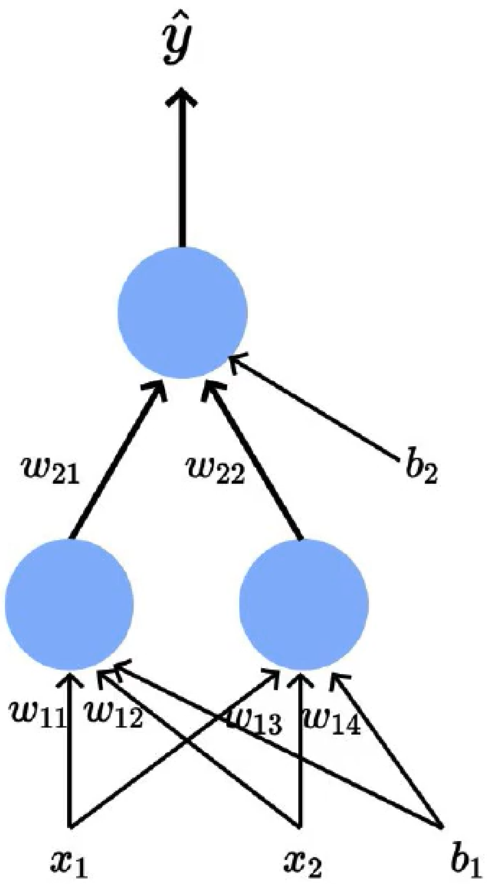
AMRITA
VISHWA VIDYAPEETHAM
श्रद्धावान् लभते ज्ञानम्
DEEMED TO BE UNIVERSITY

DEEP LEARNING

Feed Forward Neural Networks-Introduction

Citation Note: The content, of this presentation were inspired by the awesome lectures and the material offered by Prof. ***Mitesh M. Khapra*** *on NPTEL's Deep Learning course*

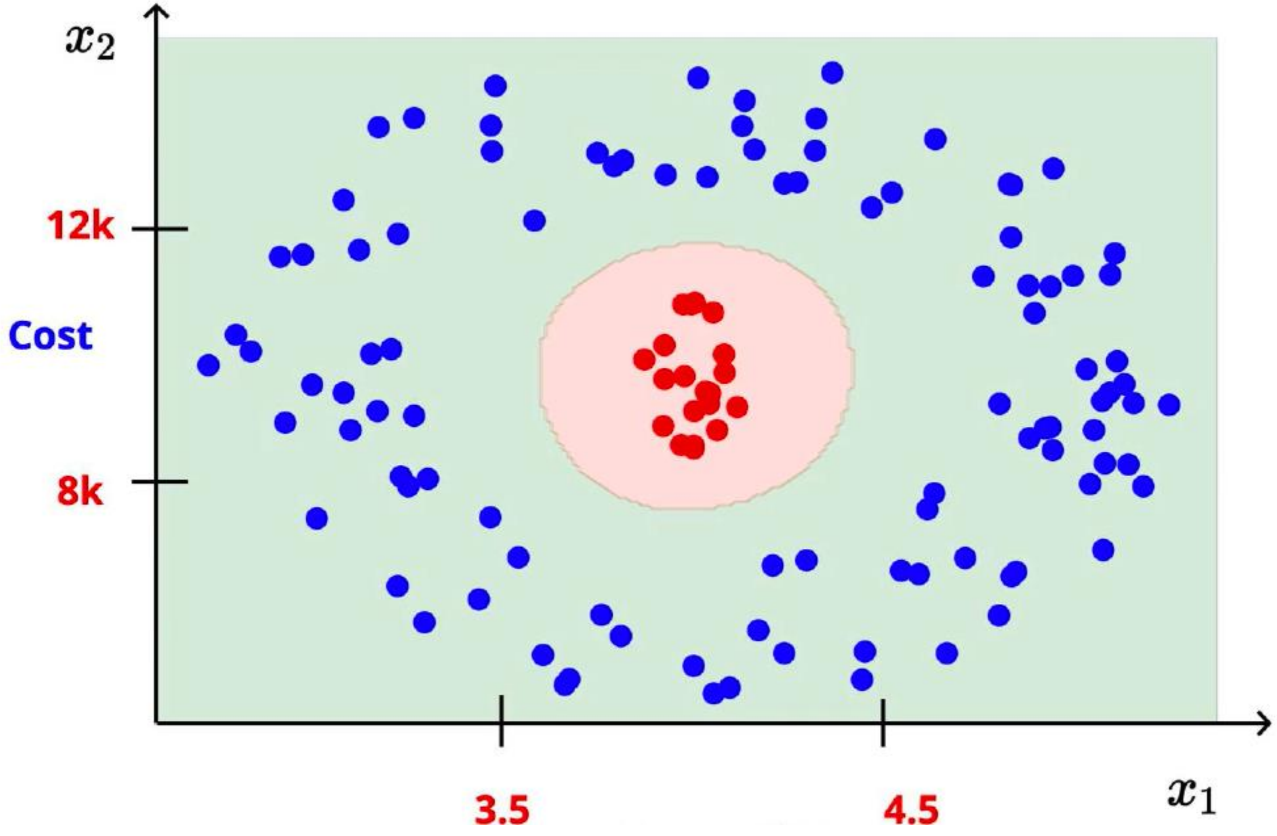
Multiple sigmoid Neurons



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



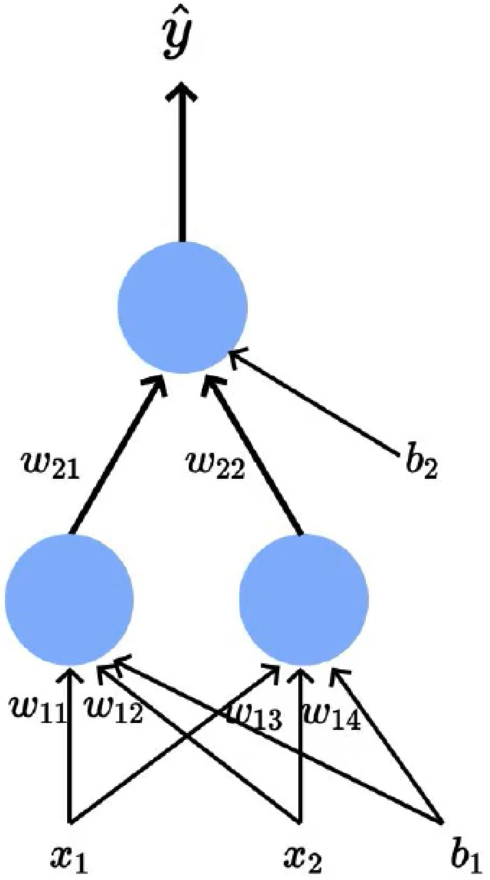
$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

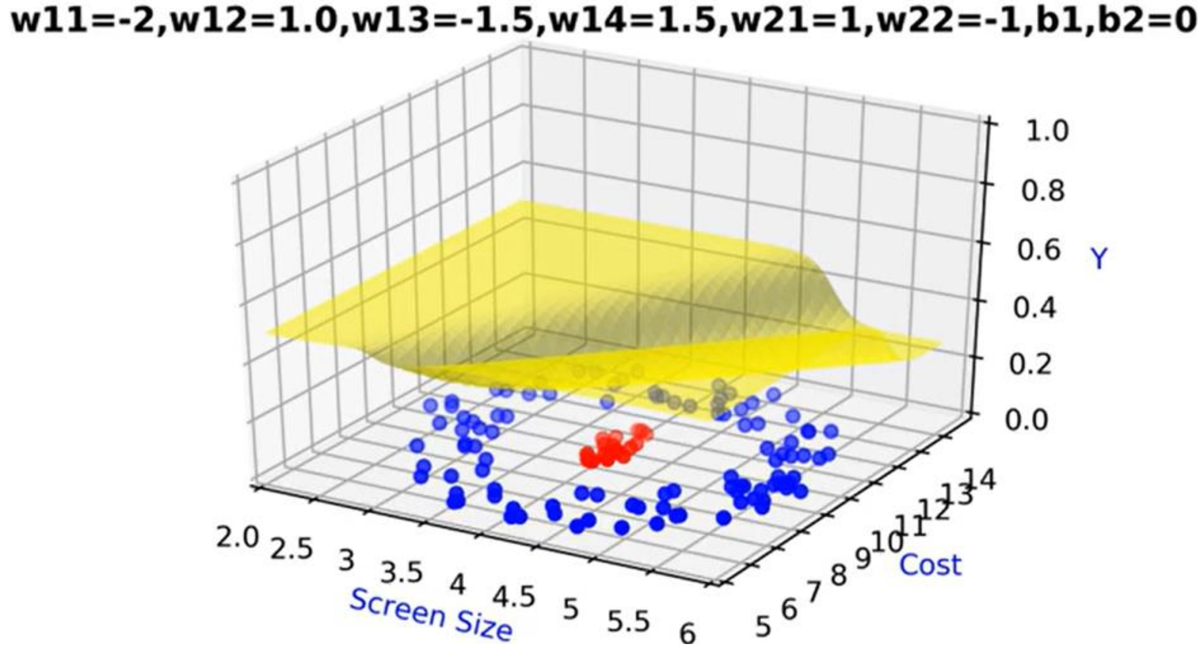
Multiple sigmoid Neurons



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



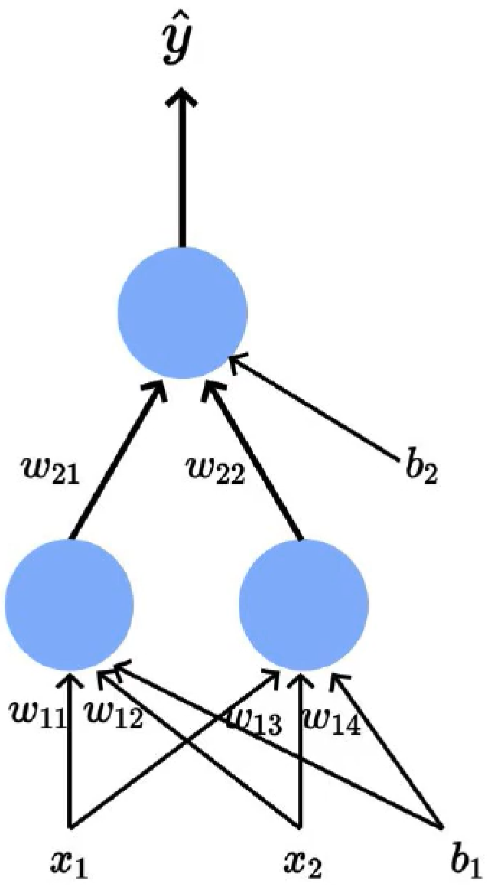
$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

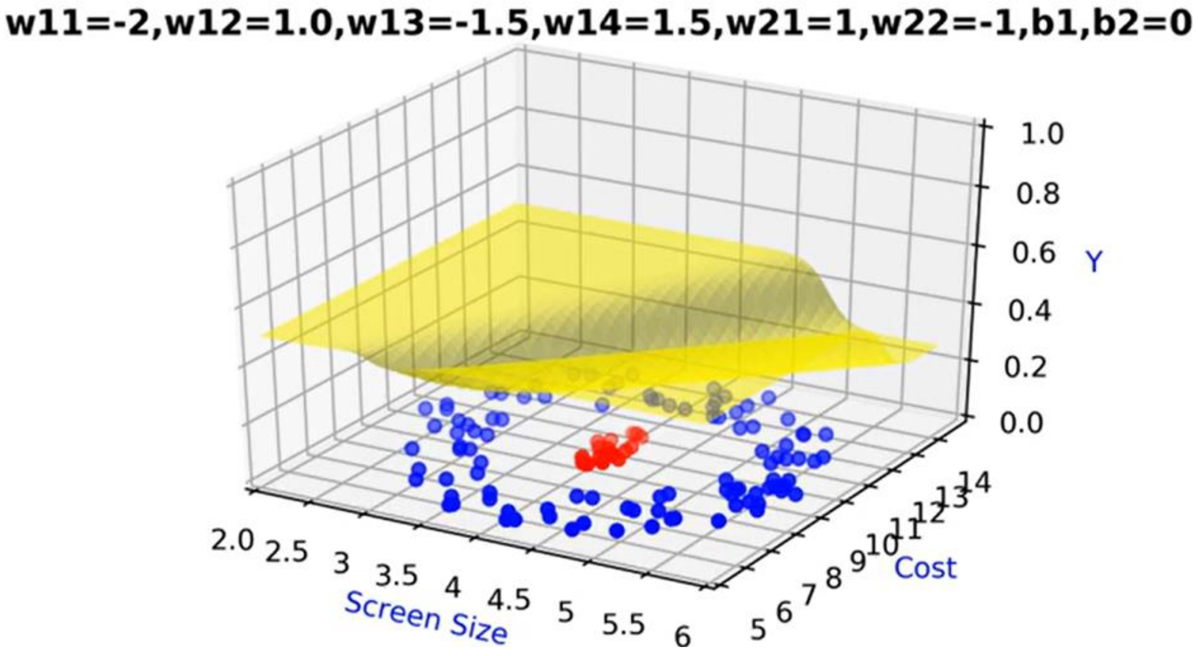
Multiple sigmoid Neurons



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



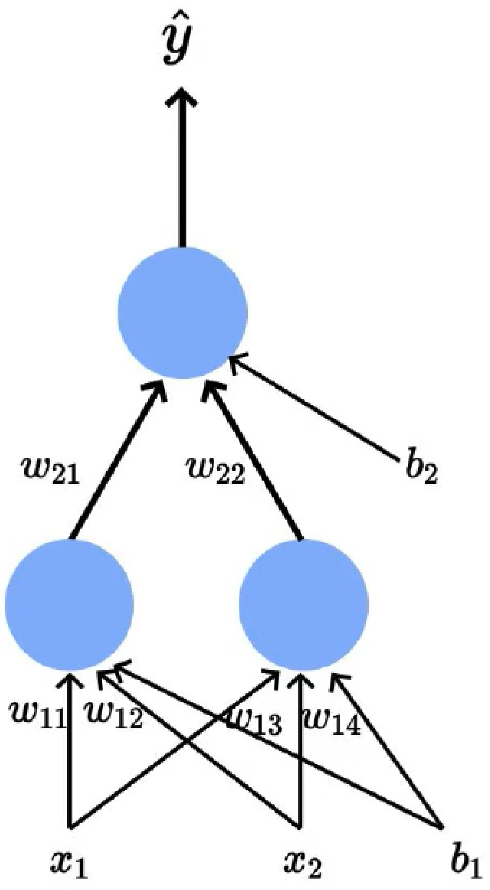
$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

