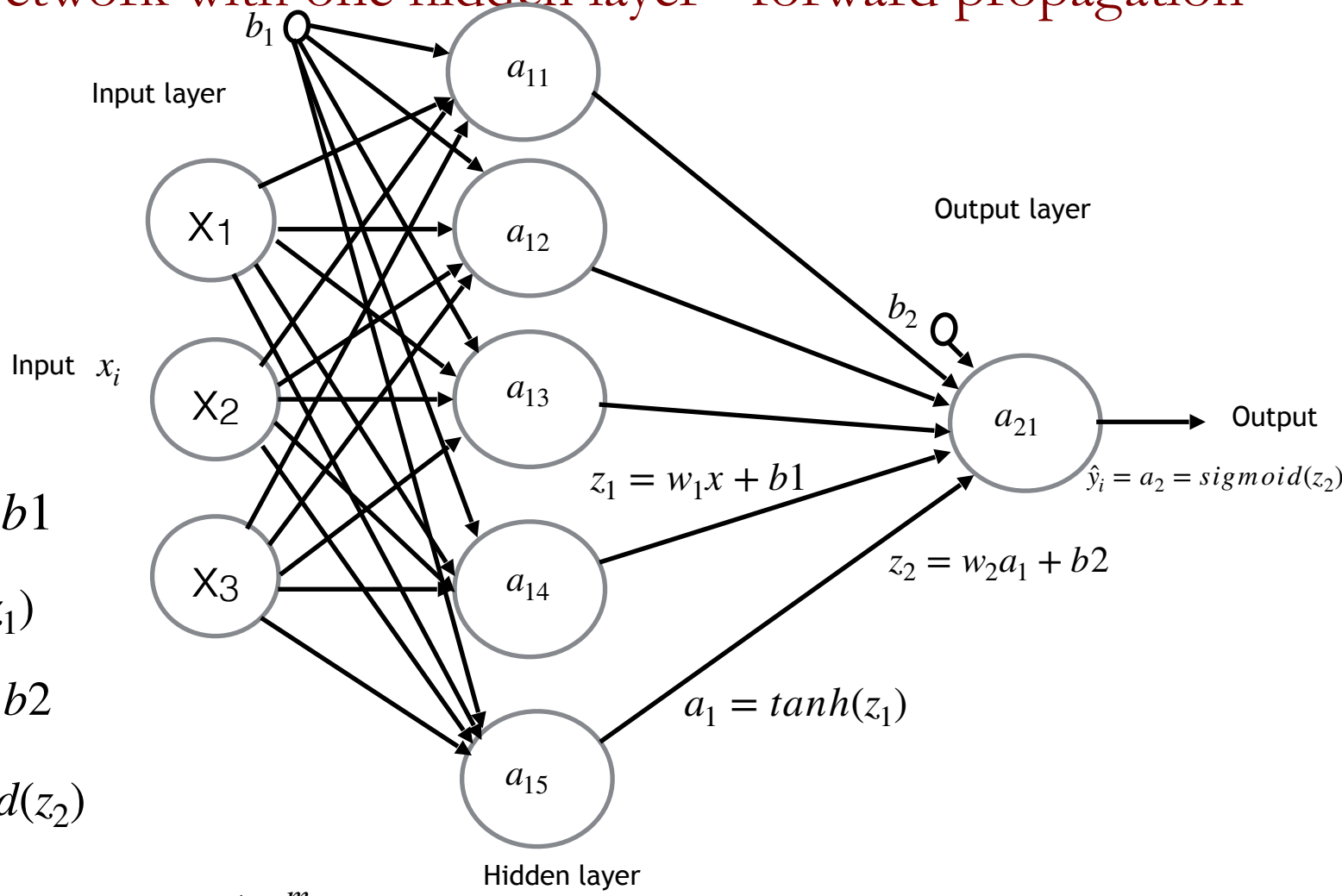
w_2 – weight matrix between output and hidden layer, $n_y \times n_h$

