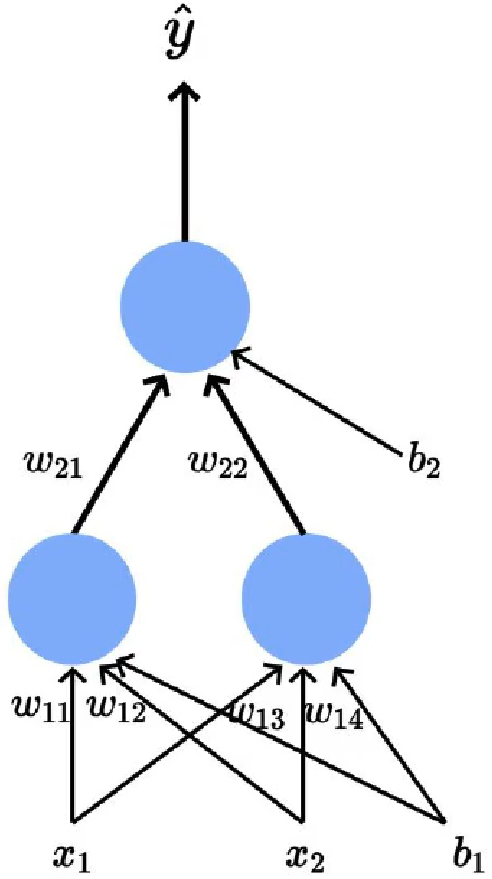
AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