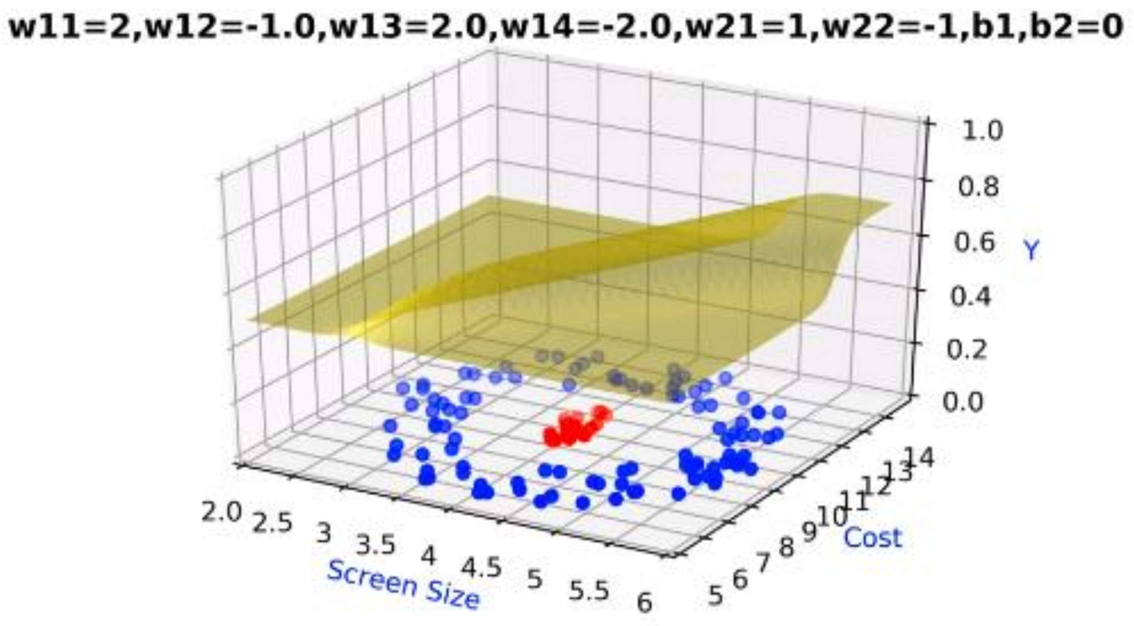
Multiple sigmoid Neurons



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



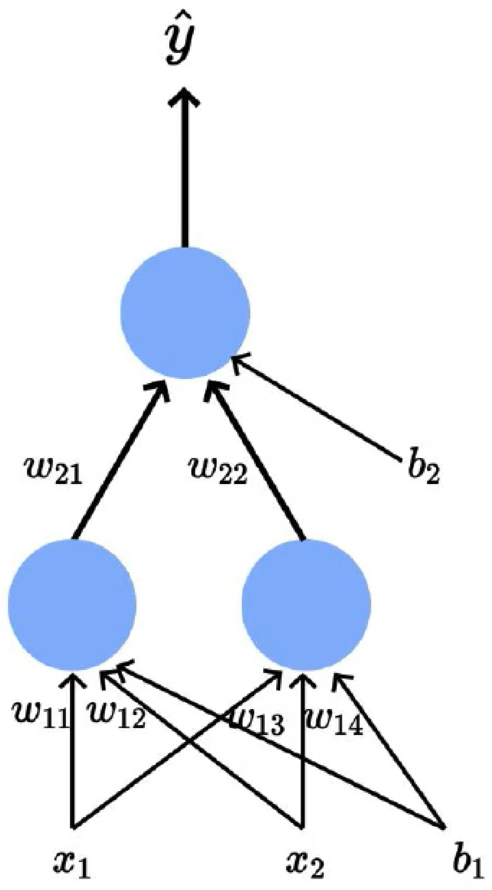
$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

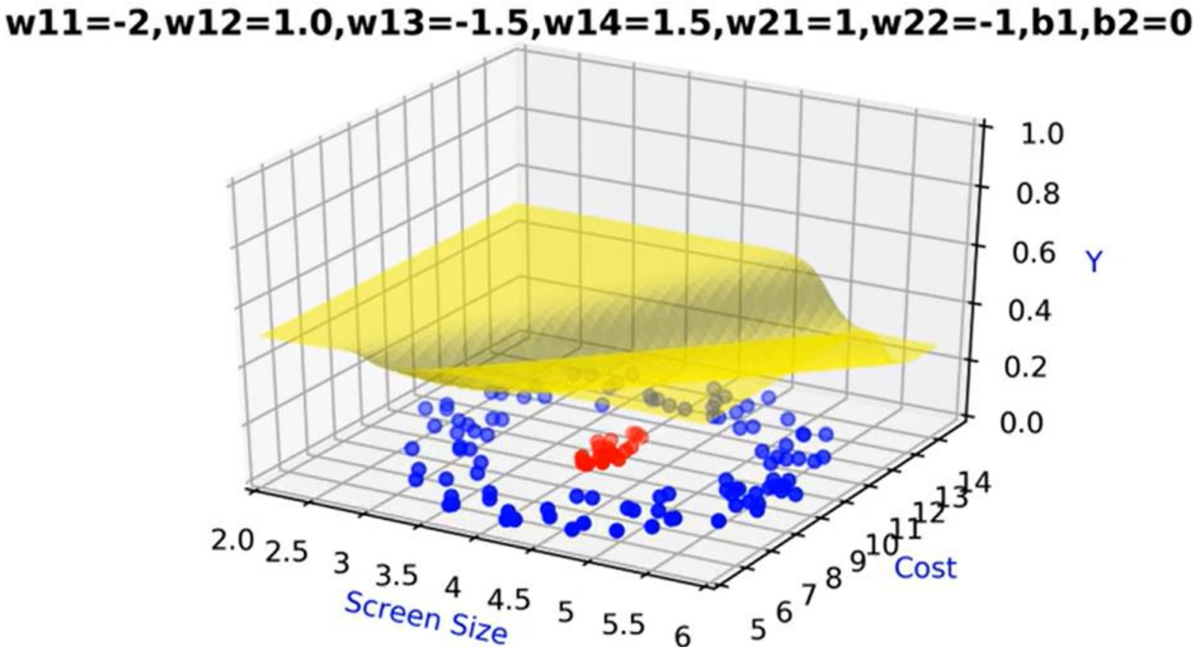
Multiple sigmoid Neurons



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



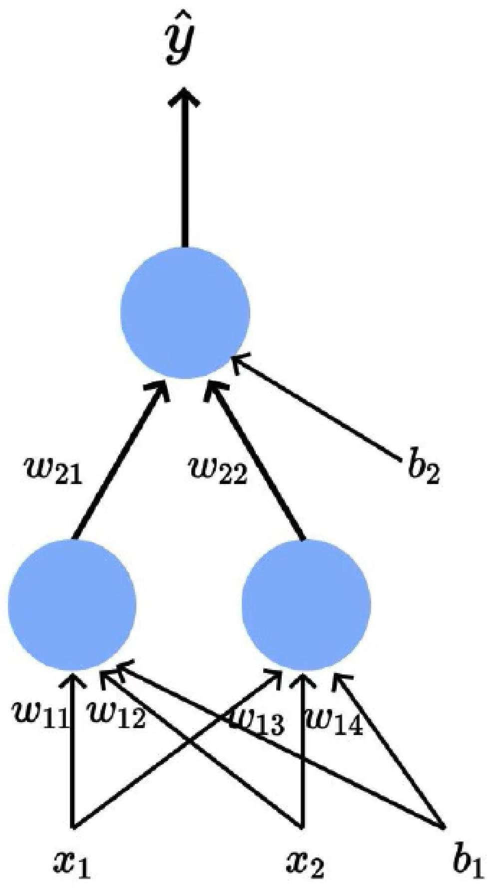
$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

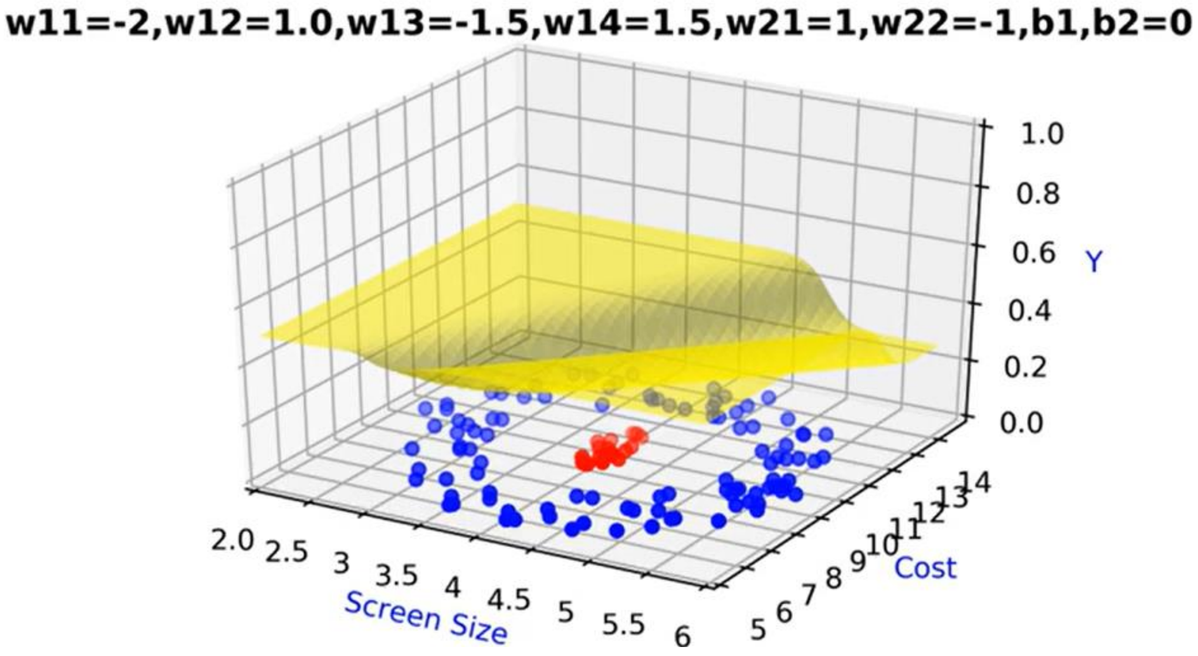
Multiple sigmoid Neurons



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



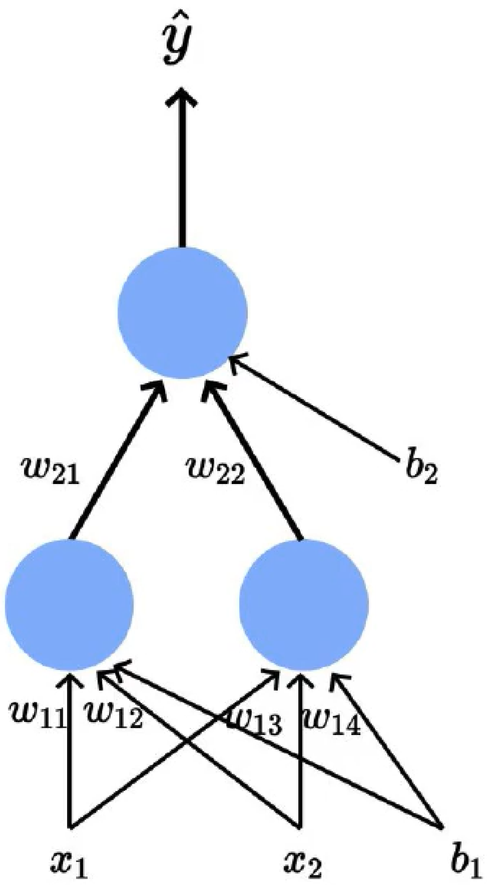
$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