b_2 – bias vector $n_y \times 1$

Neural Network with one hidden layer



Neural Network with one hidden layer - forward propagation



$$z_1 = w_1 x + b1$$

$$a_1 = tanh(z_1)$$

$$z_2 = w_2 a_1 + b2$$

$$a_2 = sigmoid(z_2)$$

Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

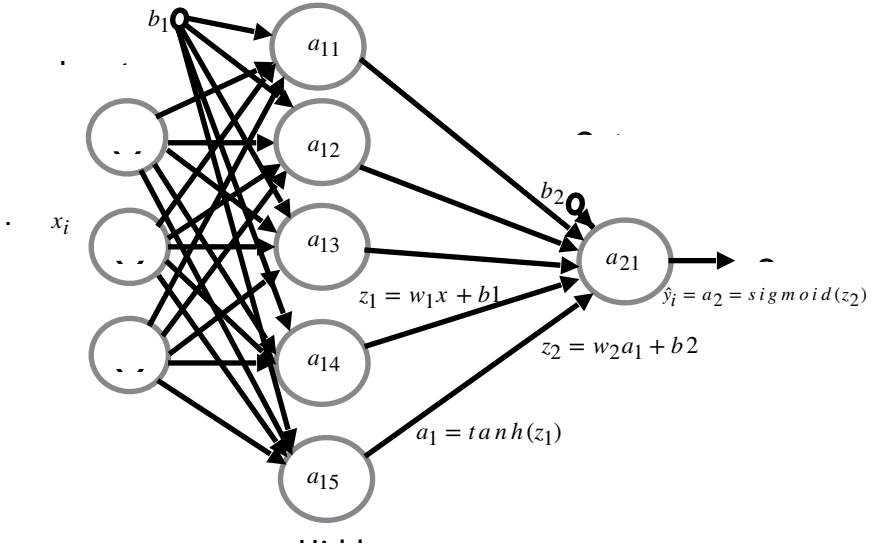
Neural Network with one hidden layer - Back propagation

$$z_1 = w_1 x + b1$$

$$a_1 = tanh(z_1)$$

$$z_2 = w_2 a_1 + b2$$

$$a_2 = sigmoid(z_2)$$

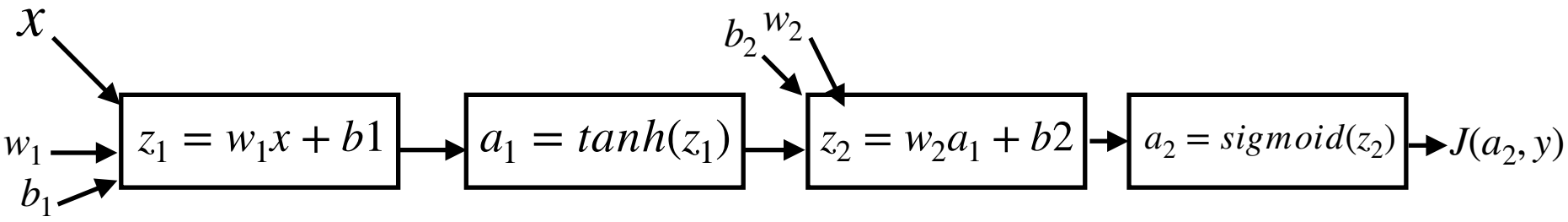


Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} \quad \frac{\partial J}{\partial w_2} \quad \frac{\partial J}{\partial b_2}$$

$$\frac{\partial J}{\partial z_1} \quad \frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial b_1}$$

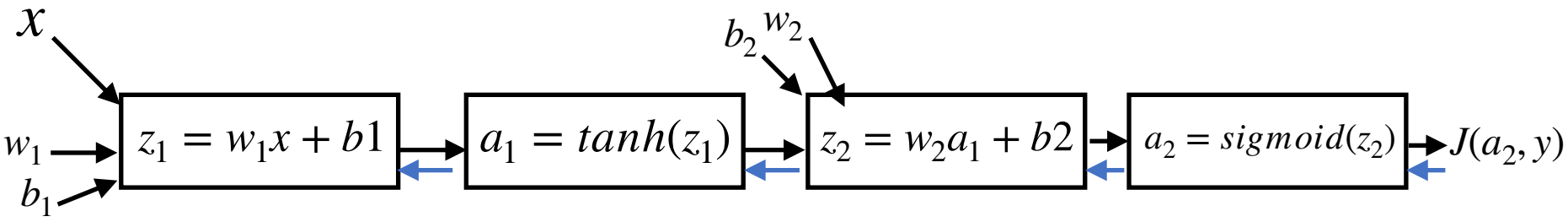
Neural Network with one hidden layer - Forward and Back propagation Block diagram



Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} \quad \frac{\partial J}{\partial w_2} \quad \frac{\partial J}{\partial b_2} \qquad \frac{\partial J}{\partial z_1} \quad \frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial b_1}$$