# 19CSE437
# DEEP LEARNING FOR COMPUTER VISION
**L-T-P-C: 2-0-3-3**

# 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*

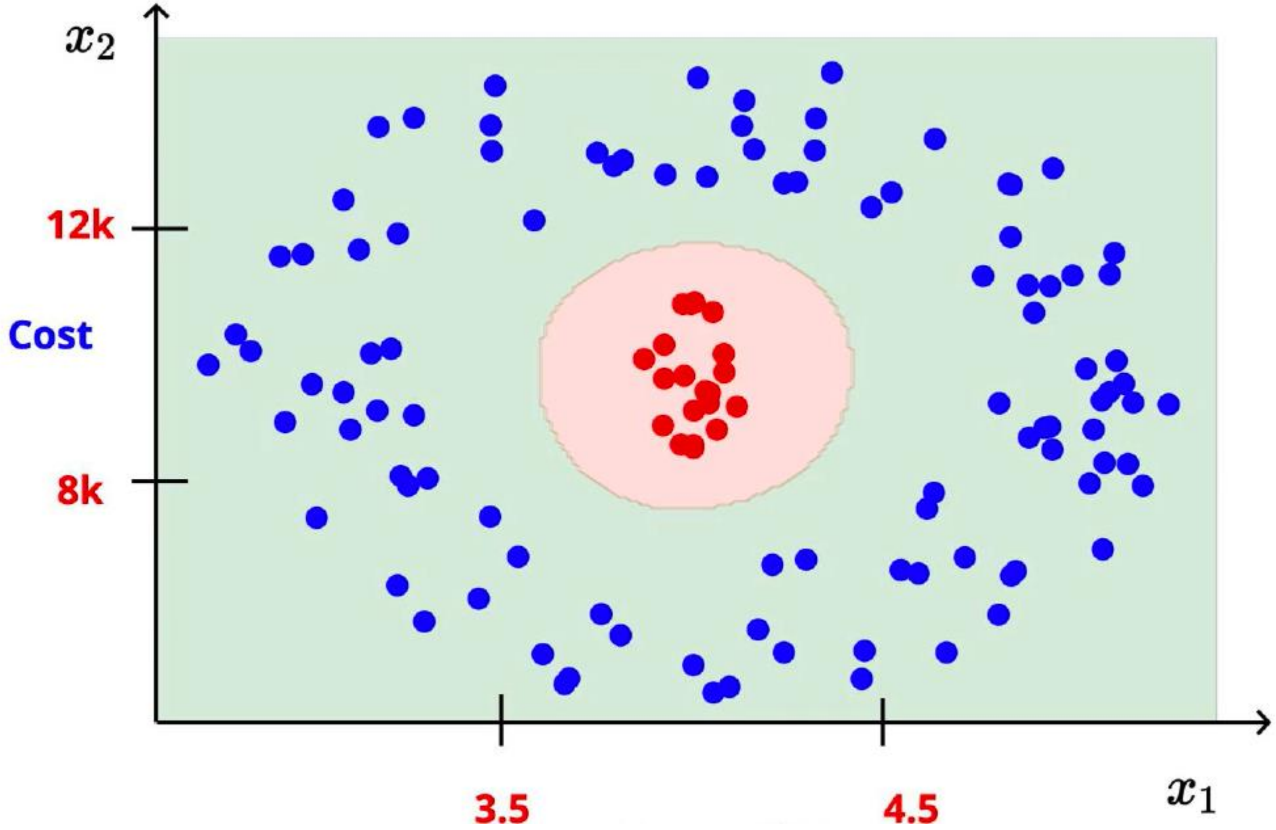
# Multiple sigmoid Neurons




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

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

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



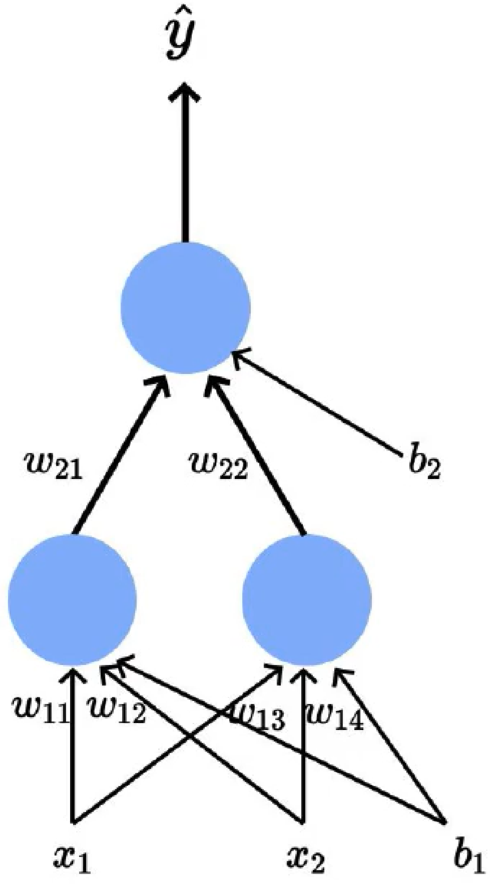

$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

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

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

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

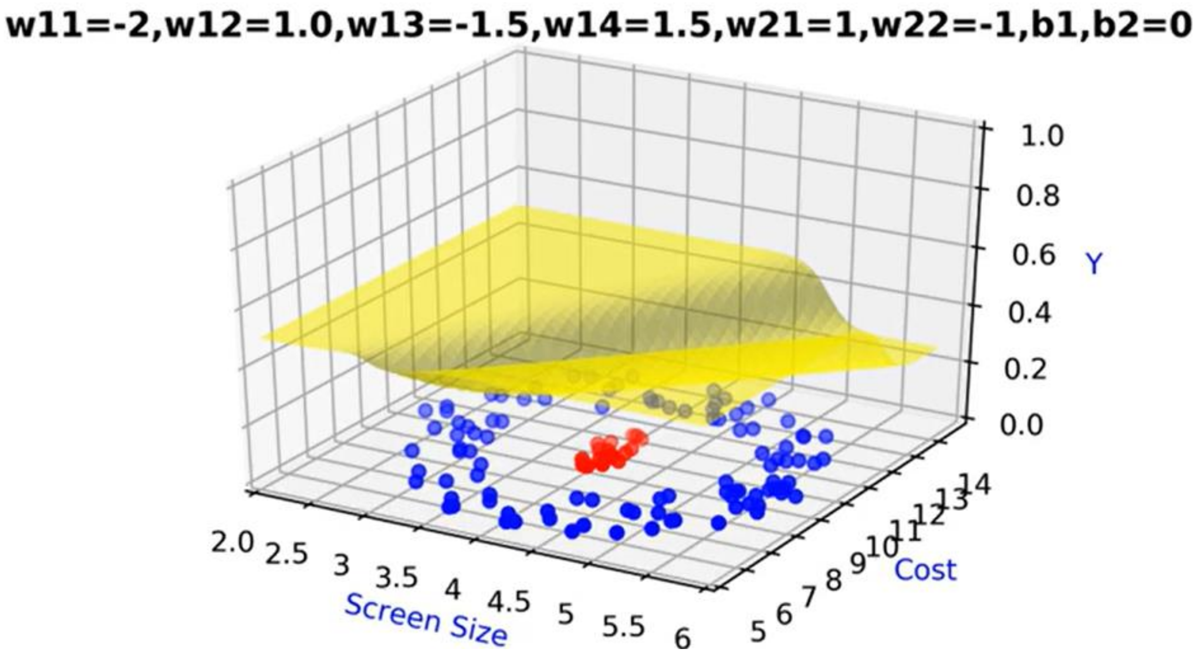
# Multiple sigmoid Neurons




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

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

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



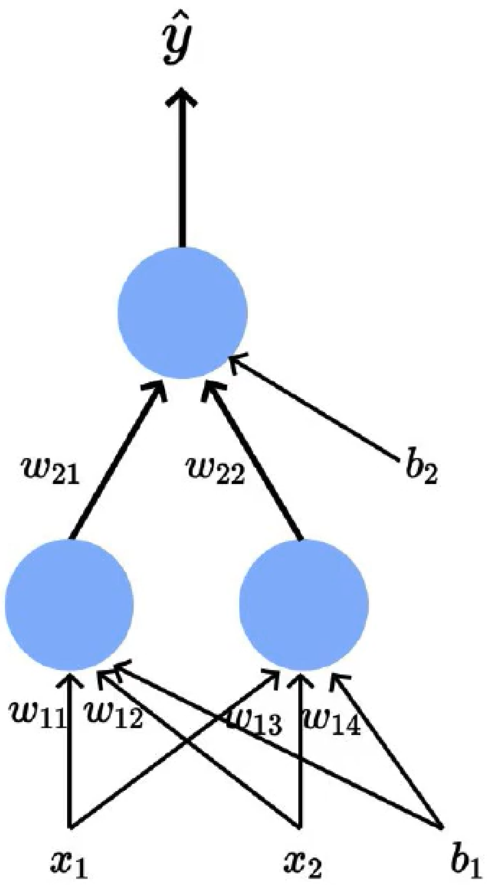

$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

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

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

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

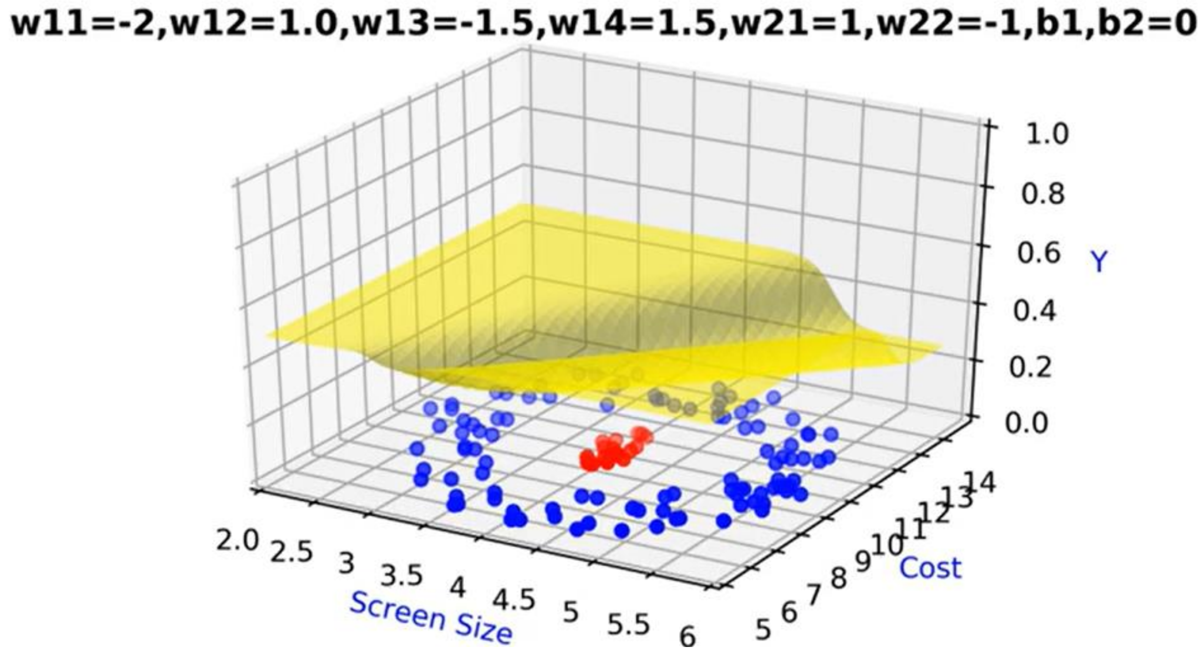
# Multiple sigmoid Neurons




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

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

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



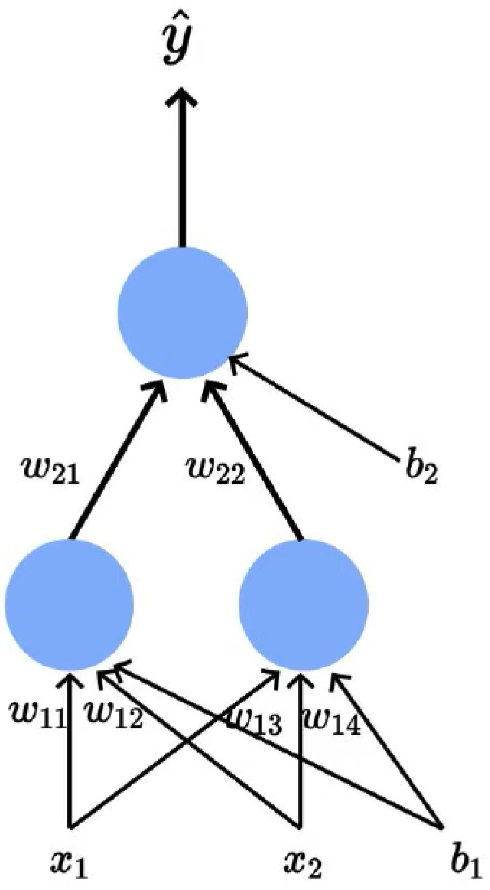

$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

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

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

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

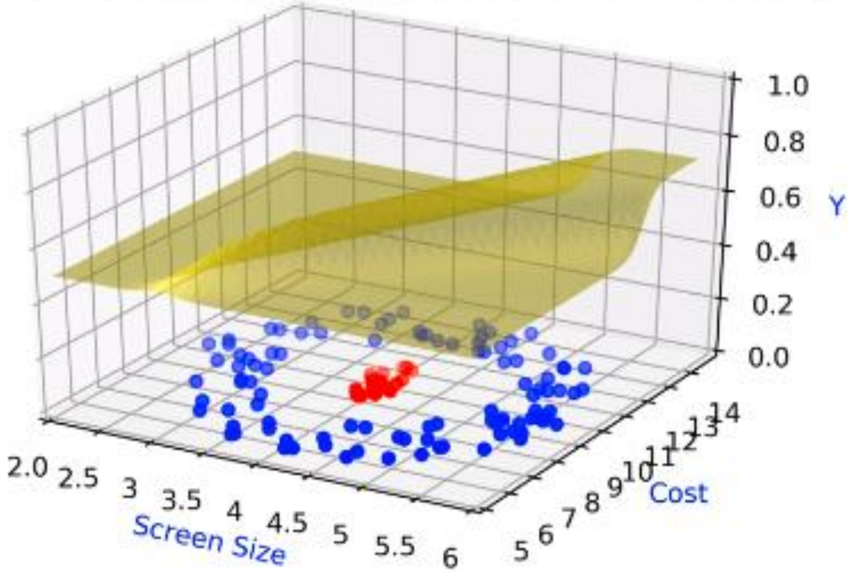
# Multiple sigmoid Neurons



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

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

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



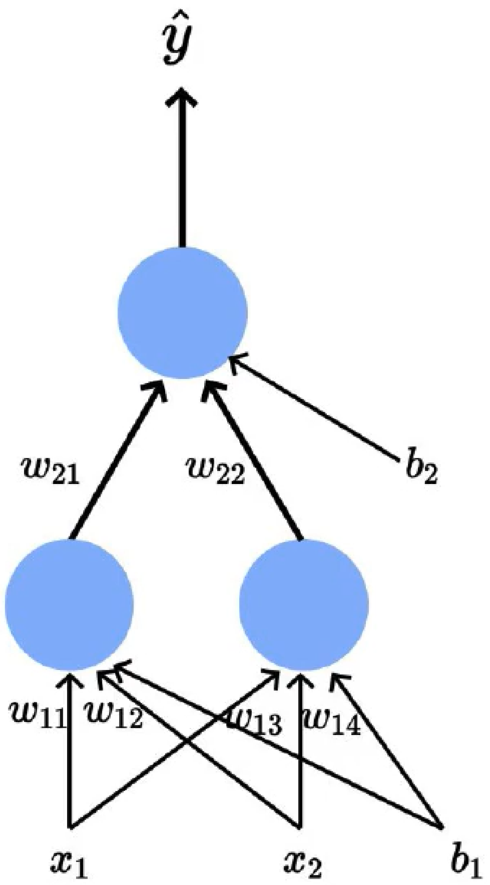

$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

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

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

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

# Multiple sigmoid Neurons




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

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

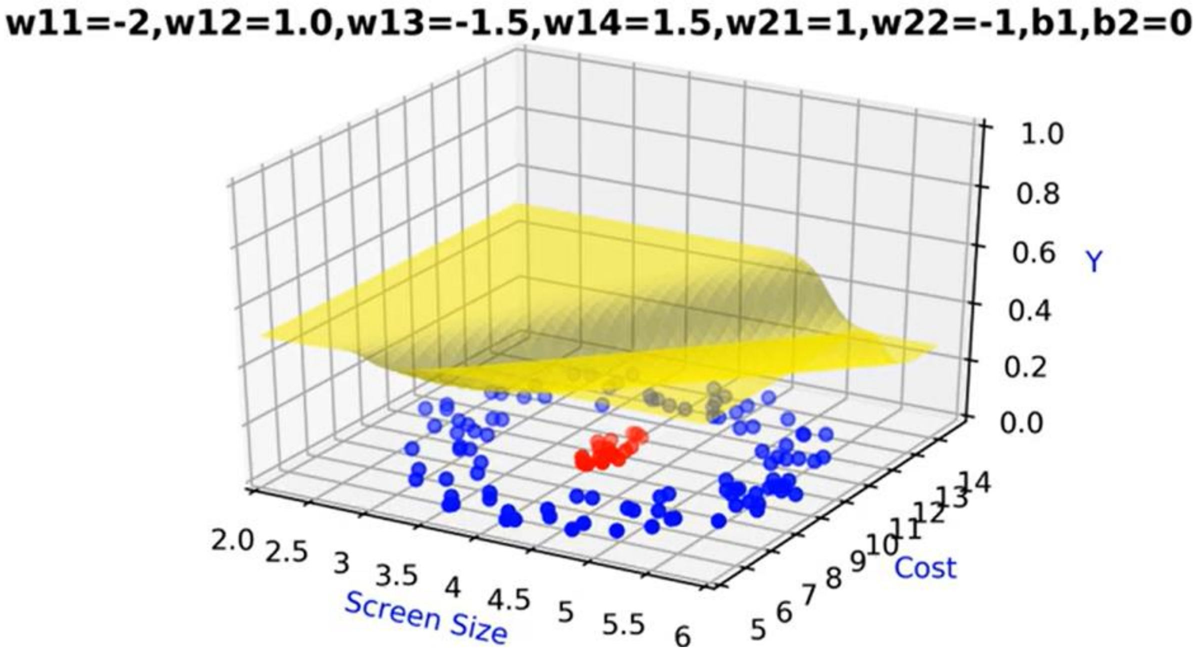
$$\hat{y} = g(h_1, h_2)$$

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

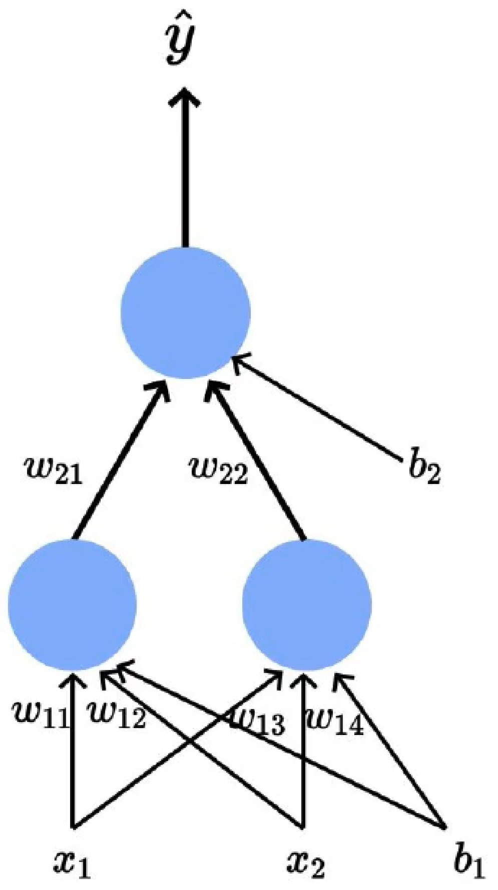
$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