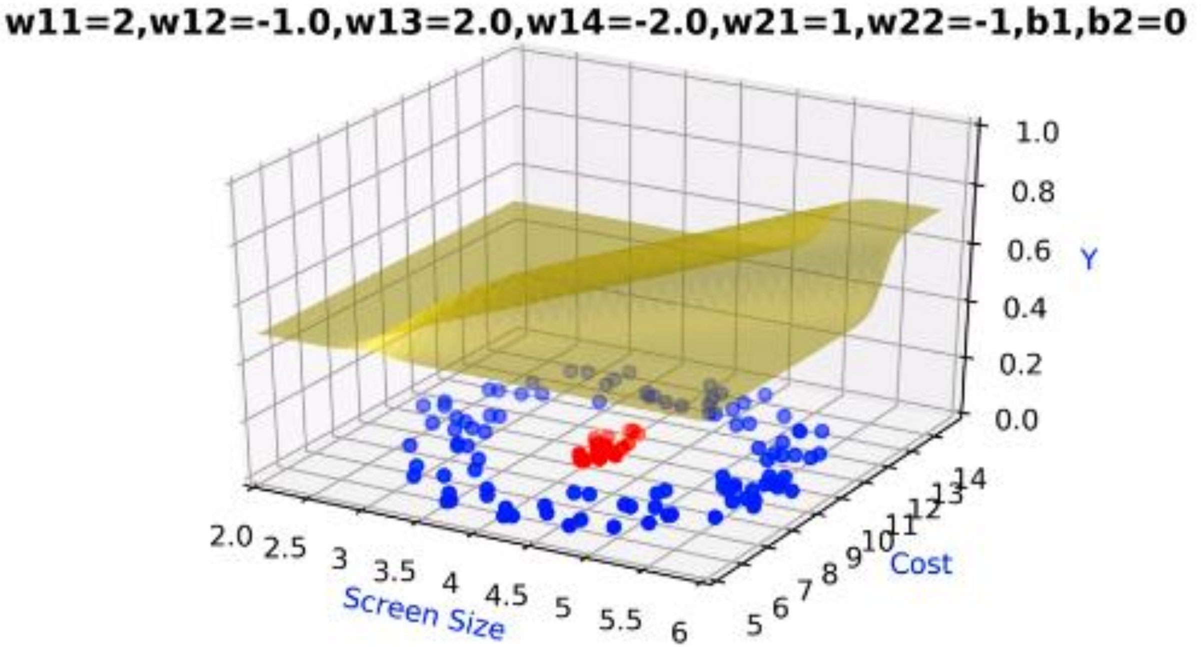
Multi Layer Neural Networks



$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$



$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

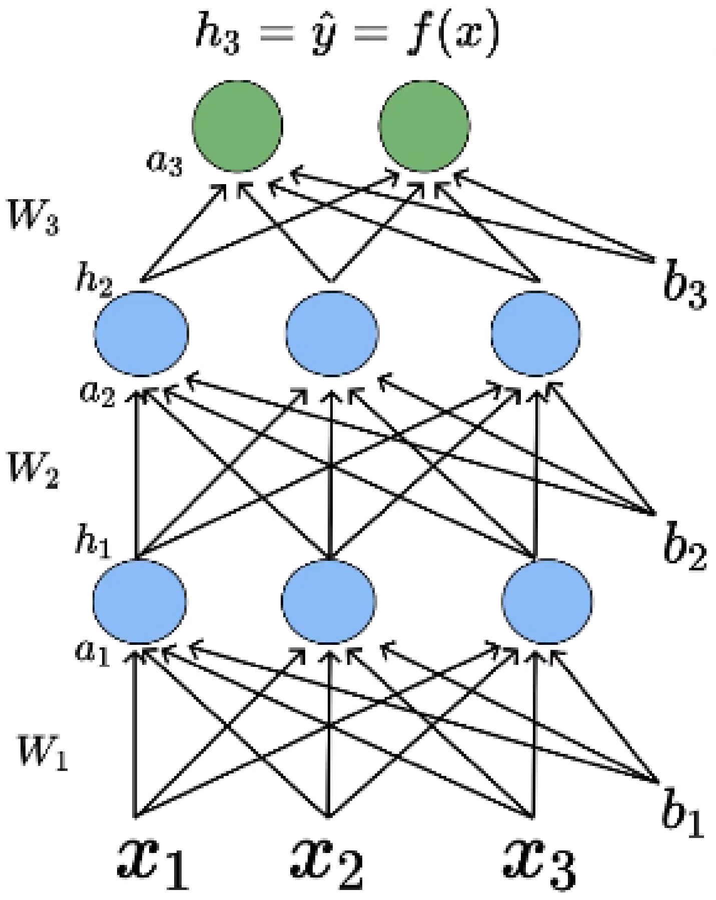
$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

Feed Forward Neural Networks

Understanding the computation

W$_{111}$= W $_{\text{Layer no, Neuron in the next layer ,Input Neuron}}$



$$W_1 = \begin{bmatrix} w_{1\,1\,1} & w_{1\,1\,2} & . & . & . & w_{1\,1\,99} & w_{1\,1\,100} \\ w_{1\,2\,1} & w_{1\,2\,2} & . & . & . & w_{1\,2\,99} & w_{1\,2\,100} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ w_{1\,10\,1} & w_{1\,10\,2} & . & . & . & w_{1\,10\,99} & w_{1\,10\,100} \end{bmatrix} \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{100} \end{bmatrix}$$

$$a_{1\,1} = w_{1\,1\,1} * x_1 + w_{1\,1\,2} * x_2 + w_{1\,1\,3} * x_3 + ... + w_{1\,1\,100} * x_{100} + b_{11}$$

$$a_{1\,2} = w_{1\,2\,1} * x_1 + w_{1\,2\,2} * x_2 + w_{1\,2\,3} * x_3 + ... + w_{1\,2\,100} * x_{100} + b_{12}$$

$$\vdots$$

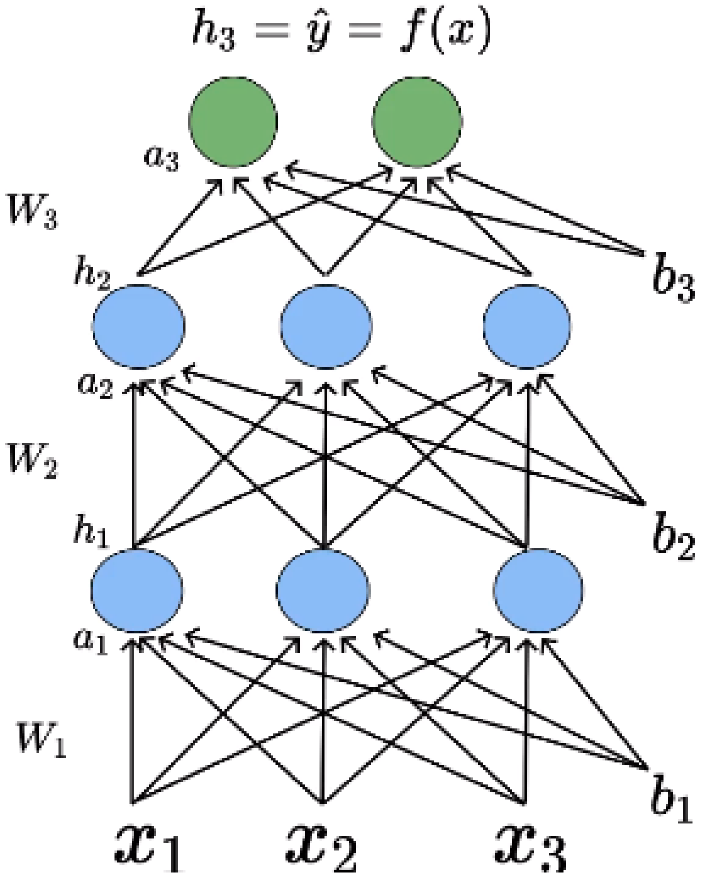
$$a_{1\,10} = w_{1\,10\,1} * x_1 + w_{1\,10\,2} * x_2 + w_{1\,10\,3} * x_3 + ... + w_{1\,10\,100} * x_{100} + b_{1,10}$$

$$a_1 = W_1 * x + b$$

(c) Or

$$h_{11} = g(a_{11}) \qquad h_{12} = g(a_{12}) \quad . \quad . \quad . \quad . \quad h_{1\,10} = g(a_{1\,10})$$

Feed Forward Neural Networks



$$W_1 = \begin{bmatrix} w_{1\,1\,1} & w_{1\,1\,2} & . & . & . & w_{1\,1\,99} & w_{1\,1\,100} \\ w_{1\,2\,1} & w_{1\,2\,2} & . & . & . & w_{1\,2\,99} & w_{1\,2\,100} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ w_{1\,10\,1} & w_{1\,10\,2} & . & . & . & w_{1\,10\,99} & w_{1\,10\,100} \end{bmatrix} \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{100} \end{bmatrix}$$

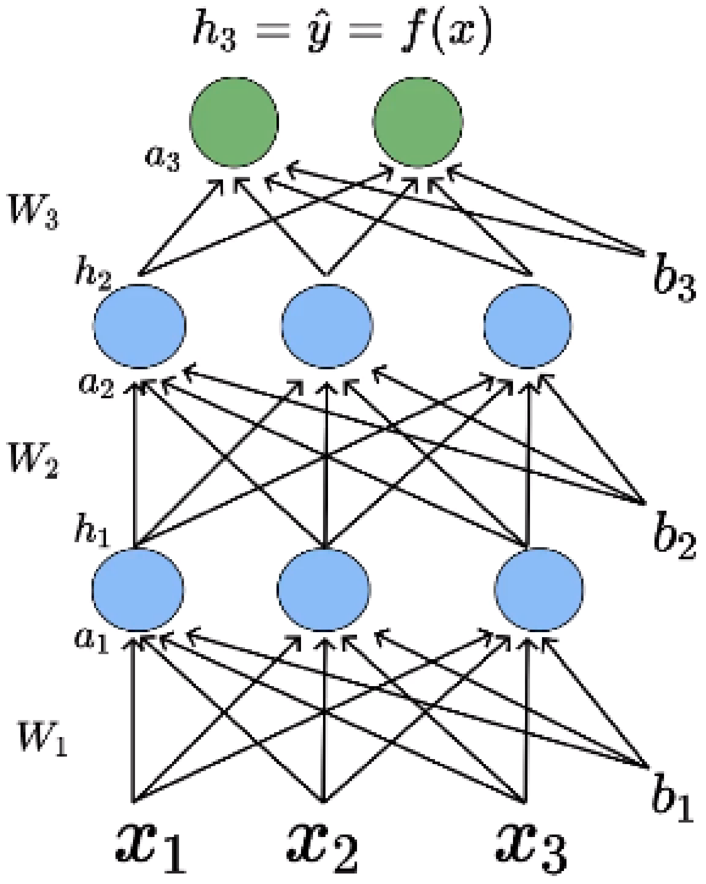
$$\boxed{a_1 = W_1 * x + b}$$

$$h_{11} = g(a_{11}) \qquad h_{12} = g(a_{12}) \quad . \quad . \quad . \quad . \quad h_{1\,10} = g(a_{1\,10})$$

$$\boxed{h_1 = g(a_1)}$$

$$\hat{y} = f(x) = O(W_3 g(W_2 g(W_1 x + b_1) + b_2) + b_3)$$

Feed Forward Neural Networks



- The pre-activation at layer 'i' is given by

  $$a_i(x) = W_i h_{i-1}(x) + b_i$$

- The activation at layer 'i' is given by

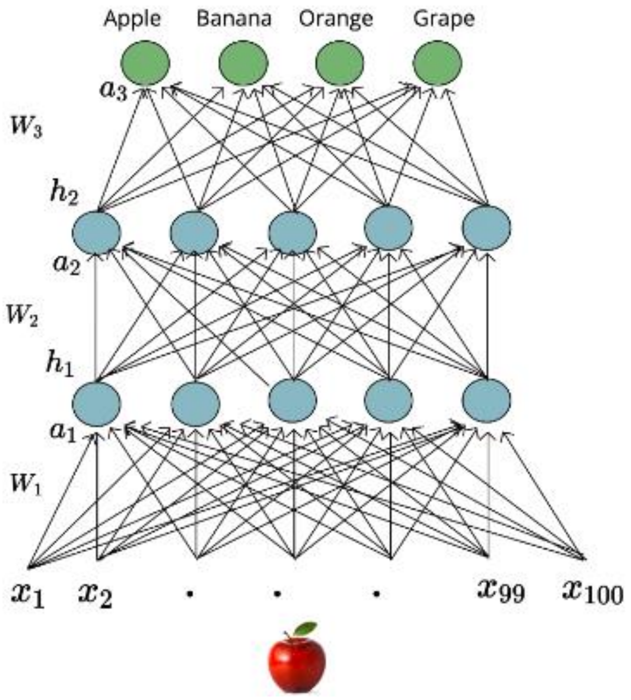
  $$h_i(x) = g(a_i(x))$$

  where 'g' is called as the activation function

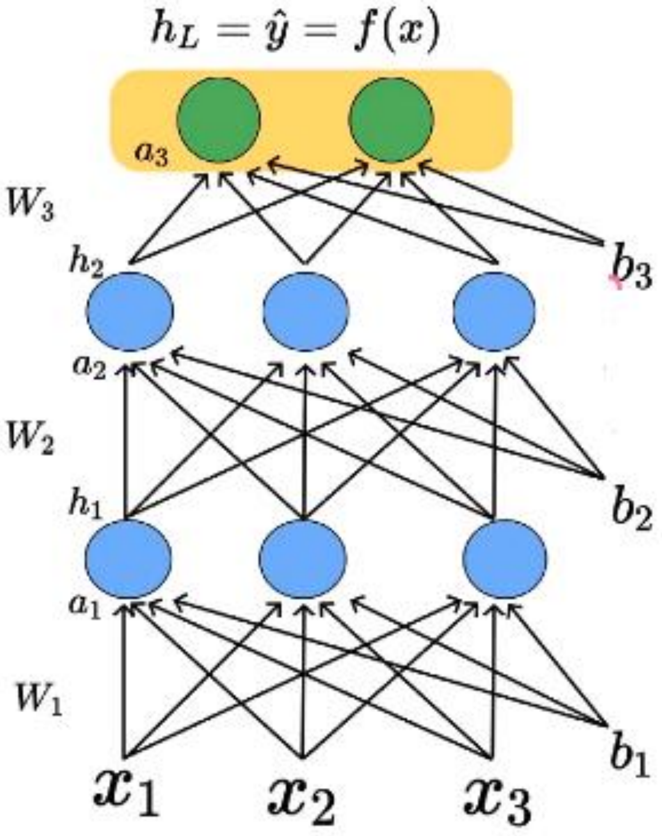
- The activation at output layer 'L' is given by