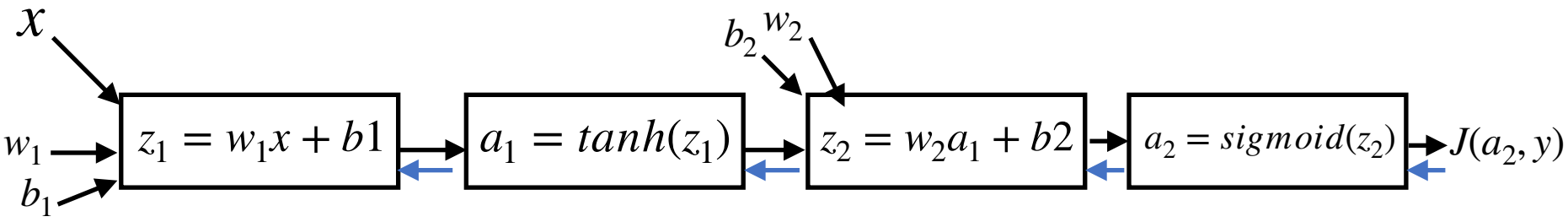
Neural Network with one hidden layer - Forward and Back propagation Block diagram



Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} \quad \frac{\partial J}{\partial w_2} \quad \frac{\partial J}{\partial b_2} \qquad \frac{\partial J}{\partial z_1} \quad \frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial b_1}$$

Neural Network with one hidden layer - Back propagation

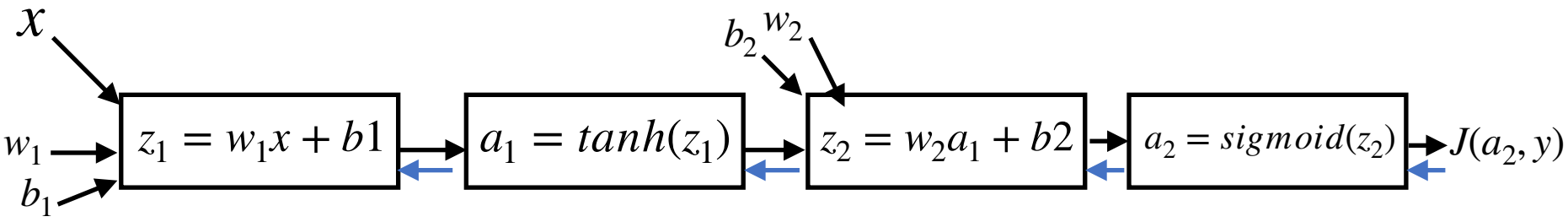


Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} \quad \frac{\partial J}{\partial w_2} \quad \frac{\partial J}{\partial b_2} \qquad \frac{\partial J}{\partial z_1} \quad \frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial b_1}$$

$$\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} = \left[\frac{-y}{a_2} + \frac{1-y}{1-a_2}\right] \cdot a_2(1-a_2) = a_2 - y$$

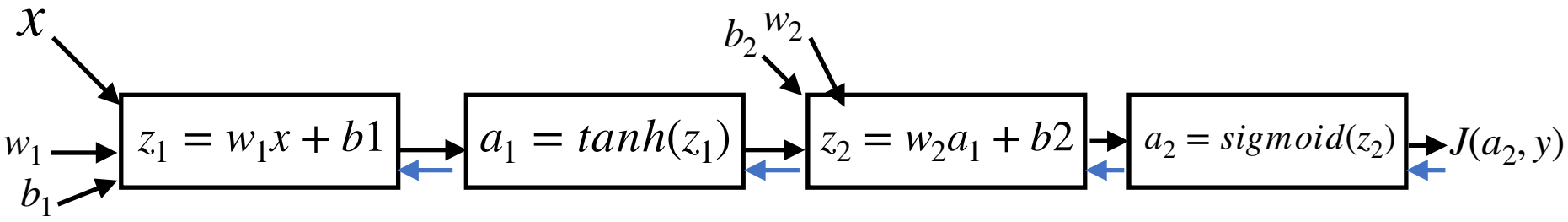
Neural Network with one hidden layer - Back propagation



Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} = a_2 - y \qquad \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2} \cdot a_1^T \qquad \frac{\partial J}{\partial b_2} \qquad \frac{\partial J}{\partial z_1} \qquad \frac{\partial J}{\partial w_1} \qquad \frac{\partial J}{\partial b_1}$$