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

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



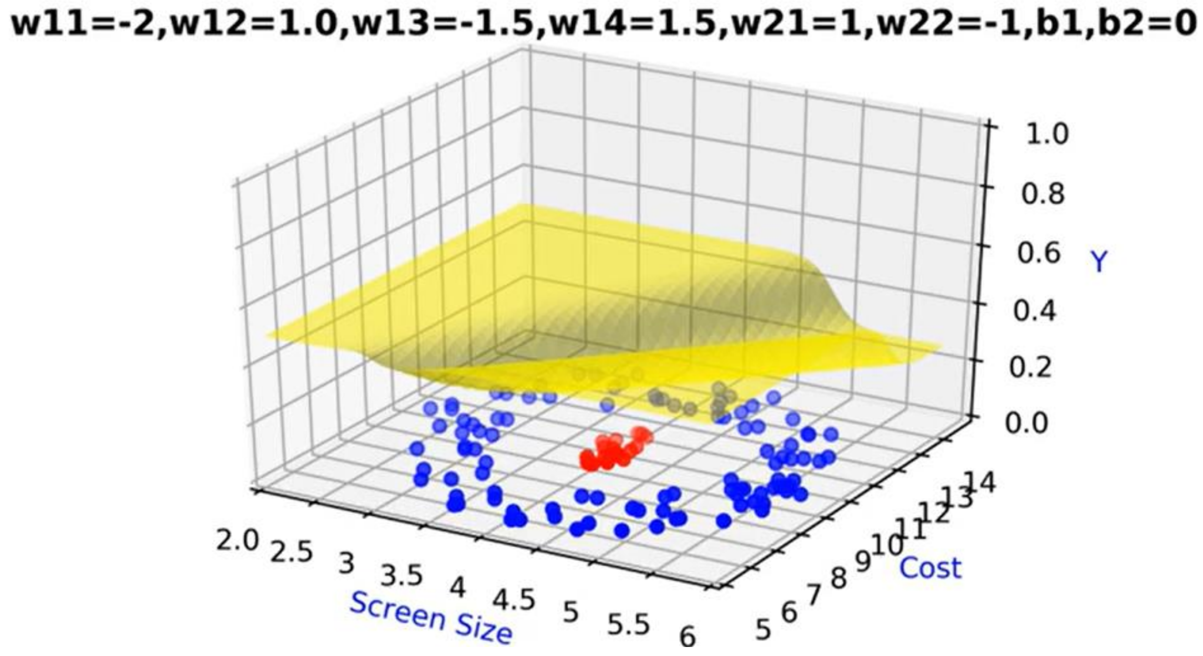
# Multiple sigmoid Neurons




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

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

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



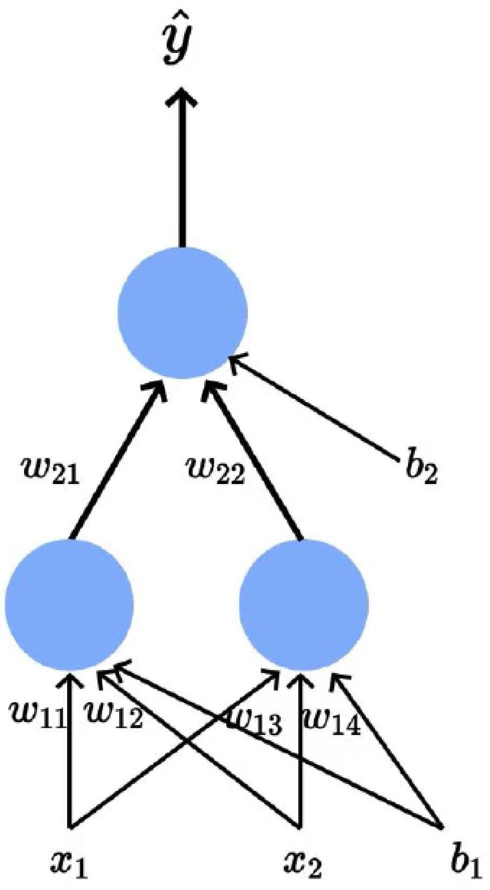

$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

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

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

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

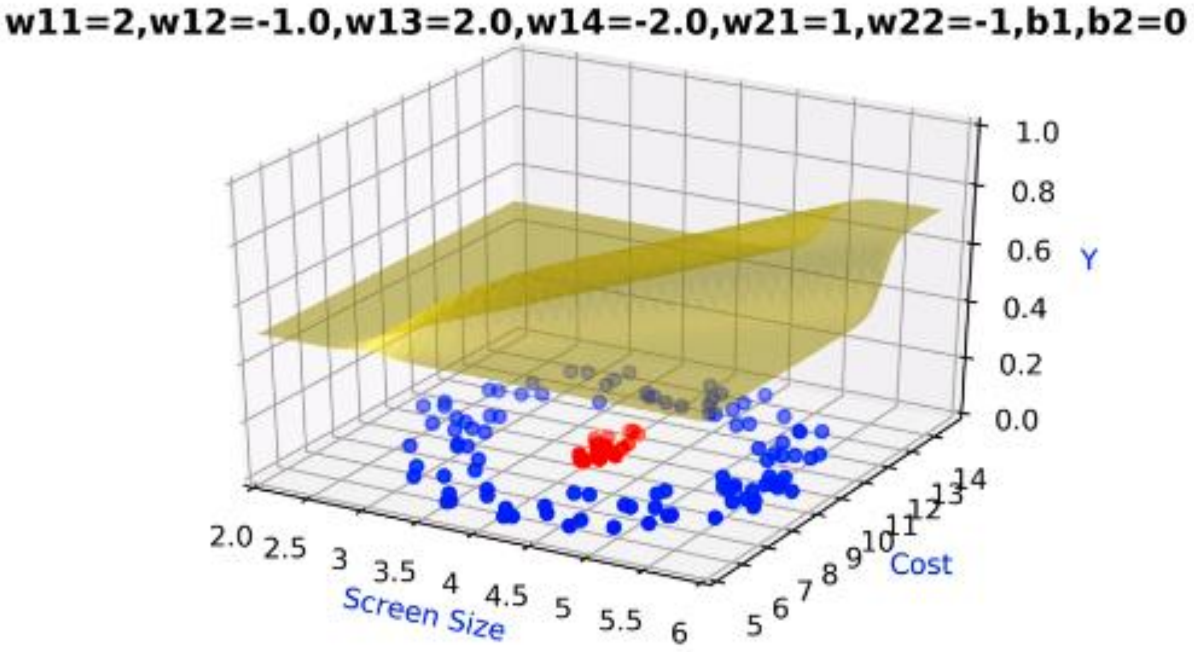
# Multi Layer Neural Networks




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

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

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




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

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

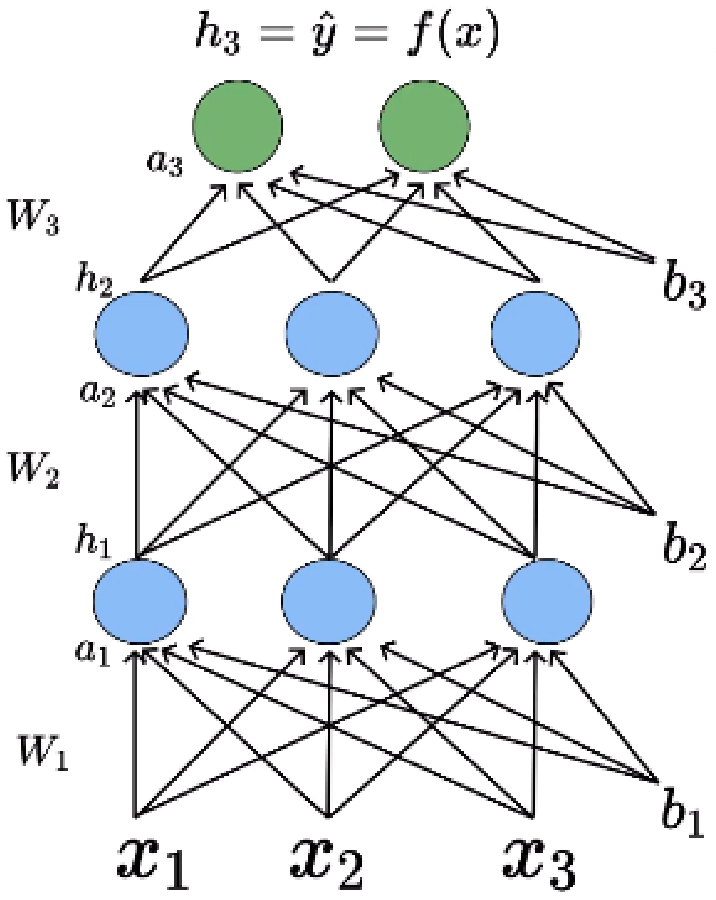
$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