  $$f(x) = h_L = O(a_L)$$

  where 'O' is called as the output activation function



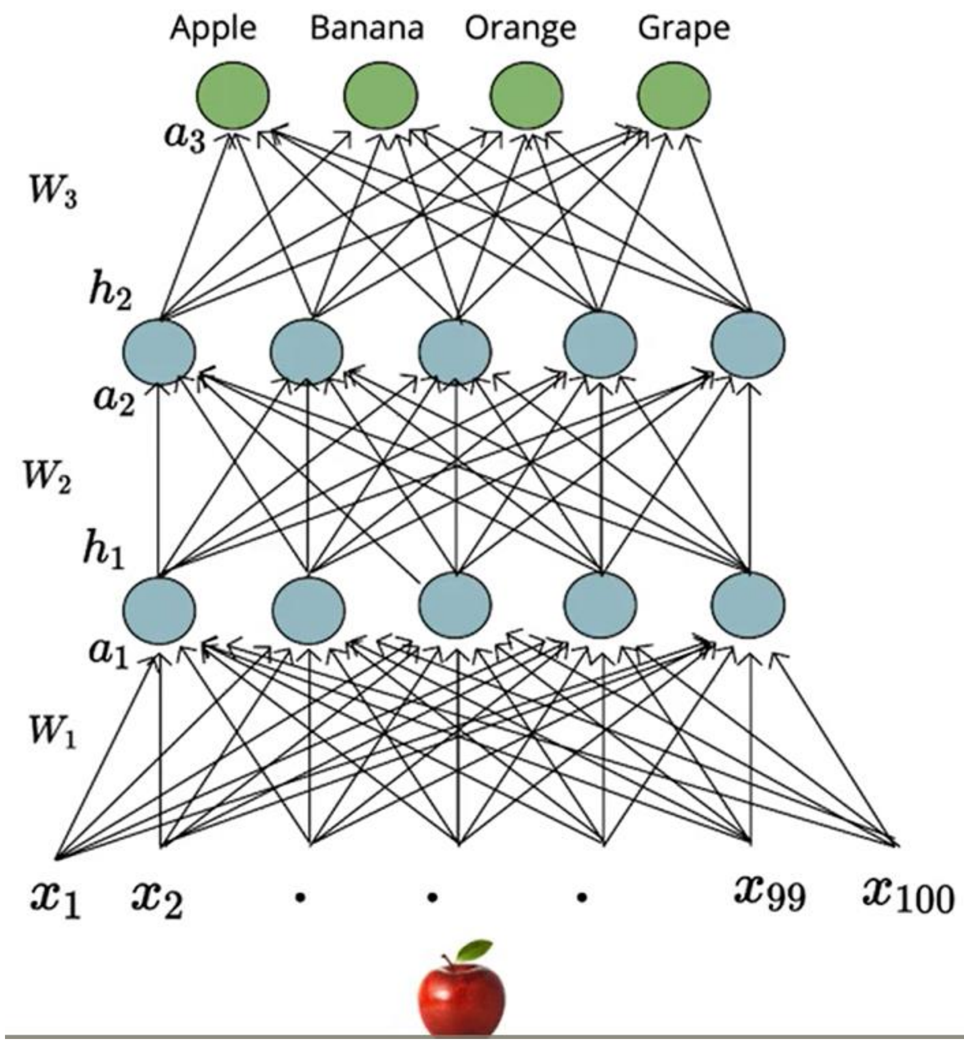
Feed Forward Neural Networks



$$\hat{y} = f(x) = O(W_3 g(W_2 g(W_1 x + b_1) + b_2) + b_3)$$

Output Activation function is chosen depending on the task at hand (can be a softmax, linear)

Is this ok?



$$Say\ a_3 = [\,3\ \ 4\ \ 10\ \ 3\,]$$

Output Activation Function has to be chosen such that output is probability

$$\hat{y}_1 = \frac{3}{(3+4+10+3)} = 0.15$$

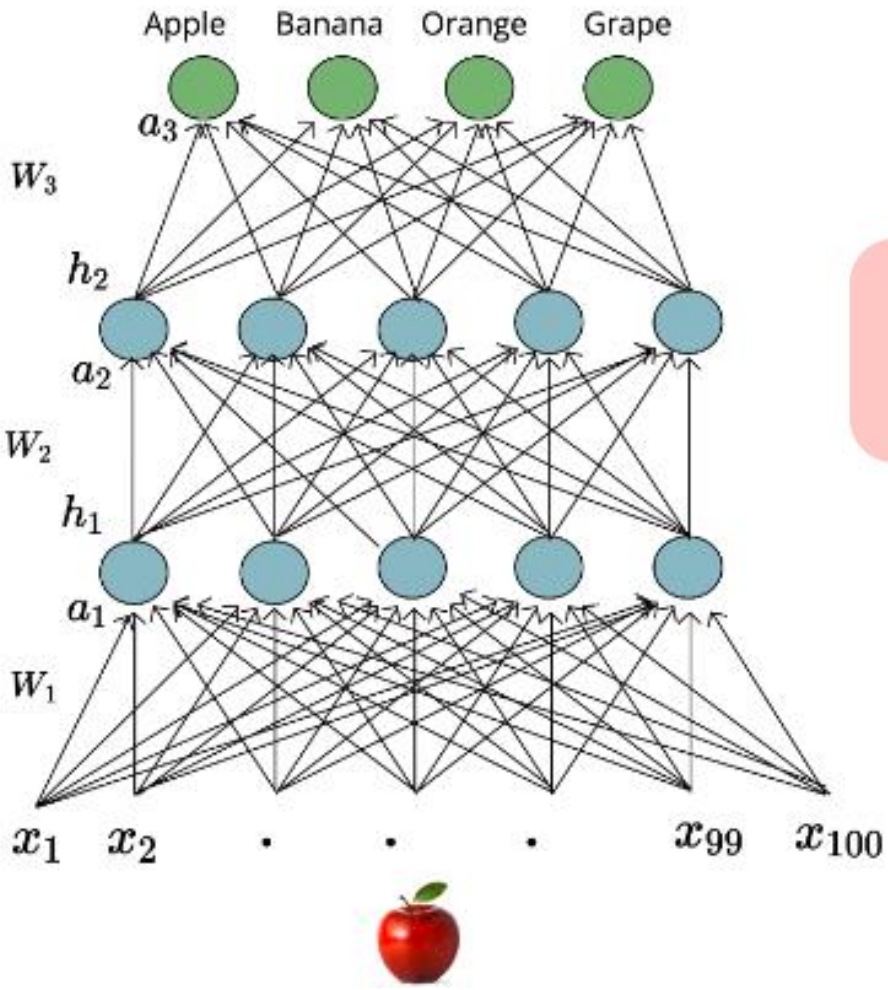
$$\hat{y}_2 = \frac{4}{(3+4+10+3)} = 0.20$$

$$\hat{y}_3 = \frac{10}{(3+4+10+3)} = 0.50$$

$$\hat{y}_4 = \frac{3}{(3+4+10+3)} = 0.15$$

Take each entry and divide by the sum of all entries

Output Layer **Is this ok?-No**



$$\textit{Say for other input } a_3 = [\,7 \quad -2 \;\; 4 \;\; 1\,]$$

Output Activation Function has to be chosen such that output is probability

$$\hat{y}_1 = \frac{7}{(7 + (-2) + 4 + 1)} = 0.70$$

$$\hat{y}_2 = \frac{-2}{(7 + (-2) + 4 + 1)} = -0.20$$

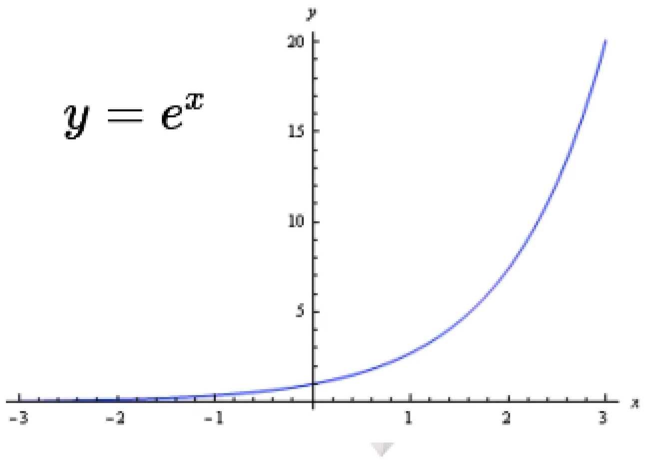
$$\hat{y}_3 = \frac{4}{(7 + (-2) + 4 + 1)} = 0.40$$

$$\hat{y}_4 = \frac{1}{(7 + (-2) + 4 + 1)} = 0.10$$

Softmax

Softmax is a kind of `activation` function with the speciality of output summing to 1.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad for\ i = 1.....k$$

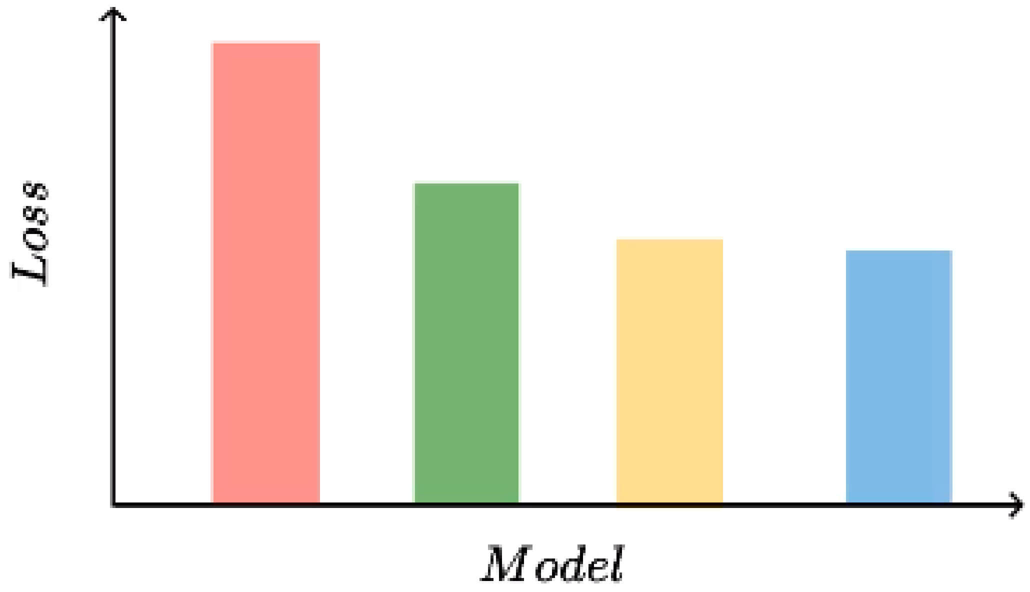


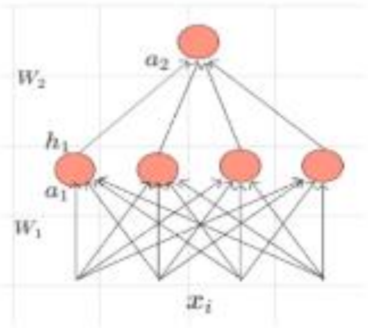
$$h = [\ h_1\ \ h_2\ \ h_3\ \ h_4\ ]$$

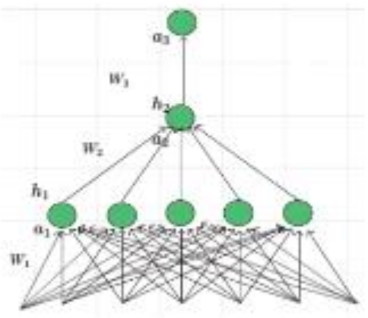
$$softmax(h) = [softmax(h_1)\ \ softmax(h_2)\ \ softmax(h_3)\ \ softmax(h_4)]$$

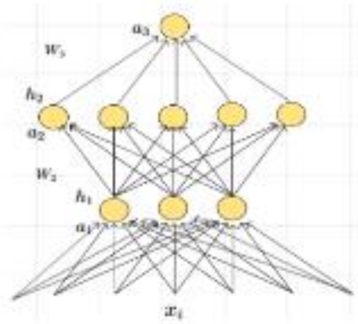
$$softmax(h) = \left[\frac{e^{h_1}}{\sum_{j=1}^{4} e^{h_j}} \quad \frac{e^{h_2}}{\sum_{j=1}^{4} e^{h_j}} \quad \frac{e^{h_3}}{\sum_{j=1}^{4} e^{h_j}} \quad \frac{e^{h_4}}{\sum_{j=1}^{4} e^{h_j}} \right]$$

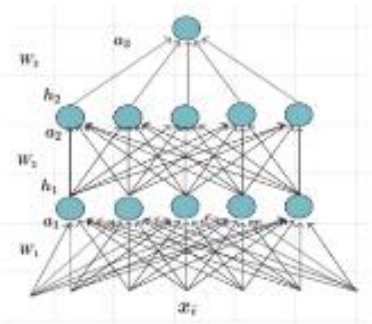
Different network configurations











Learning Algorithm- backpropagation

Initialise w, b

Iterate over data:

$compute\ \hat{y}$

$compute\ \mathscr{L}(w,b)$

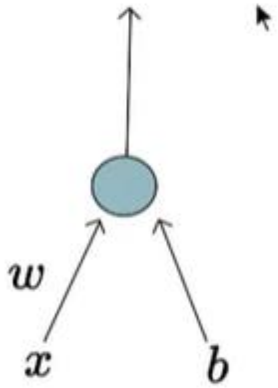
$w_{111} = w_{111} - \eta\Delta w_{111}$

$w_{112} = w_{112} - \eta\Delta w_{112}$

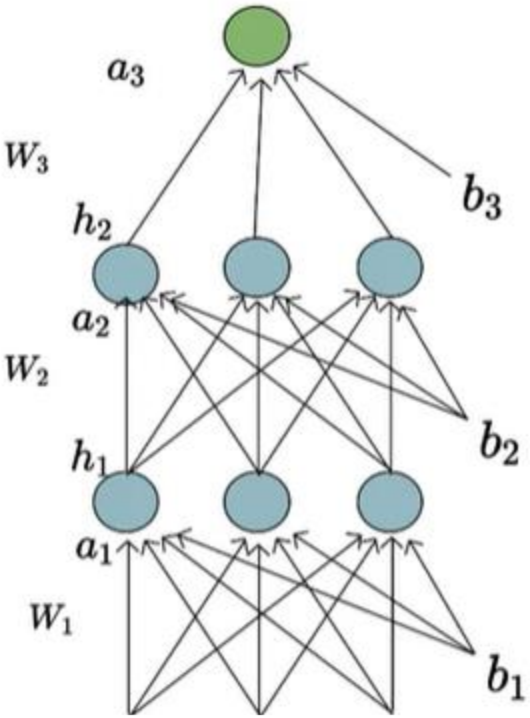
....