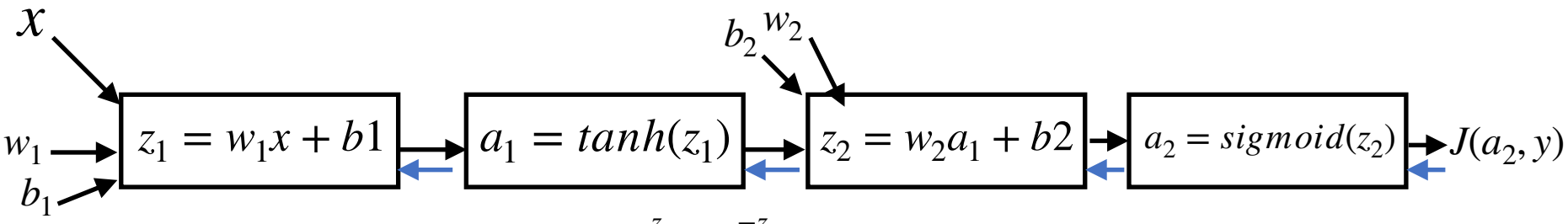
Neural Network with one hidden layer - Back propagation



Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} = a_2 - y \qquad \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2} \cdot a_1^T \qquad \frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_2} \qquad \frac{\partial J}{\partial z_1} \qquad \frac{\partial J}{\partial w_1} \qquad \frac{\partial J}{\partial b_1}$$

Neural Network with one hidden layer - Back propagation



$$a_1 = tanh(z_1) = \frac{e^{z_1} - e^{-z_1}}{e^{z_1} + e^{-z_1}}$$

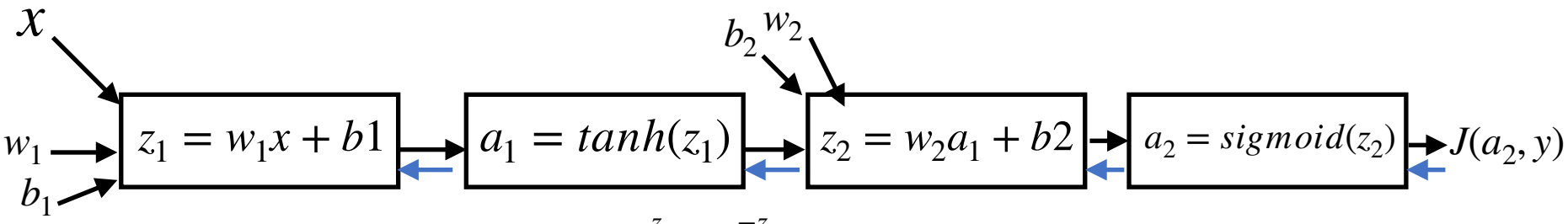
Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} = a_2 - y \qquad \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2} \cdot a_1^T \qquad \frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_2} \qquad \frac{\partial J}{\partial w_1} \qquad \frac{\partial J}{\partial b_1}$$

$$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} * \frac{\partial a_1}{\partial z_1} = w_2^T \left(\frac{\partial J}{\partial z_2} \right) * \frac{\partial a_1}{\partial z_1} = w_2^T (a_2 - y) \frac{\partial a_1}{\partial z_1} = w_2^T (a_2 - y) * (1 - a_1^2)$$

* element by element multiplication

Neural Network with one hidden layer - Back propagation



$$a_1 = tanh(z_1) = \frac{e^{z_1} - e^{-z_1}}{e^{z_1} + e^{-z_1}}$$

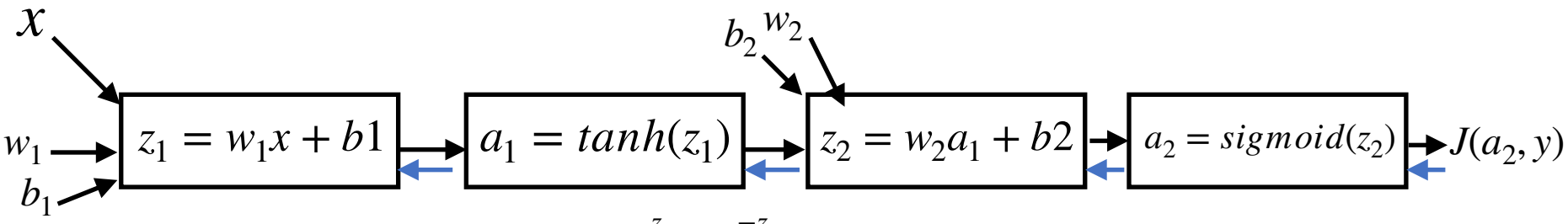
Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} = a_2 - y \quad ; \quad \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2} \cdot a_1^T \quad ; \quad \frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_2} \quad ; \quad \frac{\partial J}{\partial z_1} = w_2^T (a_2 - y) * (1 - a_1^2);$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} = \frac{\partial J}{\partial z_1} \cdot x^T \qquad \frac{\partial J}{\partial b_1}$$