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

# Feed Forward Neural Networks

# Understanding the computation

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




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

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

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

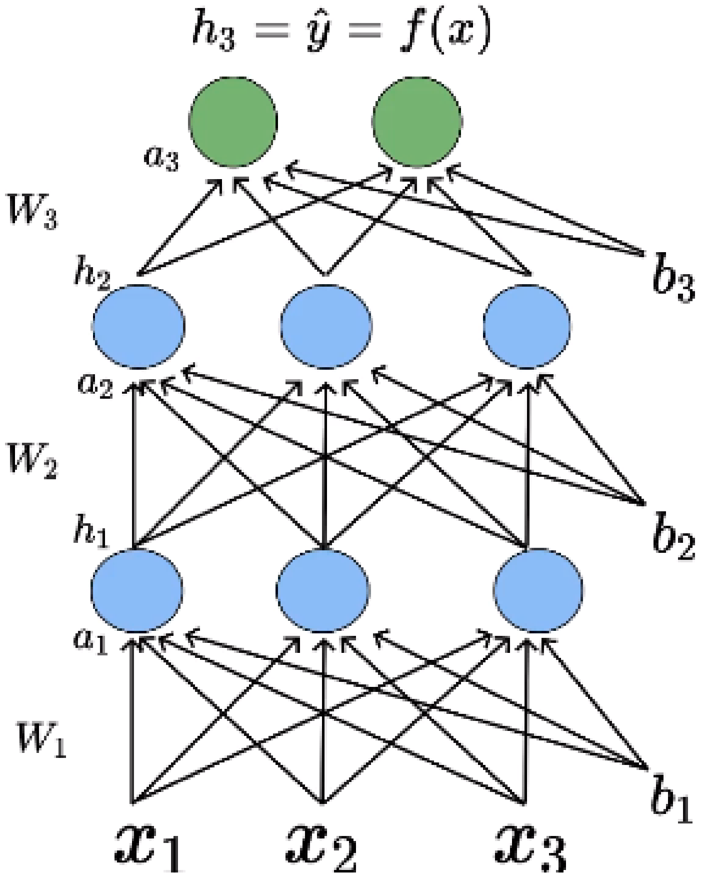
$$\vdots \qquad \vdots \qquad \vdots$$

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

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

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

# Feed Forward Neural Networks




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

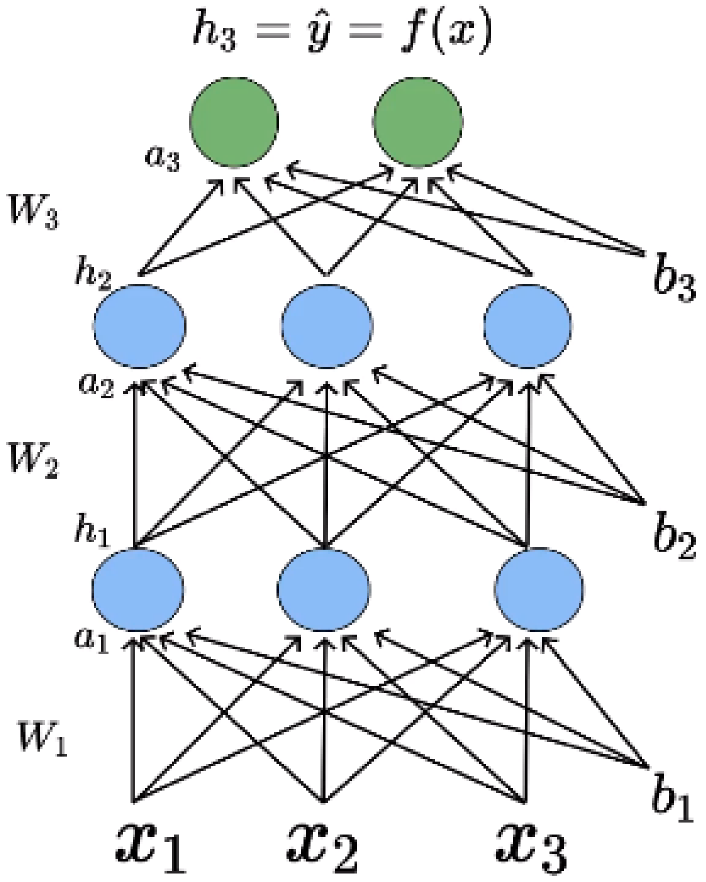
$$a_1 = W_1 * x + b$$

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

$$h_1 = g(a_1)$$

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

# 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

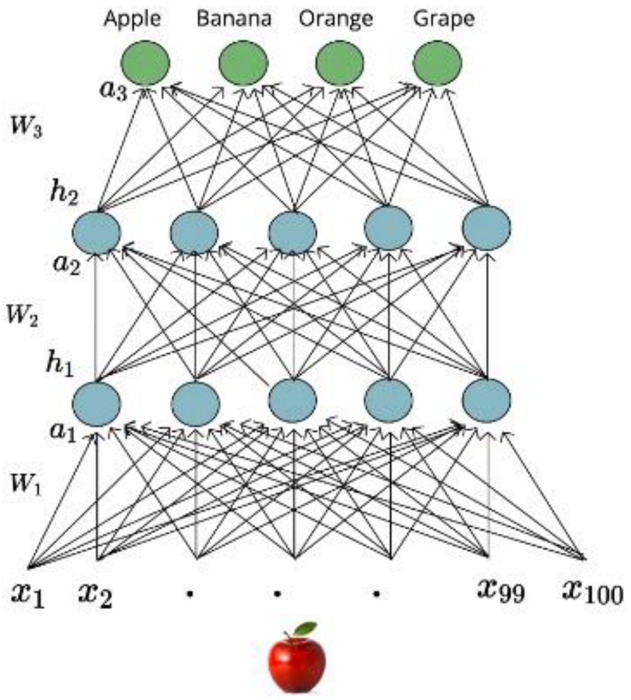
$$h_i(x) = g(a_i(x))$$

  where 'g' is called as the activation function

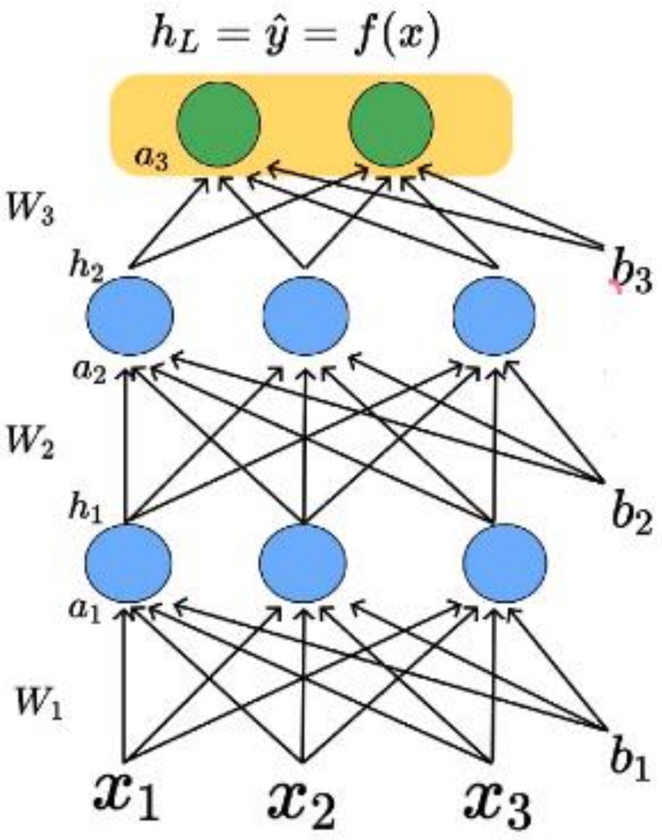
- The activation at output layer 'L' is given by