$w_{313} = w_{313} - \eta\Delta w_{313}$

till satisfied



$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$



$$\mathscr{L} = \frac{1}{5}\sum_{i=1}^{i=5}(f(x_i) - y_i)^2$$

$$\frac{\partial \mathscr{L}}{\partial w} = \frac{\partial}{\partial w}\left[\frac{1}{5}\sum_{i=1}^{i=5}(f(x_i) - y_i)\right]^2$$

$$\Delta w = \frac{\partial \mathscr{L}}{\partial w} = \frac{1}{5}\sum_{i=1}^{i=5}\frac{\partial}{\partial w}(f(x_i) - y_i)^2$$

Calculation of Δw

$$\nabla w = \frac{\partial}{\partial w}[\frac{1}{2} * (f(x) - y)^2]$$

$$= \frac{1}{2} * [2 * (f(x) - y) * \frac{\partial}{\partial w}(f(x) - y)]$$

$$= (f(x) - y) * \frac{\partial}{\partial w}(f(x))$$

$$= (f(x) - y) * \frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= (f(x) - y) * f(x) * (1 - f(x)) * x$$

$$\frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= \frac{-1}{(1+e^{-(wx+b)})^2} \frac{\partial}{\partial w}(e^{-(wx+b)}))$$

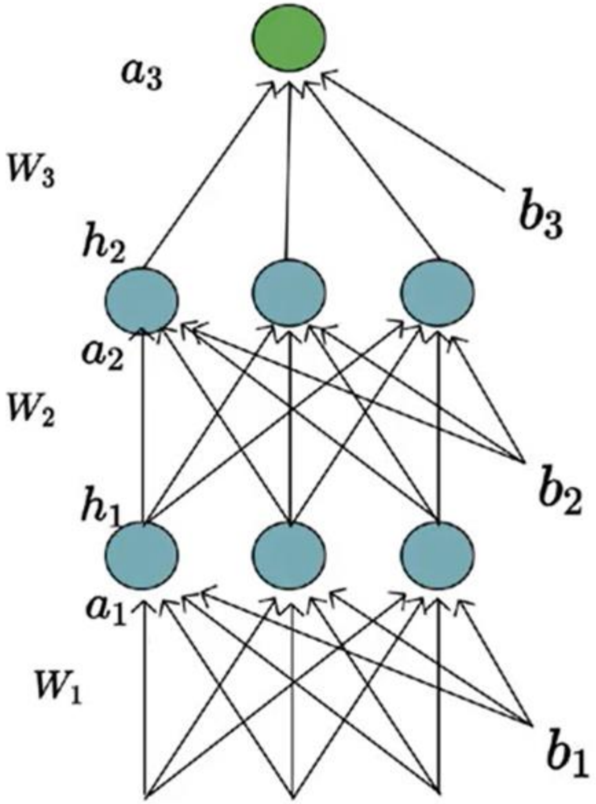
$$= \frac{-1}{(1+e^{-(wx+b)})^2} * (e^{-(wx+b)}) \frac{\partial}{\partial w}(-(wx + b)))$$

$$= \frac{-1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (-x)$$

$$= \frac{1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (x)$$

$$= f(x) * (1 - f(x)) * x$$

Partial derivative , Gradient



$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

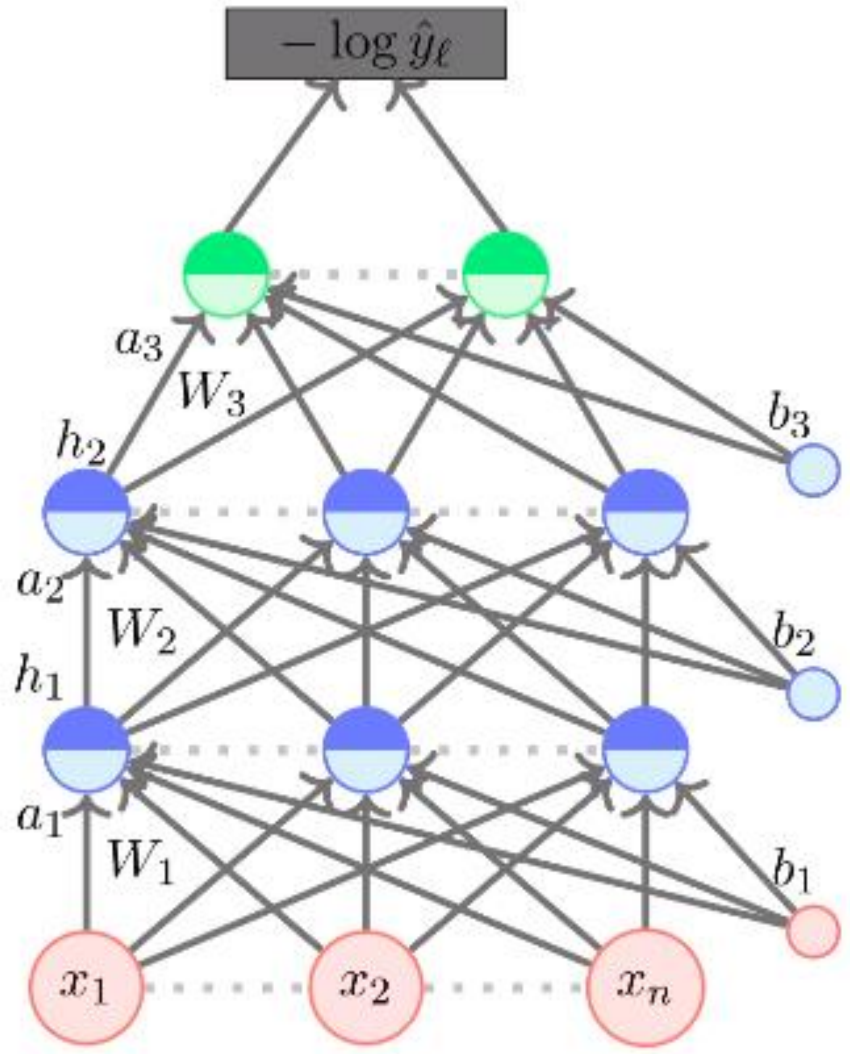
$$\begin{bmatrix}
\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{11n}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{211}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{21n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,11}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,1k}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,1k}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{11}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L1}} \\
\frac{\partial \mathscr{L}(\theta)}{\partial W_{121}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{12n}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{221}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{22n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,21}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,2k}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,2k}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{12}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L2}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \mathscr{L}(\theta)}{\partial W_{1n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{1nn}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{2n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{2nn}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,nk}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,nk}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{1n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{Lk}}
\end{bmatrix}$$

The partial derivative notation is used **to specify the derivative of a function of more than one variable with respect to one of its variables**.

The gradient of a function f, denoted as ∇f , is **the collection of all its partial derivatives into a vector**.

Backpropagation

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}}\,\frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}}\;\underbrace{\frac{\partial a_3}{\partial h_2}\,\frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}}\;\underbrace{\frac{\partial a_2}{\partial h_1}\,\frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}}\;\underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$



$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

The partial derivative notation is used **to specify the derivative of a function of more than one variable with respect to one of its variables**.

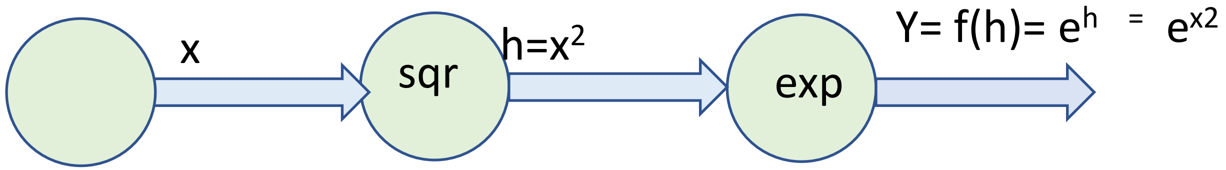
The gradient of a function f, denoted as ∇f , is **the collection of all its partial derivatives into a vector**.

Calculus basics - Chain rule

The chain rule is **a method for finding the derivative of composite functions, or functions that are made by combining one or more functions**.

$$\frac{de^x}{dx} = e^x \qquad \frac{dx^2}{dx} = 2x \qquad \frac{d(1/x)}{dx} = -\frac{1}{x^2}$$

$$\frac{de^{x^2}}{dx} = \frac{de^{x^2}}{dx^2} \cdot \frac{dx^2}{dx} = \frac{de^z}{dz} \cdot \frac{dx^2}{dx} = (e^z).(2x) = (e^{x^2}).(2x) = 2xe^{x^2}$$



$h=f(x)$

$Y=f(h)= e^h$

$$\frac{dy}{dx} = \frac{dy}{dh}\,\frac{dh}{dx} = \frac{de^h}{dh}\,\frac{dx^2}{dx} = e^h\, 2x = 2x\, e^{x2}$$

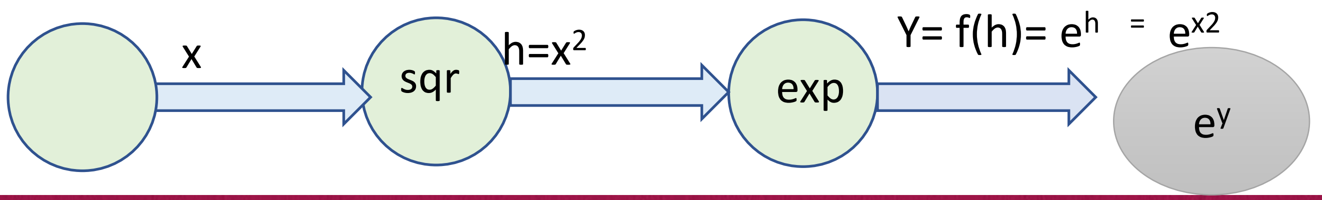
Chain rule

$$\boxed{\frac{de^{e^{x^2}}}{dx} = \frac{de^{e^{x^2}}}{de^{x^2}} \cdot \frac{de^{x^2}}{dx^2} \cdot \frac{dx^2}{dx}}$$

$$\frac{de^x}{dx} = e^x \qquad \frac{dx^2}{dx} = 2x \qquad \frac{d(1/x)}{dx} = -\frac{1}{x^2}$$

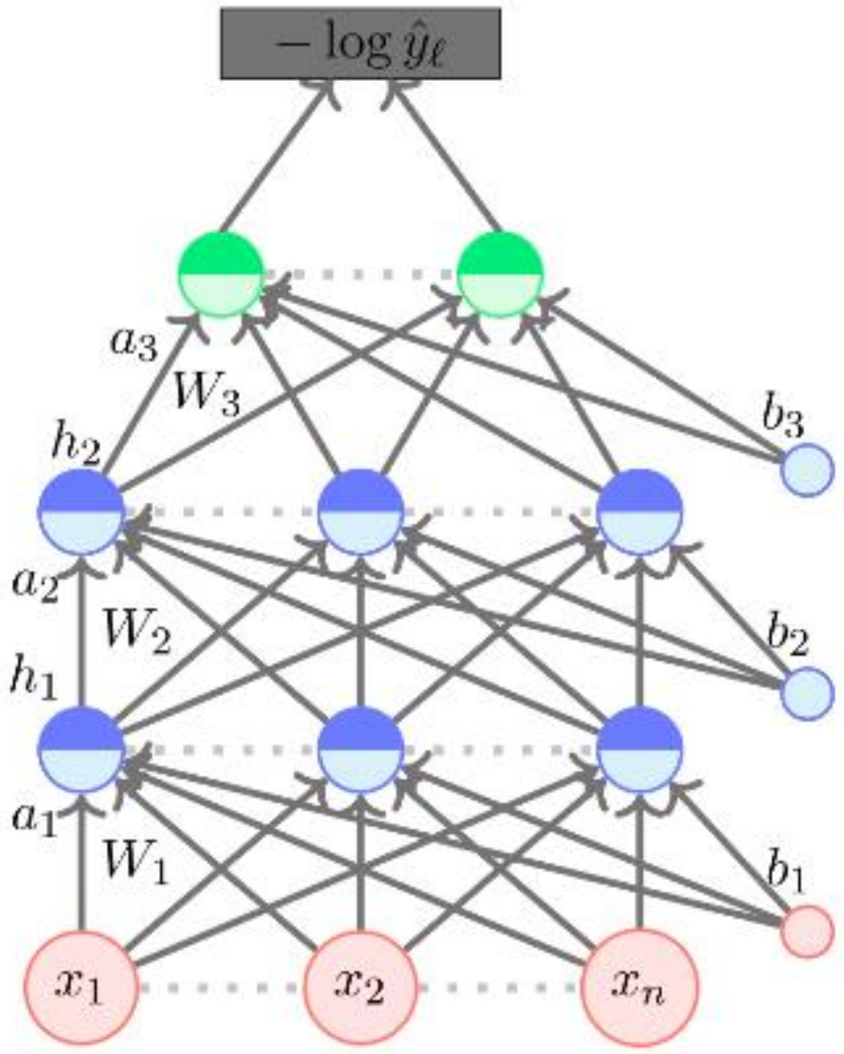
$$\frac{de^{x^2}}{dx} = \frac{de^{x^2}}{dx^2} \cdot \frac{dx^2}{dx} = \frac{de^z}{dz} \cdot \frac{dx^2}{dx} = (e^z).(2x) = (e^{x^2}).(2x) = 2xe^{x^2}$$

$$\frac{de^{e^{x^2}}}{dx} = \frac{de^{e^{x^2}}}{de^{x^2}} \cdot \frac{de^{x^2}}{dx} = \frac{de^z}{dz} \cdot \frac{de^{x^2}}{dx} = (e^z).(2xe^{x^2}) = (e^{e^{x^2}}).(2xe^{x^2}) = 2xe^{x^2}e^{e^{x^2}}$$



Backpropagation

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$



$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

The partial derivative notation is used **to specify the derivative of a function of more than one variable with respect to one of its variables**.