* element by element multiplication

Neural Network with one hidden layer - Back propagation



$$a_1 = tanh(z_1) = \frac{e^{z_1} - e^{-z_1}}{e^{z_1} + e^{-z_1}}$$

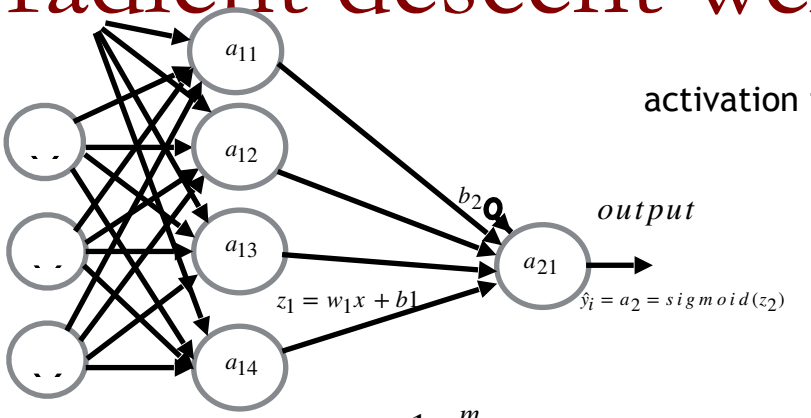
Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$

$$\frac{\partial J}{\partial z_2} = a_2 - y \quad ; \qquad \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2} \cdot a_1^T \quad ; \qquad \frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_2} \quad ;$$

$$\frac{\partial J}{\partial z_1} = w_2^T (a_2 - y) * (1 - a_1^2) \; ; \qquad \frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_1} \cdot x^T \; ; \qquad \frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial z_1} ;$$

* element by element multiplication

Gradient descent weight updates



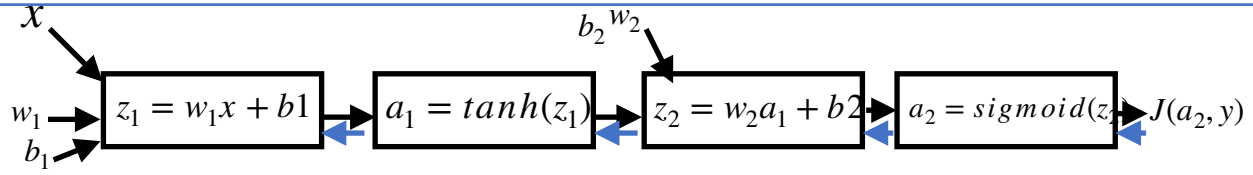
activation function for hidden layer, $a_1 = g(z1) = tanh(z_1) = \frac{e^{z_1} - e^{-z_1}}{e^{z_1} + e^{-z_1}}$

$$g'(z1) = 1 - a_1^2$$

output, $a_2 = g(z_2) = \frac{1}{1 + e^{-z}}$

$$g'(z_2) = \frac{\partial}{\partial z} g(z) = \frac{\partial}{\partial z}\left(\frac{1}{1 + e^{-z}}\right) = g(z_2)[1 - g(z_2)] = a_2(1 - a_2)$$

Cross Entropy loss, $J = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(a_2) + (1 - y_i) * log(1 - a_2)$



feedforward $z_1 = w_1 x + b1 \quad a_1 = tanh(z_1) \quad z_2 = w_2 a_1 + b2 \quad a_2 = sigmoid(z_2)$

backprop

$$\frac{\partial J}{\partial z_2} = a_2 - y \;; \qquad \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2} \cdot a_1^T \;; \qquad \frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial z_2} \;;$$

$$\frac{\partial J}{\partial z_1} = w_2^T (a_2 - y) * (1 - a_1^2) \;; \qquad \frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_1} \cdot x^T \;; \qquad \frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial z_1} \;;$$

Learning rate, α

grad descent weight update

$$w_2 = w_2 - \alpha \frac{\partial J}{\partial w_2}; \qquad b_2 = w_4 - \alpha \frac{\partial J}{\partial b_2}; \qquad w_1 = w_1 - \alpha \frac{\partial J}{\partial w_1}; \qquad b_1 = b_1 - \alpha \frac{\partial J}{\partial b_1};$$