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

  where 'O' is called as the output activation function



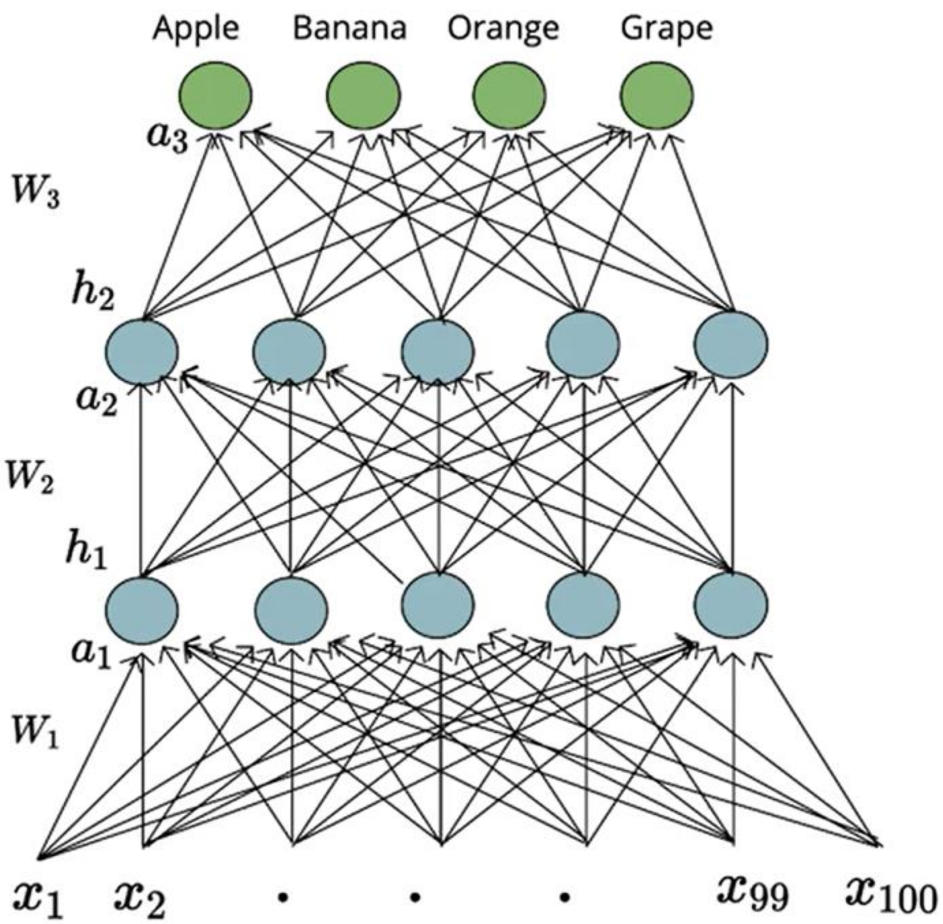
# Feed Forward Neural Networks




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

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

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




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

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

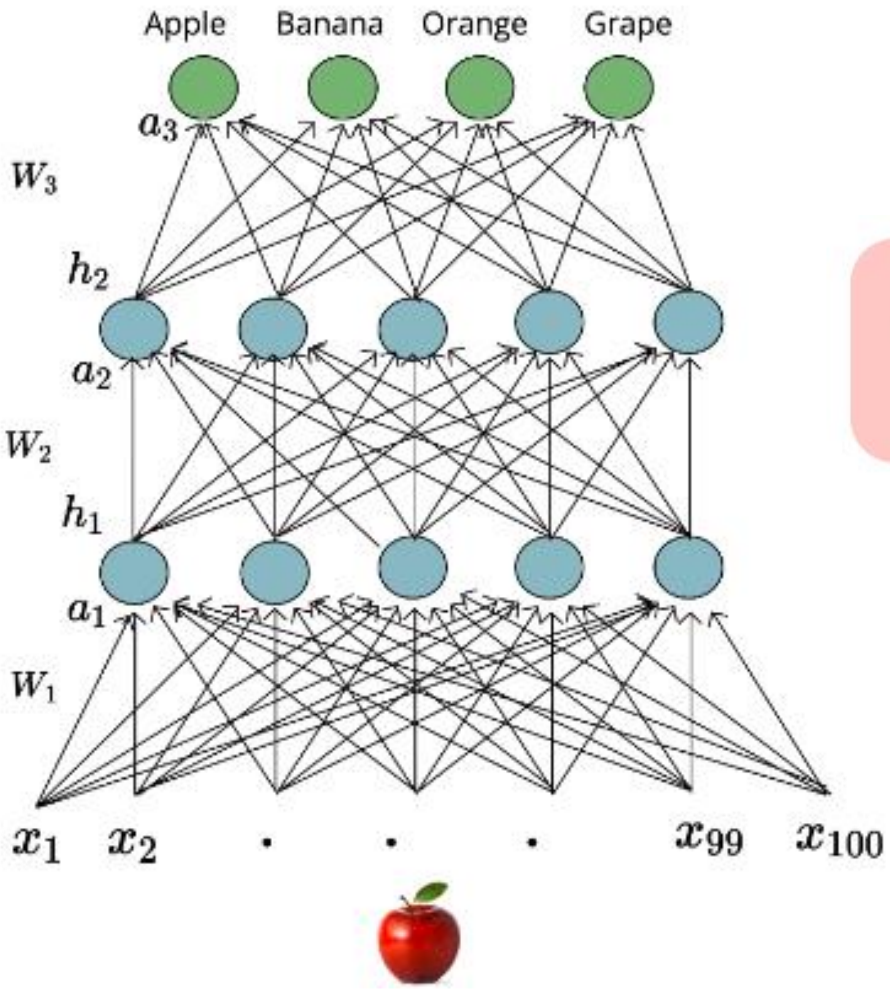
$$\hat{y}_2 = \frac{4}{(3+4+10+3)} = 0.20$$

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

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

Take each entry and divide by the sum of all entries

# Output Layer




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

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

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

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

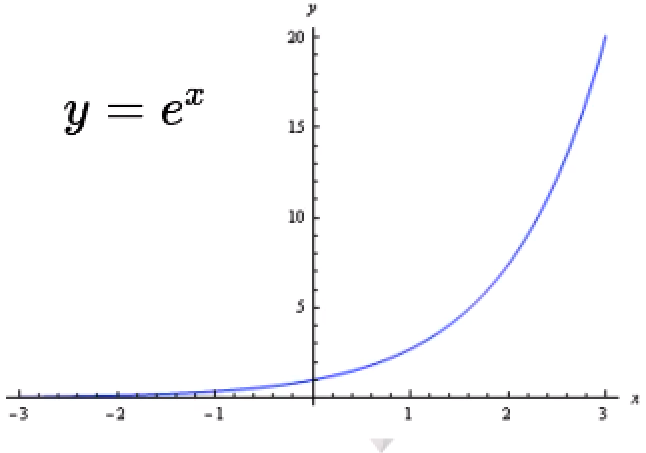
$$\hat{y}_3 = \frac{4}{(7 + (-2) + 4 + 1)} = 0.40$$

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

# Softmax

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

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

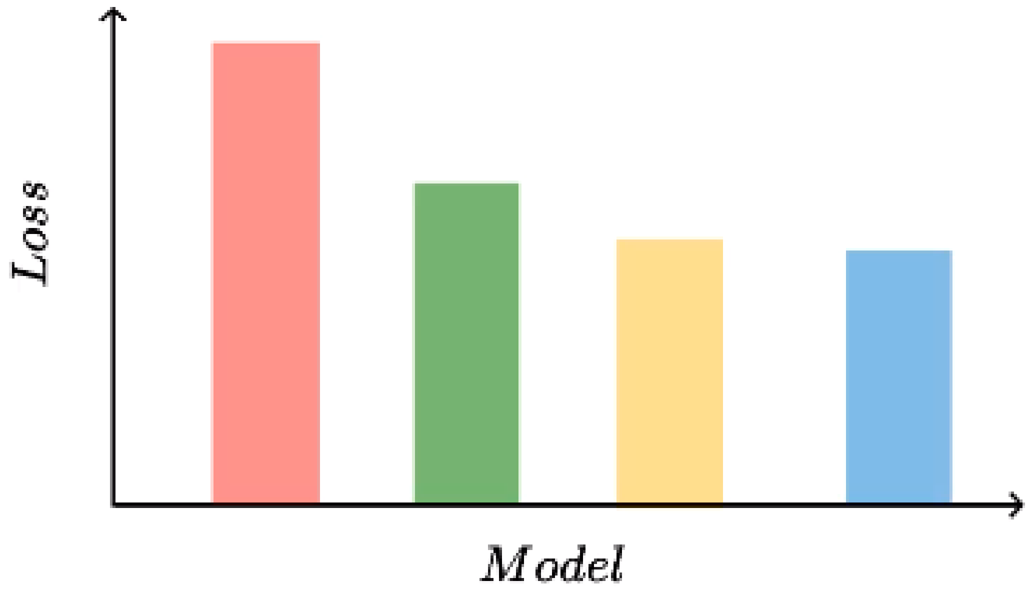


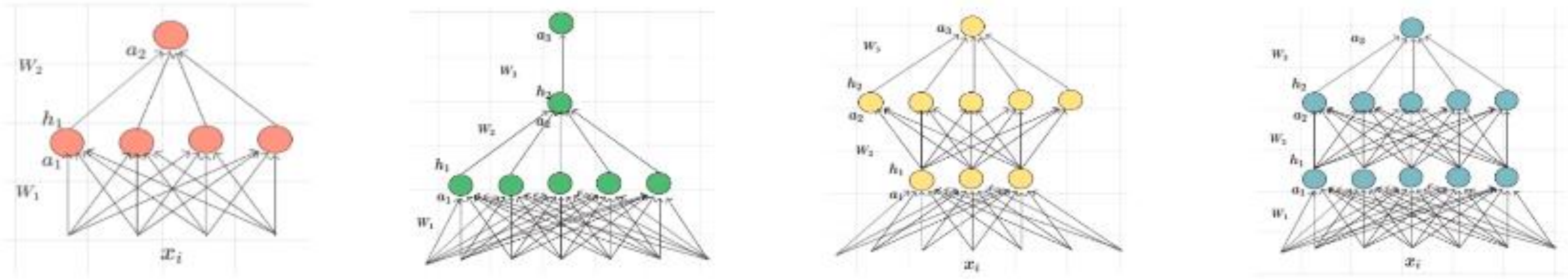

$$h = [\ h_1\ \ h_2\ \ h_3\ \ h_4\ ]$$

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

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

# Different network configurations



# Cross Entropy Loss Function

**Cross Entropy Loss- binary class**

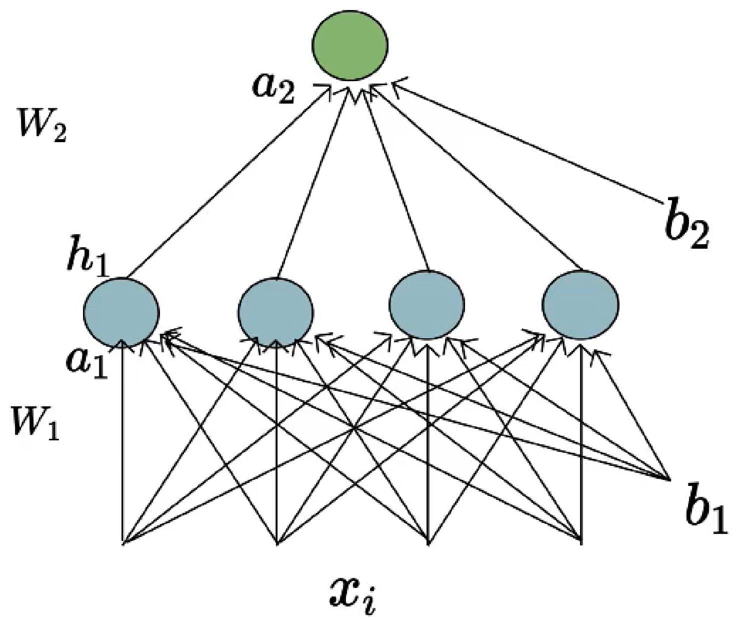
$$L(\Theta) = \begin{cases} -log(\hat{y}) & \text{if } y = 1 \\ -log(1-\hat{y}) & \text{if } y = 0 \end{cases}$$