The gradient of a function f, denoted as ∇f , is **the collection of all its partial derivatives into a vector**.

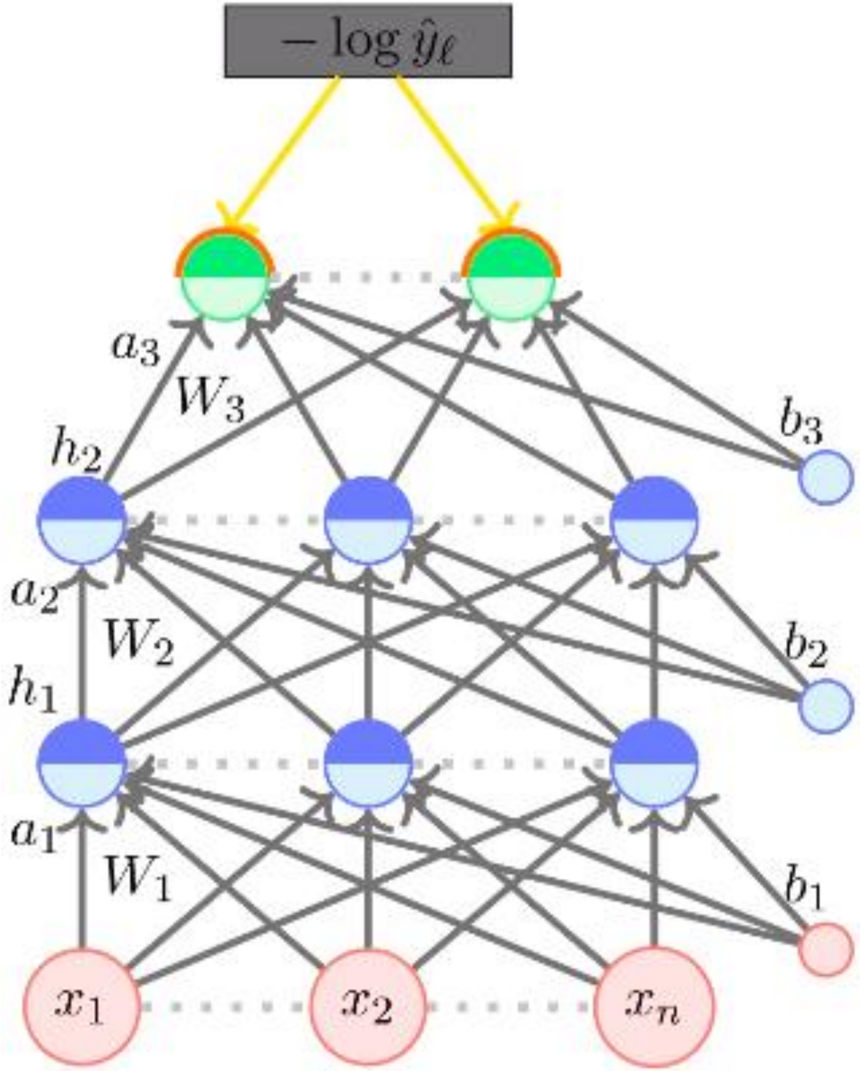
Backpropagation

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\substack{\text{Talk to the}\\ \text{weight directly}}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial a_3}}_{\substack{\text{Talk to the}\\ \text{output layer}}} \underbrace{\frac{\partial a_3}{\partial h_2}\frac{\partial h_2}{\partial a_2}}_{\substack{\text{Talk to the}\\ \text{previous hidden}\\ \text{layer}}} \underbrace{\frac{\partial a_2}{\partial h_1}\frac{\partial h_1}{\partial a_1}}_{\substack{\text{Talk to the}\\ \text{previous}\\ \text{hidden layer}}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\substack{\text{and now}\\ \text{talk to}\\ \text{the}\\ \text{weights}}}$$

$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

The partial derivative notation is used **to specify the derivative of a function of more than one variable with respect to one of its variables**.

The gradient of a function f, denoted as ∇f , is **the collection of all its partial derivatives into a vector**.



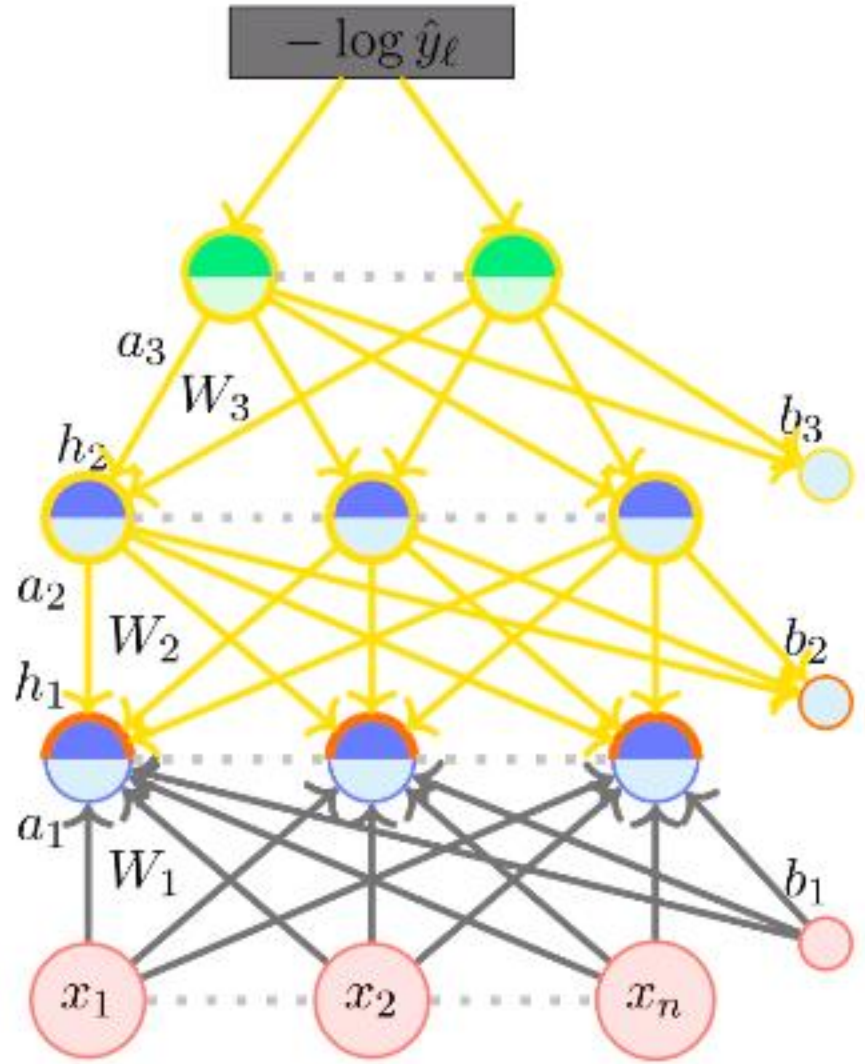
Backpropagation

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$

$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

The partial derivative notation is used **to specify the derivative of a function of more than one variable with respect to one of its variables**.

The gradient of a function f, denoted as ∇f , is **the collection of all its partial derivatives into a vector**.



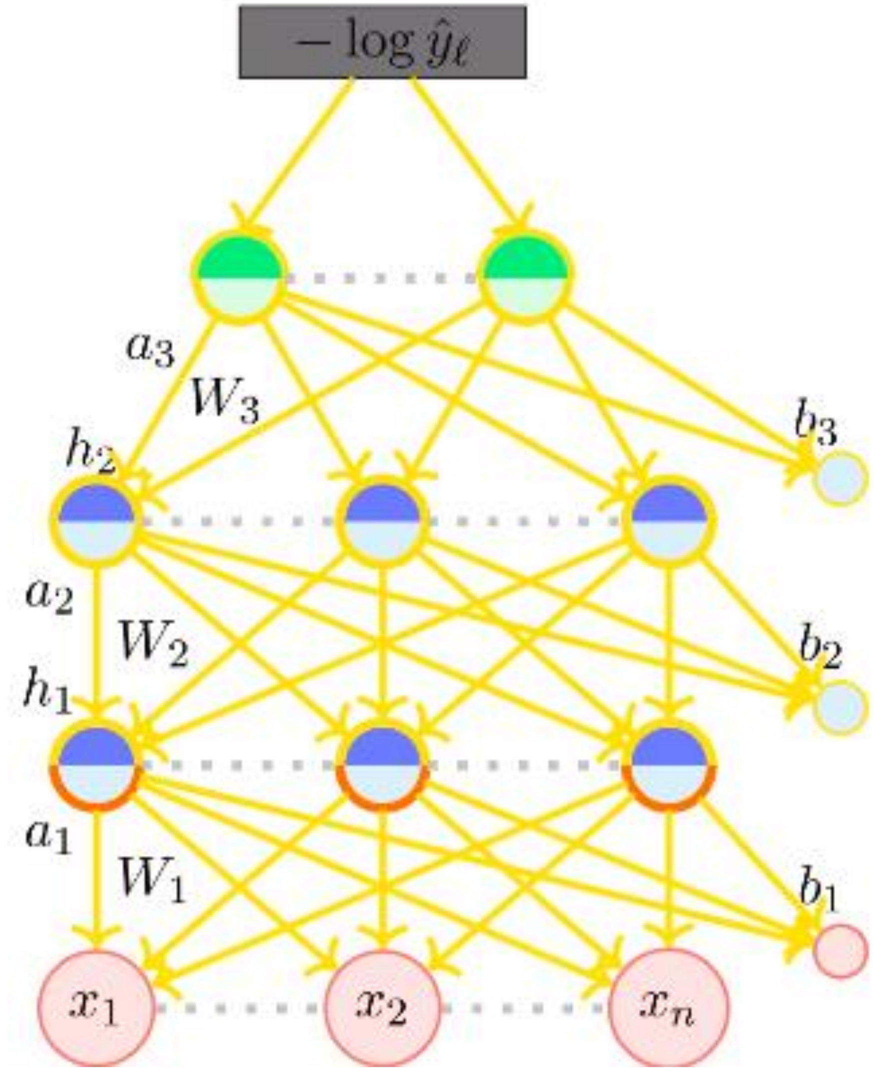
Backpropagation

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$

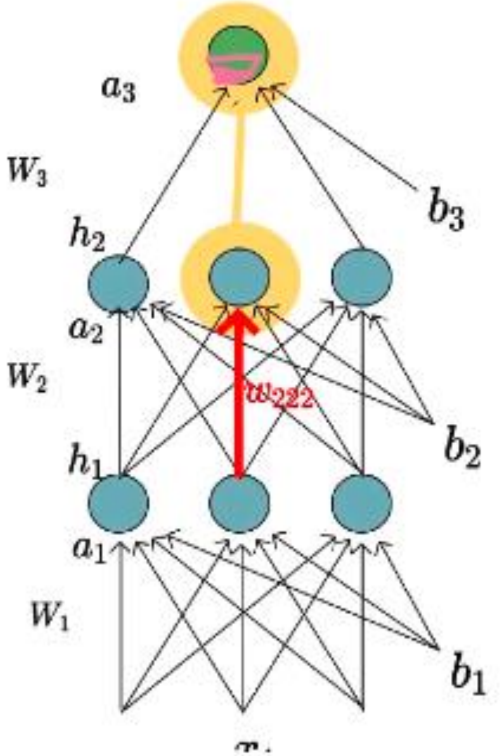
$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

The partial derivative notation is used **to specify the derivative of a function of more than one variable with respect to one of its variables**.

The gradient of a function f, denoted as ∇f , is **the collection of all its partial derivatives into a vector**.