**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




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

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

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

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

**Output :**

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

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

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

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

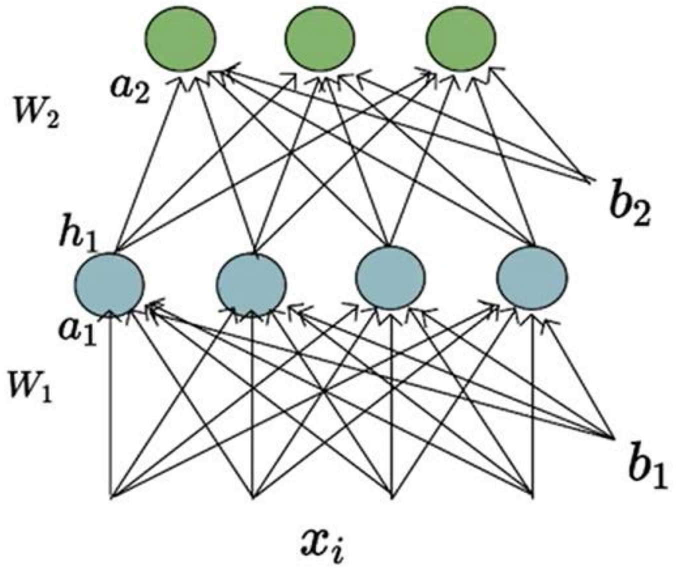
**Cross Entropy Loss:**

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

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

$$= 1.2560$$

# 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:**

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

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

*Can we use the same Gradient Descent algorithm as before ?*

# Learning Algorithm

**Initialise** $w, b$

**Iterate over data:**

$compute\ \hat{y}$

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

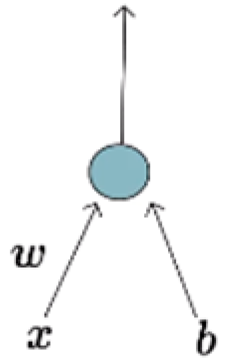
$$w_{111} = w_{111} - \eta \Delta w_{111}$$

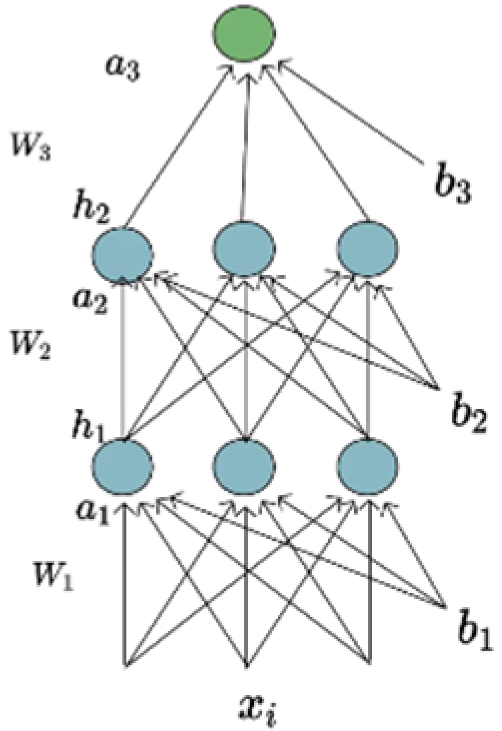
$$w_{112} = w_{112} - \eta \Delta w_{112}$$

....

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

**till satisfied**



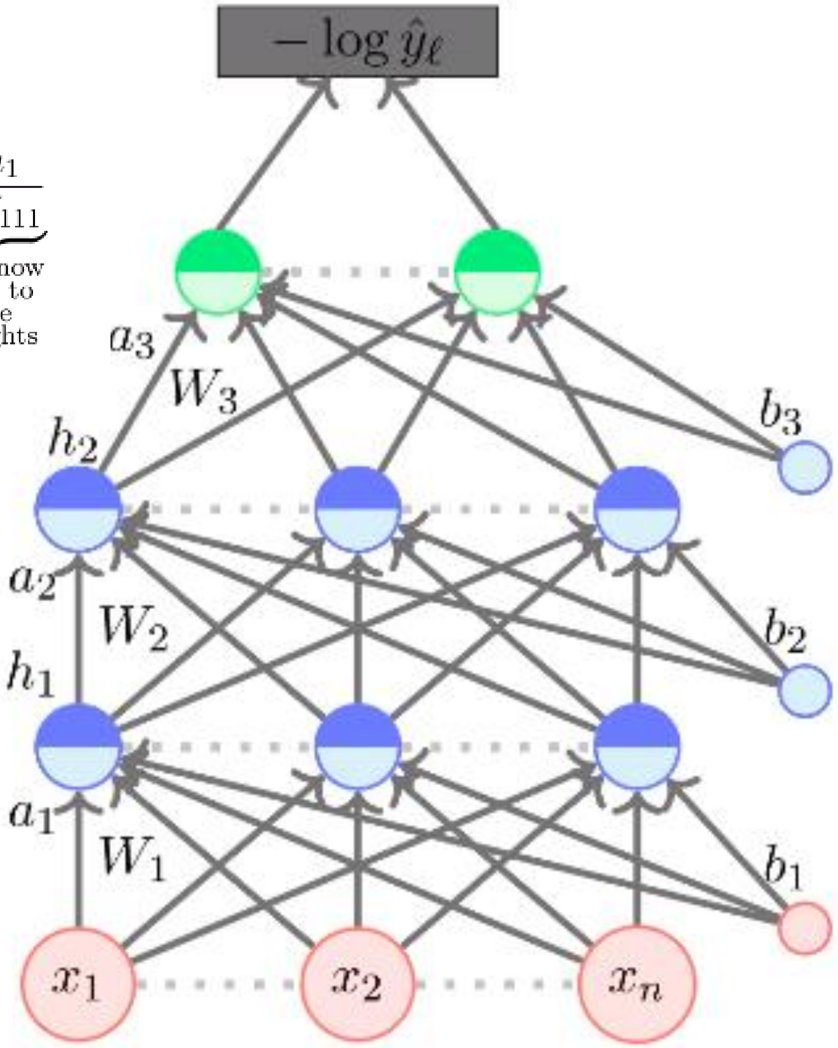

$Earlier : w, b$

$Now : w_{111}, w_{112}, \ldots$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}$$

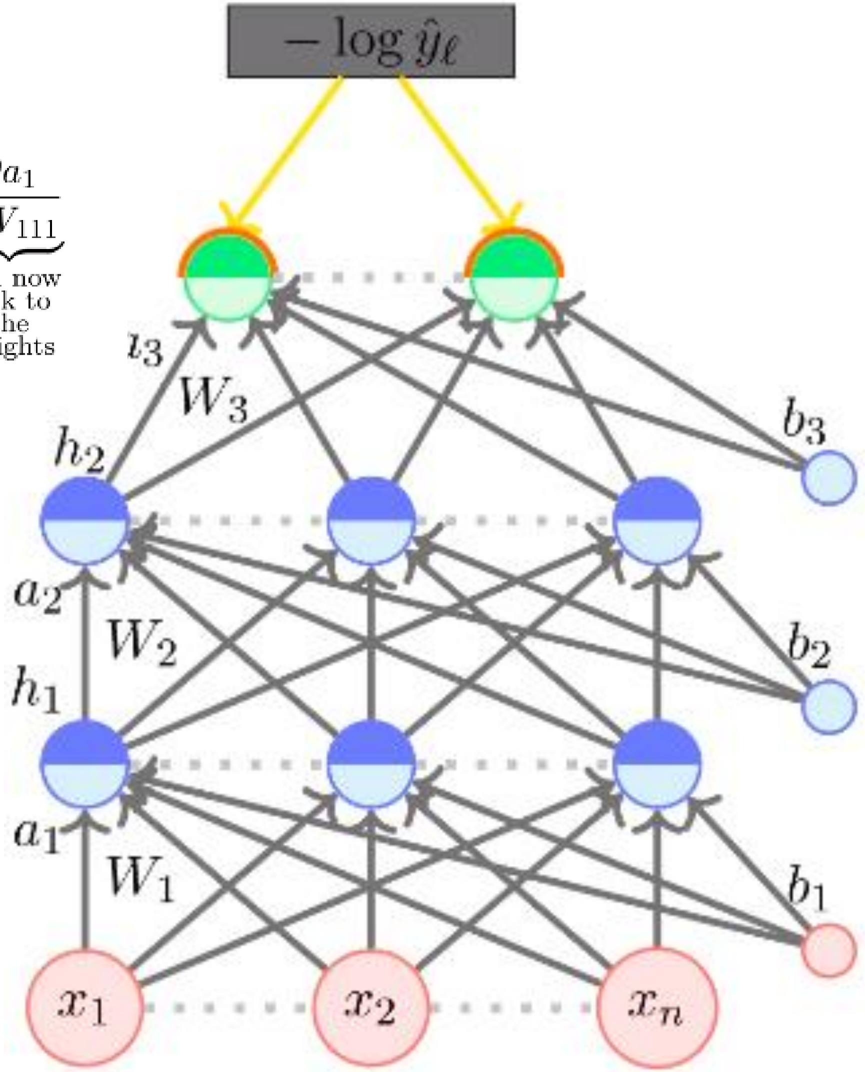
# Backpropagation

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



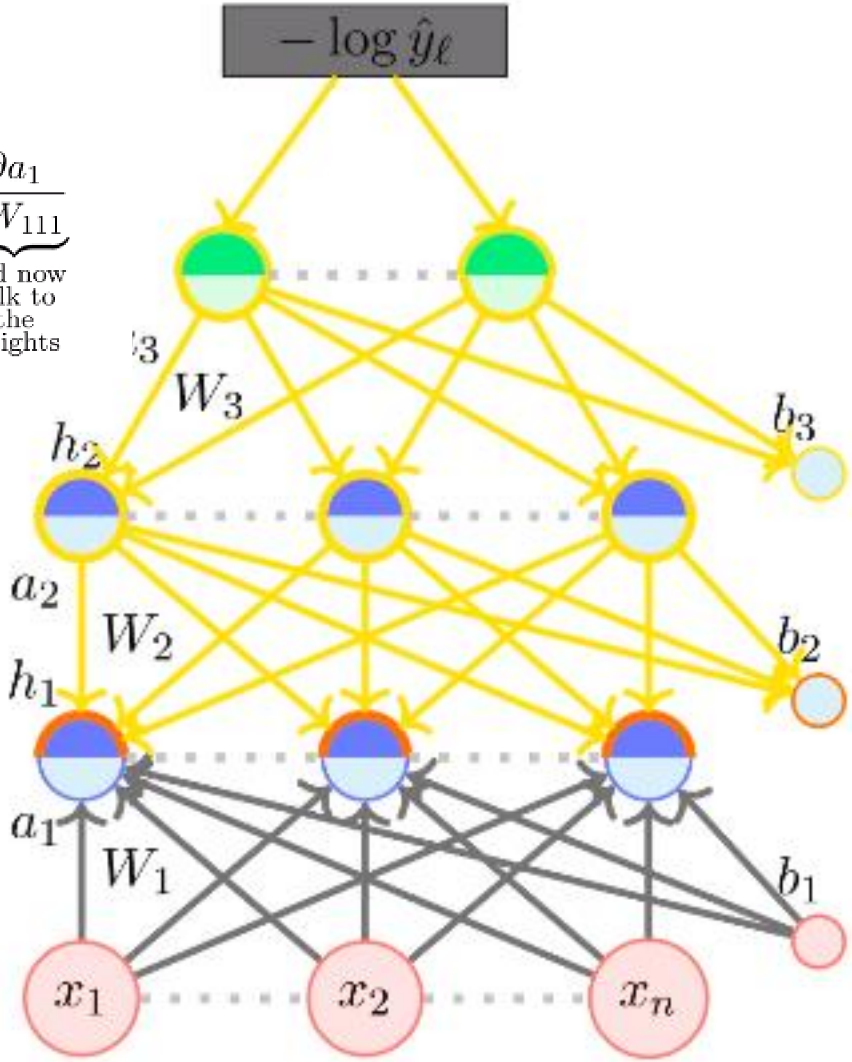
# Backpropagation

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



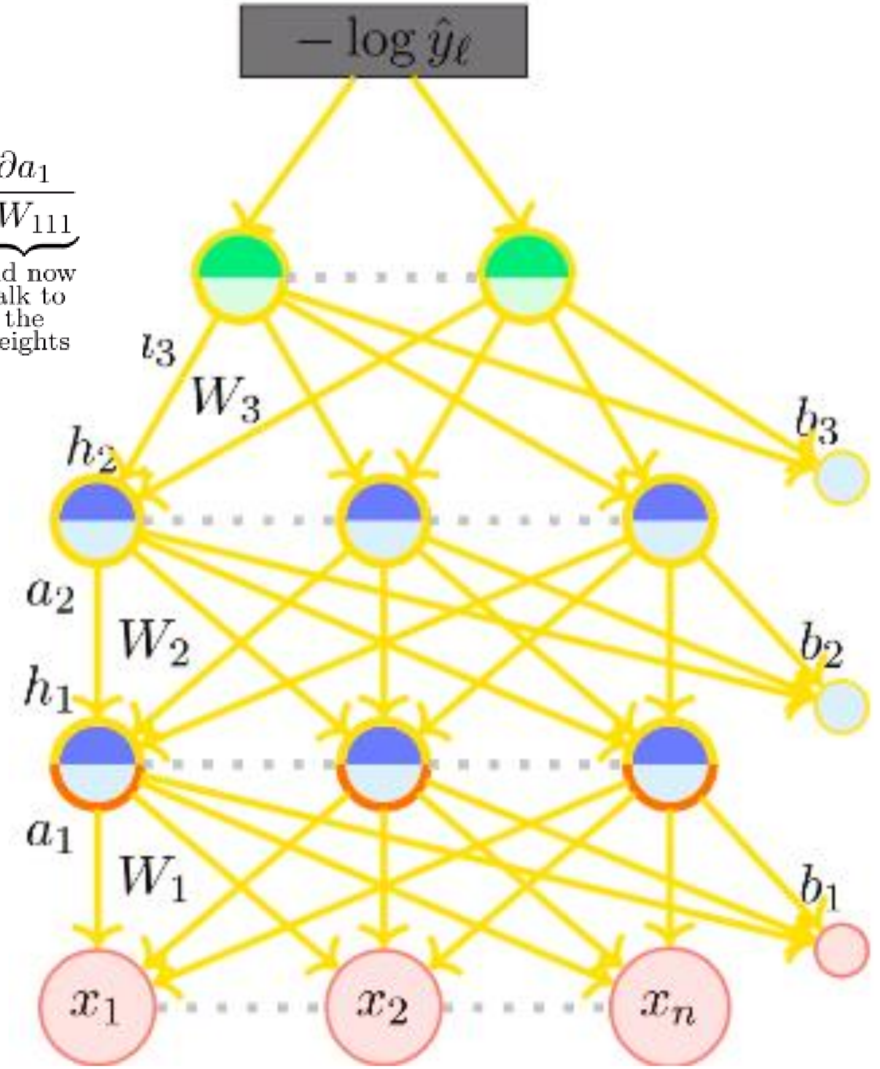
# Backpropagation

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



# Backpropagation

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

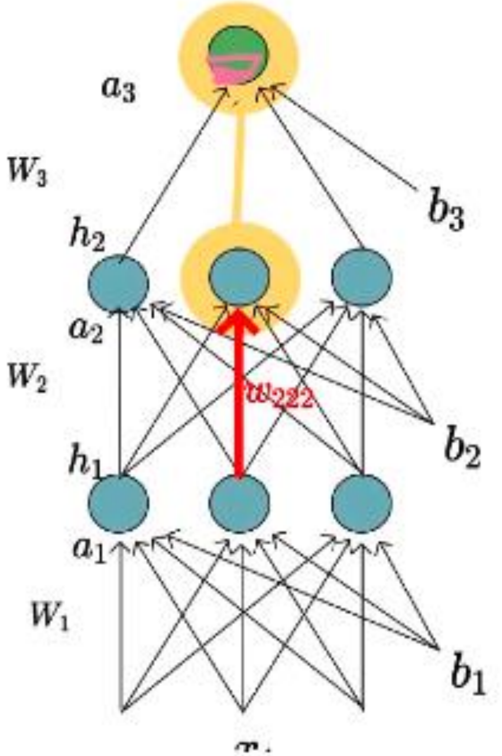


# Chain rule

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

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

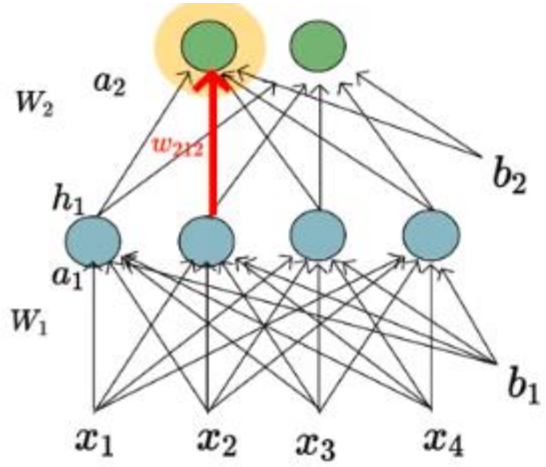
# Learning algorithm- Back propagation




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

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

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




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

$$W_1 = \begin{bmatrix} 0.1 & 0.3 & 0.8 & -0.4 \\ -0.3 & -0.2 & 0.5 & 0.5 \\ -0.3 & 0 & 0.5 & 0.4 \\ 0.2 & 0.5 & -0.9 & 0.7 \end{bmatrix}$$

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

$x = [\ 2 \quad 5 \quad 3 \quad 3\ ]$ $\qquad$ $y = [\ 1 \quad 0\ ]$

$$\frac{\partial L}{\partial w_{212}} = \left(\frac{\partial L}{\partial a_{21}}\right).\left(\frac{\partial a_{21}}{\partial w_{212}}\right) = \left(\frac{\partial L}{\partial \hat{y}_1}\right).\left(\frac{\partial \hat{y}_1}{\partial a_{21}}\right).\left(\frac{\partial a_{21}}{\partial w_{212}}\right)$$

$$\frac{\partial L}{\partial \hat{y}_1} = -2(y_1 - \hat{y}_1) = -0.46$$

$$\frac{\partial \hat{y}_1}{\partial a_{21}} = \hat{y}_1 * (1 - \hat{y}_1)(-a_{22}) = -0.079$$

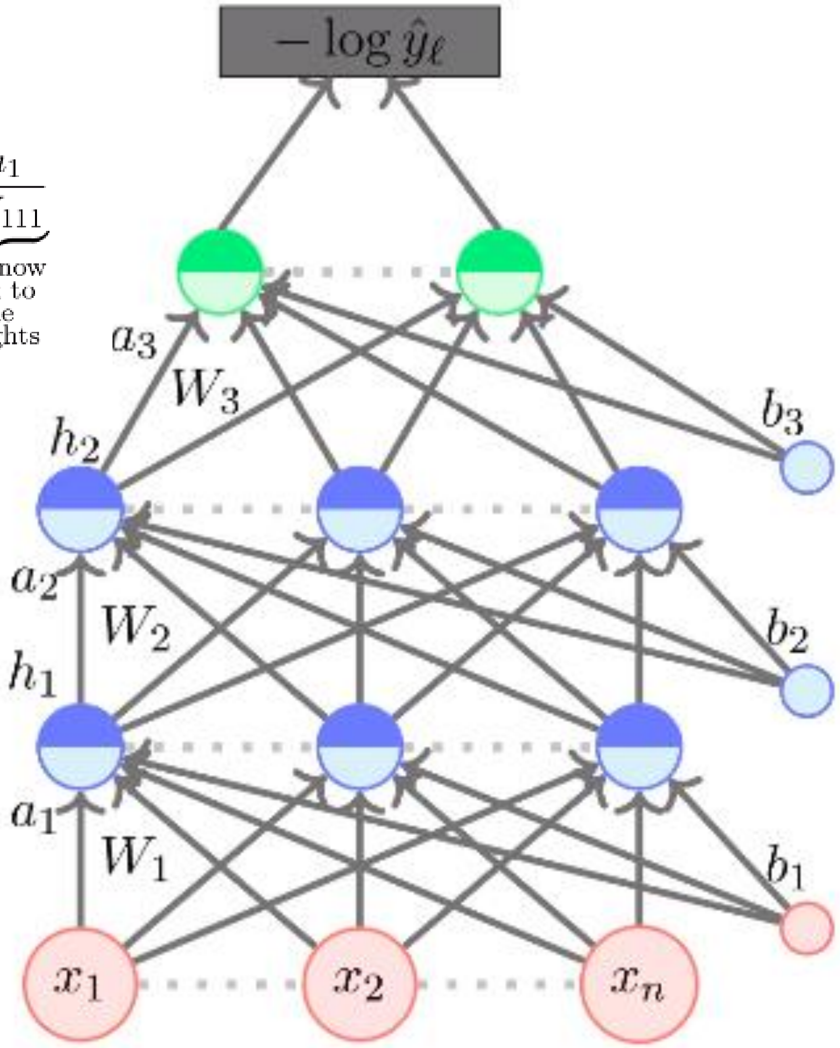
$$\frac{\partial a_{21}}{\partial w_{212}} = h_{12} = 0.80$$