Learning algorithm- Back propagation

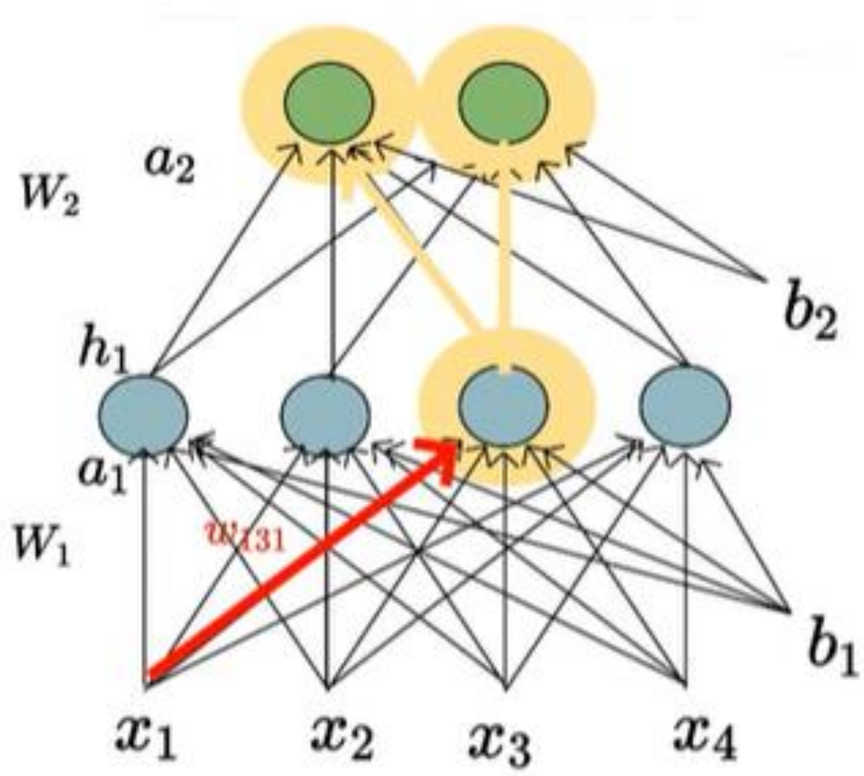


- Let us focus on the highlighted weight (w_{222})
- To learn this weight, we have to compute partial derivative w.r.t loss function

$$(w_{222})_{t+1} = (w_{222})_t - \eta * \left(\frac{\partial L}{\partial w_{222}}\right)$$

$$\begin{aligned}
\frac{\partial L}{\partial w_{222}} &= \left(\frac{\partial L}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right) \\
&= \left(\frac{\partial L}{\partial h_{22}}\right).\left(\frac{\partial h_{22}}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right) \\
&= \left(\frac{\partial L}{\partial a_{31}}\right).\left(\frac{\partial a_{31}}{\partial h_{22}}\right).\left(\frac{\partial h_{22}}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right) \\
&= \left(\frac{\partial L}{\partial \hat{y}}\right).\left(\frac{\partial \hat{y}}{\partial a_{31}}\right).\left(\frac{\partial a_{31}}{\partial h_{22}}\right).\left(\frac{\partial h_{22}}{\partial a_{22}}\right).\left(\frac{\partial a_{22}}{\partial w_{222}}\right)
\end{aligned}$$

Multiple paths



- There are 2 different paths connecting w131 with loss function
- Consider all those paths through which gradient is flowing back
- Sum up the gradients along all these paths(2 here)
- ie Apply independent chain rules to all those multiple paths and sum up the derivatives across all these paths and get the total derivative of the loss function w.r.t w131

Learning Algorithm- backpropagation

Initialise w, b

Iterate over data:

$compute\ \hat{y}$

$compute\ \mathscr{L}(w, b)$

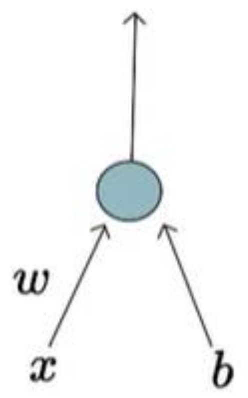
$w_{111} = w_{111} - \eta\Delta w_{111}$

$w_{112} = w_{112} - \eta\Delta w_{112}$

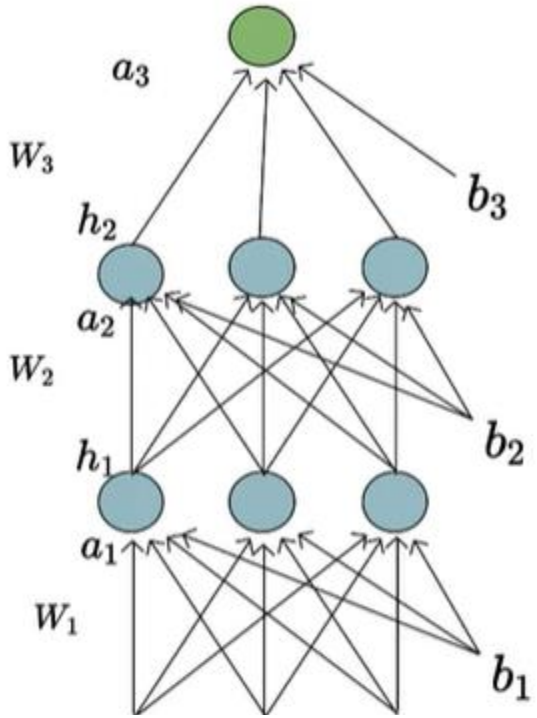
....

$w_{313} = w_{313} - \eta\Delta w_{313}$

till satisfied



$$\Delta w_t = \frac{\partial \mathscr{L}(w,b)}{\partial w}$$

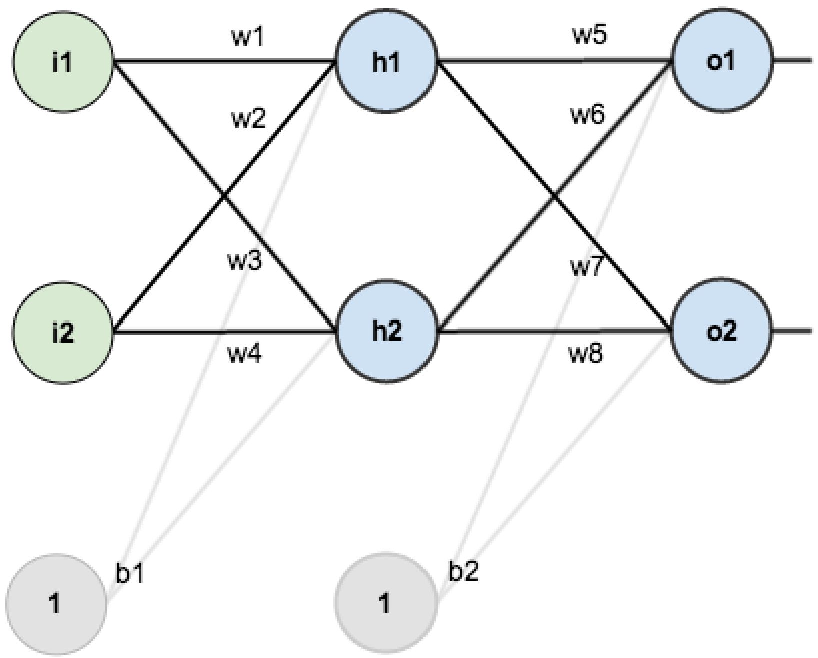


$$\mathscr{L} = \frac{1}{5}\sum_{i=1}^{i=5}(f(x_i) - y_i)^2$$

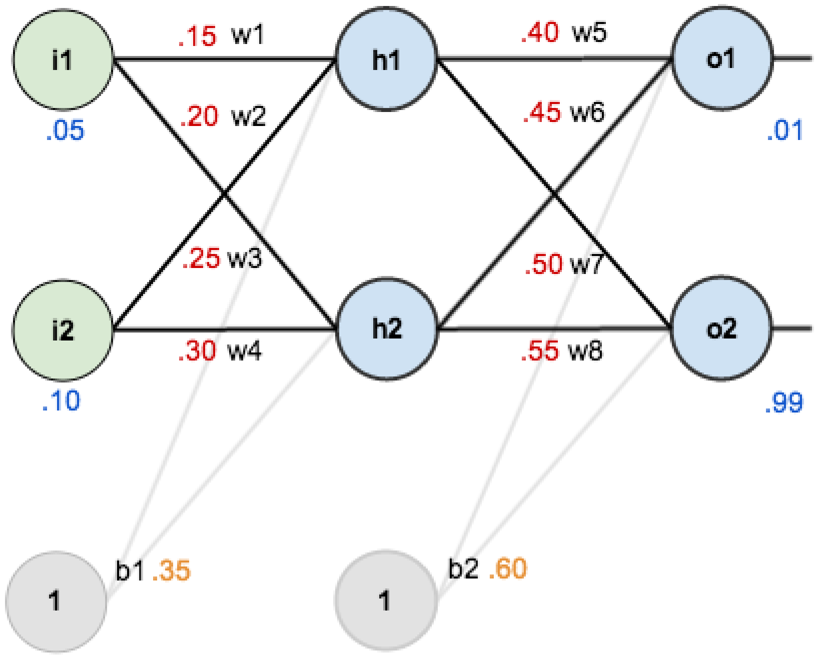
$$\frac{\partial \mathscr{L}}{\partial w} = \frac{\partial}{\partial w}\left[\frac{1}{5}\sum_{i=1}^{i=5}(f(x_i) - y_i)\right]^2$$

$$\Delta w = \frac{\partial \mathscr{L}}{\partial w} = \frac{1}{5}\sum_{i=1}^{i=5}\frac{\partial}{\partial w}(f(x_i) - y_i)^2$$

Practice Problem- Back Propagation

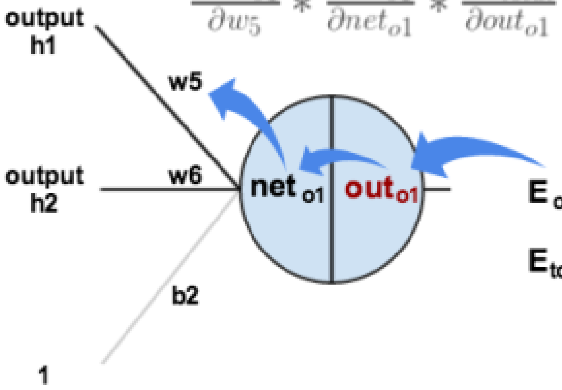


The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.



In order to have some numbers to work with, here are the initial weights, the biases, and training inputs/outputs:

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$$E_{o1} = \tfrac{1}{2}(target_{o1} - out_{o1})^2$$

$$E_{total} = E_{o1} + E_{o2}$$

Given a single training set: given inputs 0.05 and 0.10, we want the neural network to output 0.01 and 0.99.

Forward Pass

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$

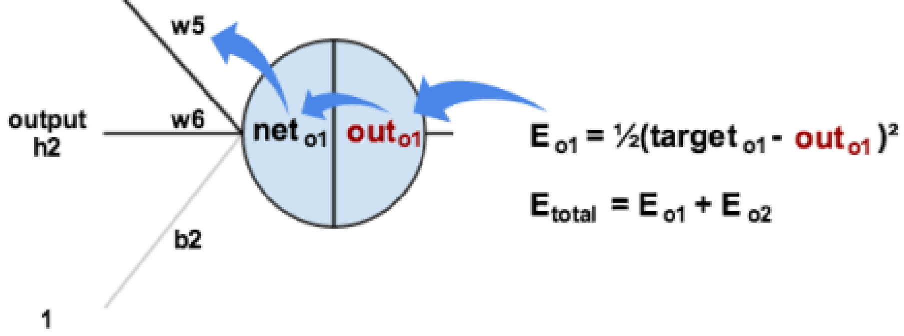
$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



Notation as per previous slides

Net O1= a1

Outo1=h1

- We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.
- Here's the output for o_1:

Here's the output for o_1:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$

Calculating the Total Error

- We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

The Backwards Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$

$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$

Finally, how much does the total net input of $o1$ change with respect to w_5?

$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$

Putting it all together:

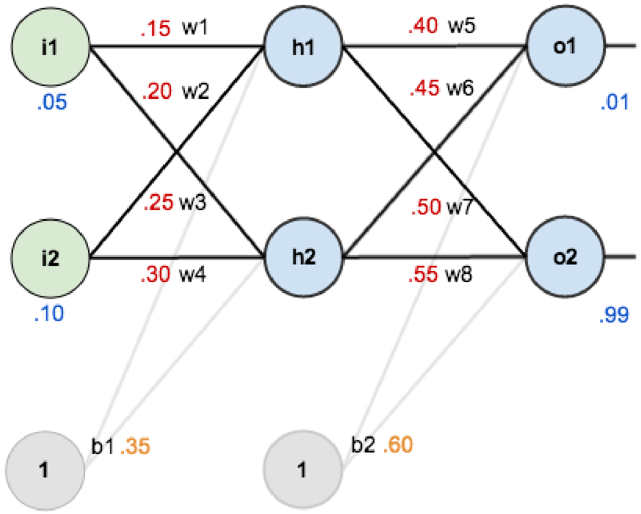
$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$

$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$



$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

Hidden Layer

Next, we'll continue the backwards pass by calculating new values for w_1, w_2, w_3, and w_4.

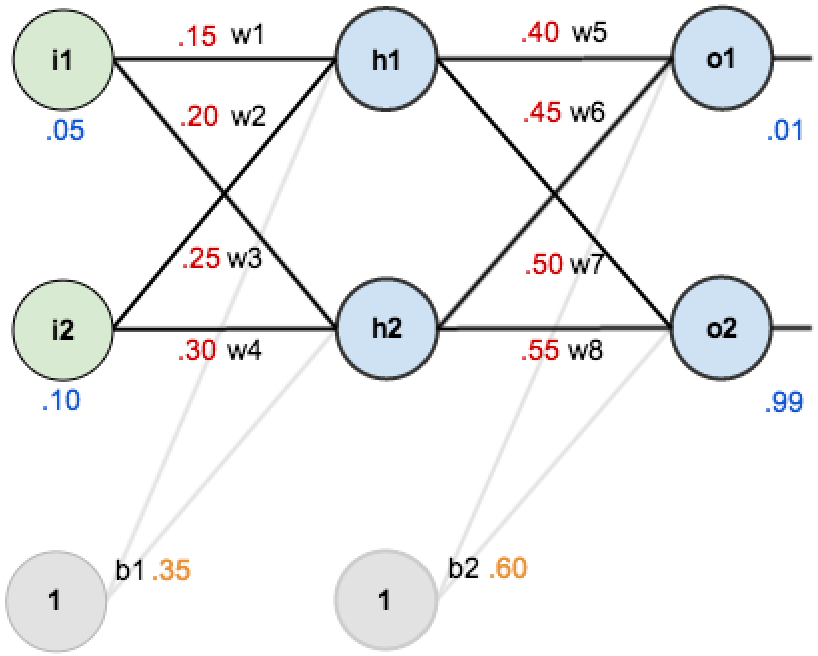
Big picture, here's what we need to figure out:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

We can now update w_1:

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Repeating this for w_2, w_3, and w_4



$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

Cross Entropy Loss Function

Cross Entropy Loss- binary class

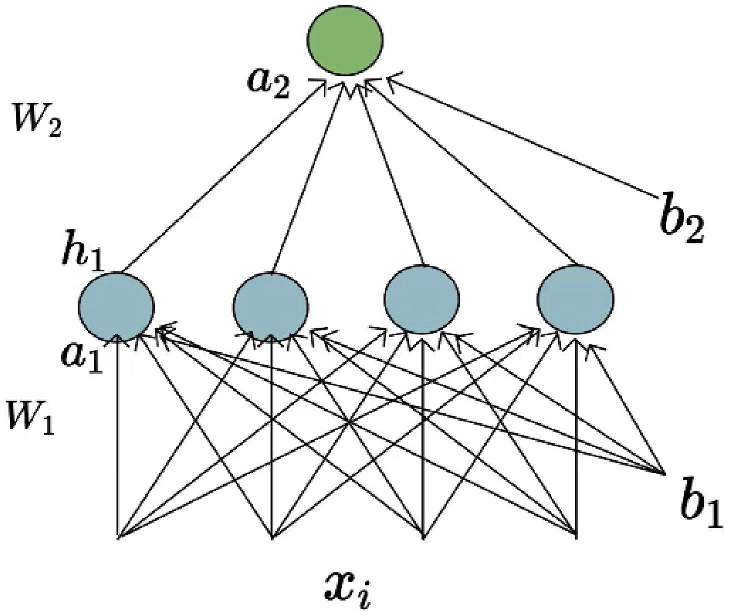
$$L(\Theta) = \begin{cases} -log(\hat{y}) & \text{if } y = 1 \\ -log(1-\hat{y}) & \text{if } y = 0 \end{cases}$$

Cross Entropy Loss: **Multiclass**

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

- Also called **logarithmic loss**, **log loss** or **logistic loss**.
- Each predicted class probability is compared to the actual class desired output 0 or 1 and a score/loss is calculated that **penalizes the probability based on how far it is from the actual expected value.**
- The **penalty is logarithmic in nature** yielding a large score for large differences close to 1 and small score for small differences tending to 0.
- Cross-entropy loss is used when adjusting model weights during training. **The aim is to minimize the loss**, i.e, the smaller the loss the better the model. A **perfect model has a cross-entropy loss of 0**.

Loss function for binary class classification



$$x = [\ -0.6 \quad -0.6 \quad 0.2 \quad 0.3\ ] \qquad y = 0$$

Output :

$$a_1 = W_1 * x + b_1 = [\ 0.01 \quad 0.71 \quad 0.42 \quad 0.63\ ]$$

$$h_1 = sigmoid(a_1) = [\ 0.50 \quad 0.67 \quad 0.60 \quad 0.65\ ]$$

$$a_2 = W_2 * h_1 + b_2 = 0.921$$

$$\hat{y} = sigmoid(a_2) = 0.7152$$

$$b = [\ 0.5 \quad 0.3\ ]$$

$$W_1 = \begin{bmatrix} 0.9 & 0.2 & 0.4 & 0.3 \\ -0.5 & 0.4 & 0.3 & 0.3 \\ 0.1 & 0.1 & -0.1 & 0.2 \\ -0.2 & 0.5 & 0.5 & 0.7 \end{bmatrix}$$

$$W_2 = [0.5 \quad 0.8 \quad -0.6 \quad 0.3]$$

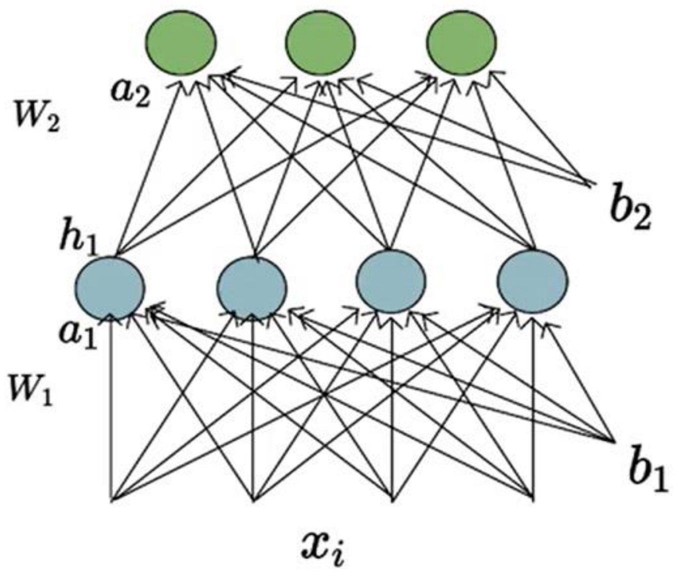
Cross Entropy Loss:

$$L(\Theta) = \begin{cases} -log(\hat{y}) & \text{if } y = 1 \\ -log(1-\hat{y}) & \text{if } y = 0 \end{cases}$$

$$L(\Theta) = -1 * \log(1 - 0.7152)$$

$$= 1.2560$$

Loss function for multi class classification



$$b = [\ 0 \quad 0\ ]$$

$$W_1 = \begin{bmatrix} 0.1 & 0.3 & 0.8 & -0.4 \\ -0.3 & -0.2 & 0.5 & 0.5 \\ -0.3 & 0.1 & 0.5 & 0.4 \\ 0.2 & 0.5 & -0.9 & 0.7 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.3 & 0.8 & -0.2 & -0.4 \\ 0.5 & -0.2 & -0.3 & 0.5 \end{bmatrix}$$

$$x = [\ 0.6 \quad 0.4 \quad 0.6 \quad 0.1\ ] \qquad y = [\ 0 \quad 0 \quad 1\ ]$$

Output :

$$a_1 = W_1 * x + b_1 = [\ 0.62 \quad 0.09 \quad 0.2 \quad -0.15\ ]$$

$$h_1 = sigmoid(a_1) = [\ 0.65 \quad 0.52 \quad 0.55 \quad 0.46\ ]$$

$$a_2 = W_2 * h_1 + b_2 = [\ 0.32 \quad 0.29 \quad 0.85\ ]$$

$$\hat{y} = softmax(a_2) = [\ 0.2718 \quad 0.2634 \quad 0.4648\ ]$$

Cross Entropy Loss:

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

$$L(\Theta) = -1 * \log(0.4648)$$
$$= 0.7661$$

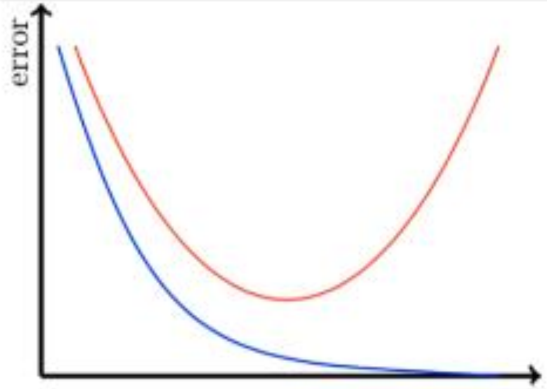
Hyper parameter tuning

Network Architectures

- Number of layers
- Number of neurons

Algorithms

- Vanilla/Momentum /Nesterov GD
- AdaGrad
- RMSProp
- Adam



Activation Functions

- tanh (RNNs)
- relu (CNNs, DNNs)
- leaky relu (CNNs)

Strategies

- Batch
- Mini-Batch (32, 64, 128)
- Stochastic
- Learning rate schedule

Regularization

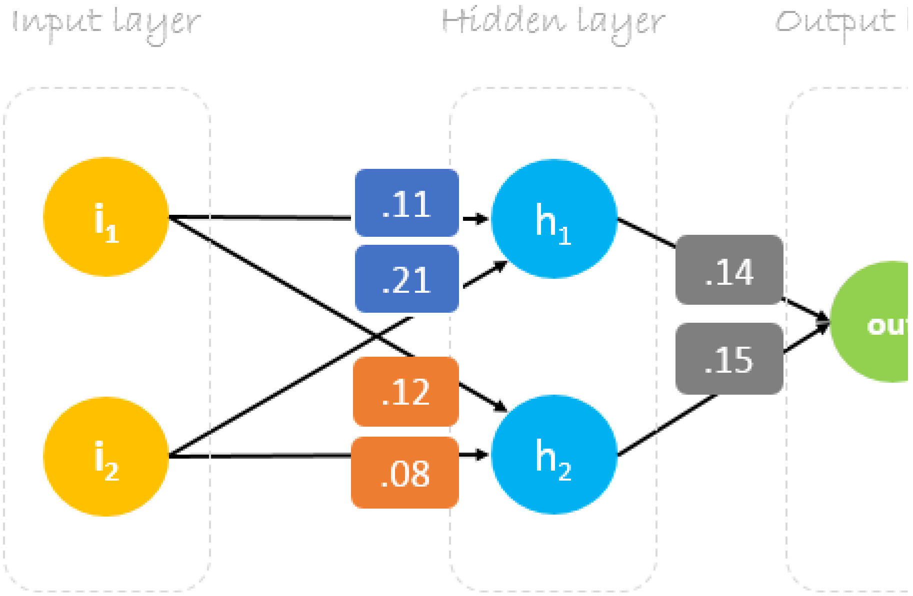
- L2
- Early stopping
- Dataset augmentation
- Drop-out
- Batch Normalizat

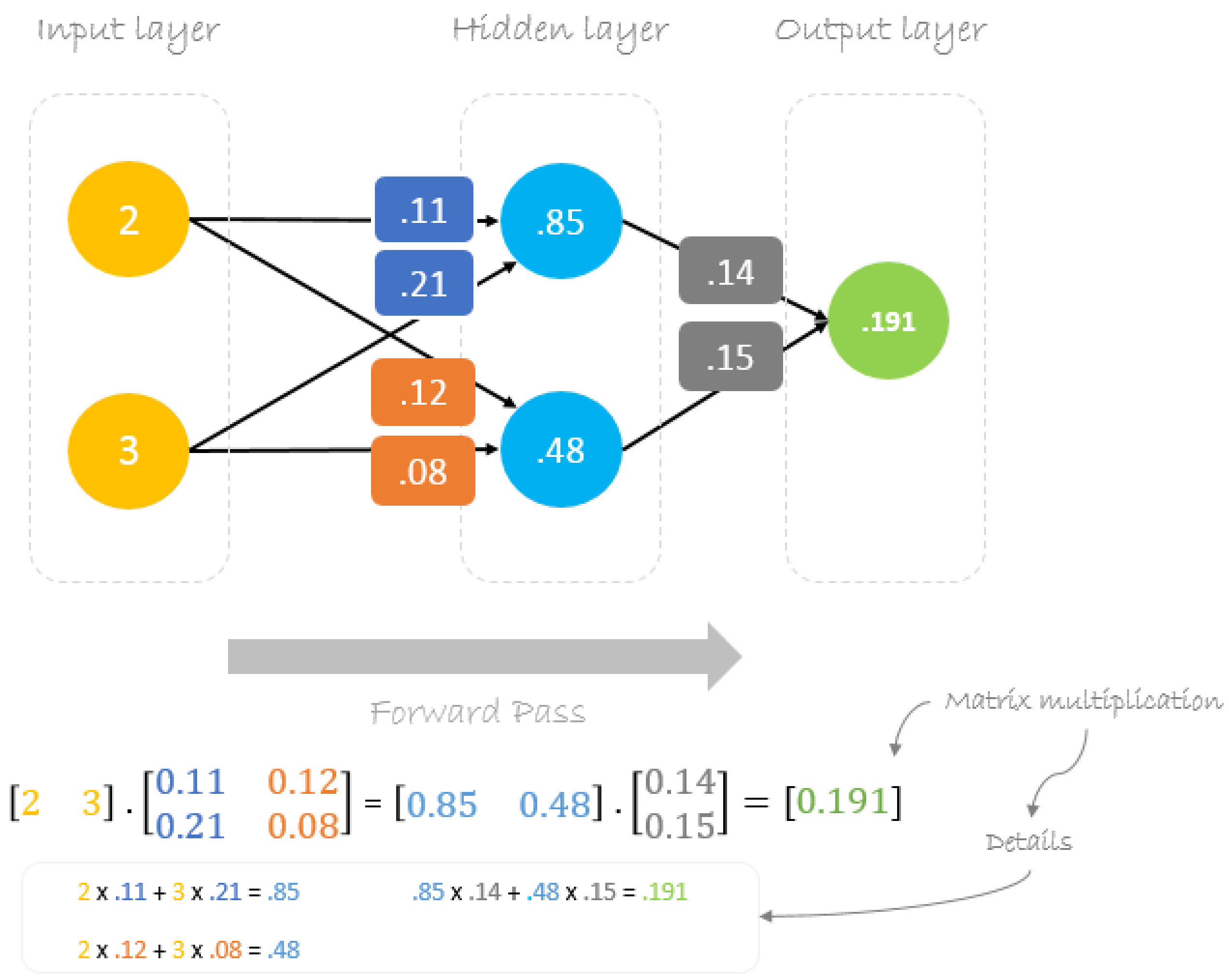
Initialization Methods

- *Xavier*
- *He*

Namah Shivaya

https://hmkcode.com/a





https://visualstudiomagazine.com/articles/2014/04/01/neural-network-cross-entropy-error.aspx