$$\frac{\partial L}{\partial w_{212}} = (-2(y_1 - \hat{y}_1)) * (\hat{y}_1(1 - \hat{y}_1)(-a_{22})) * (h_{12})$$

$$= (-0.46) * (-0.079) * (0.80) = -0.029$$

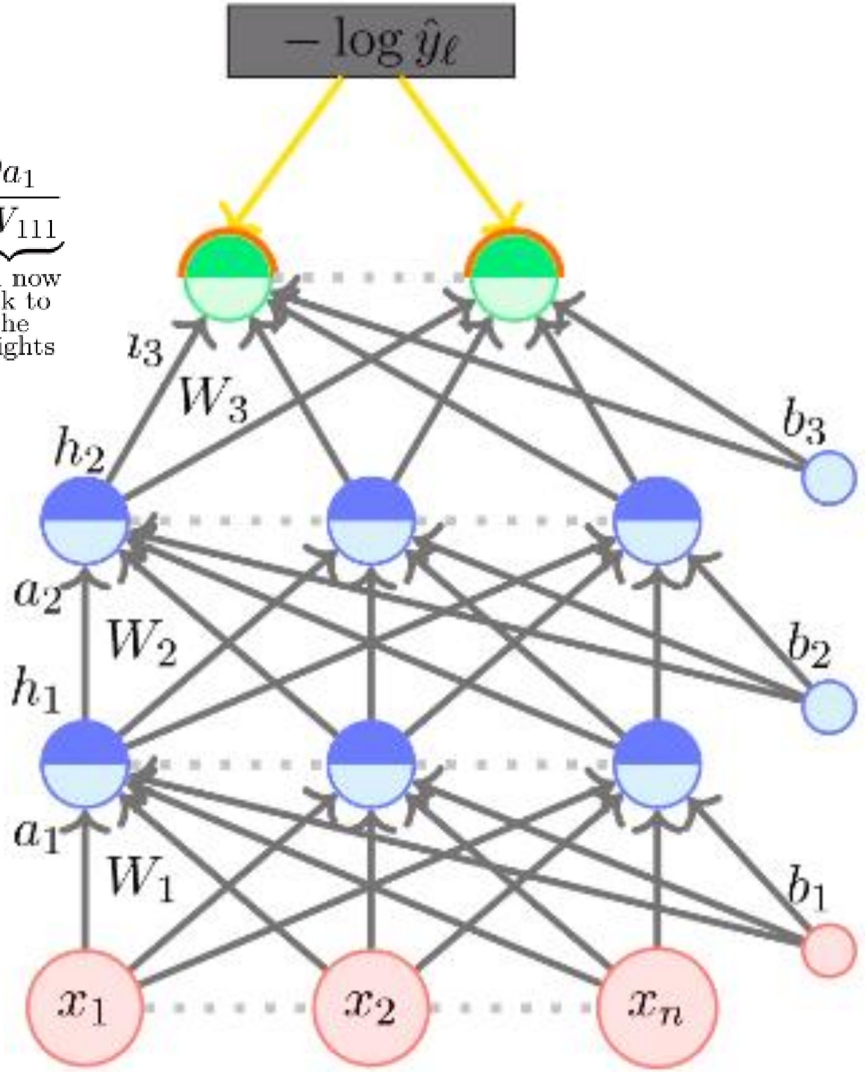
# Backpropagation

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



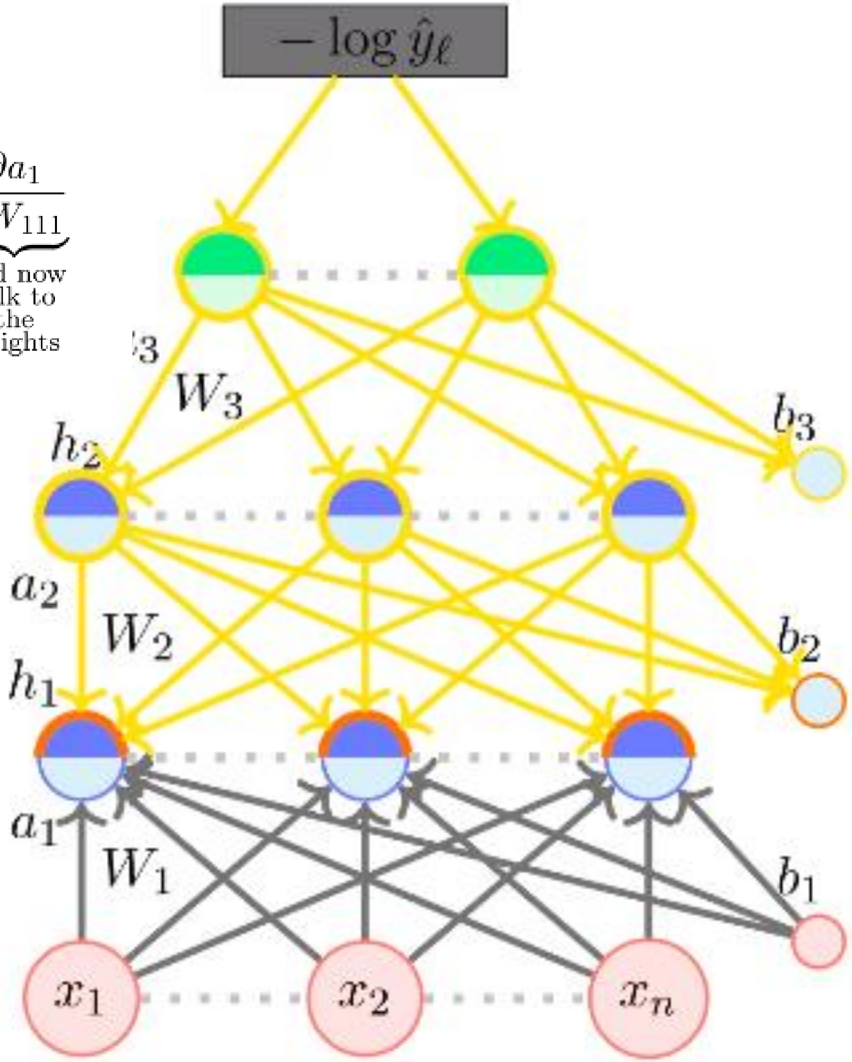
# Backpropagation

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



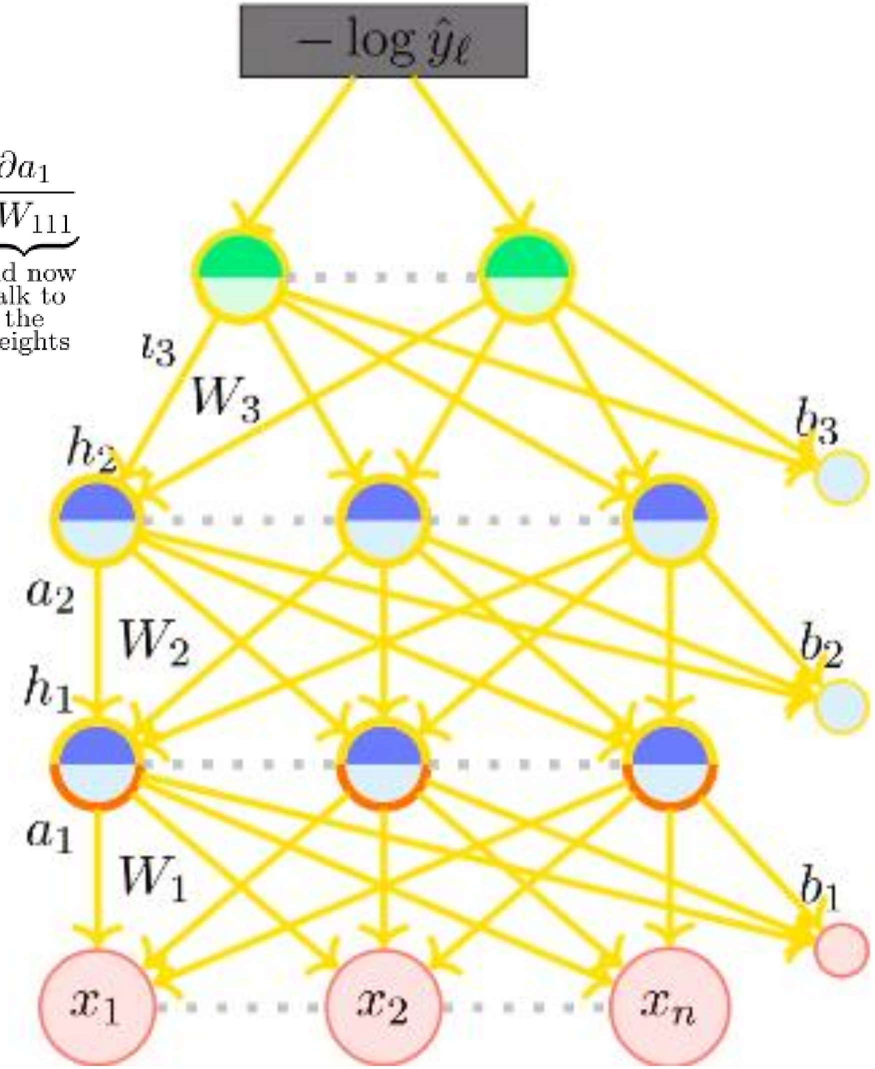
# Backpropagation

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



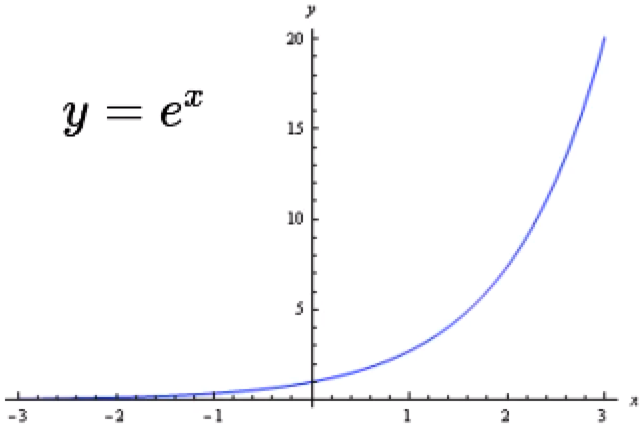
# Backpropagation

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



# Softmax- Output layer activation function

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



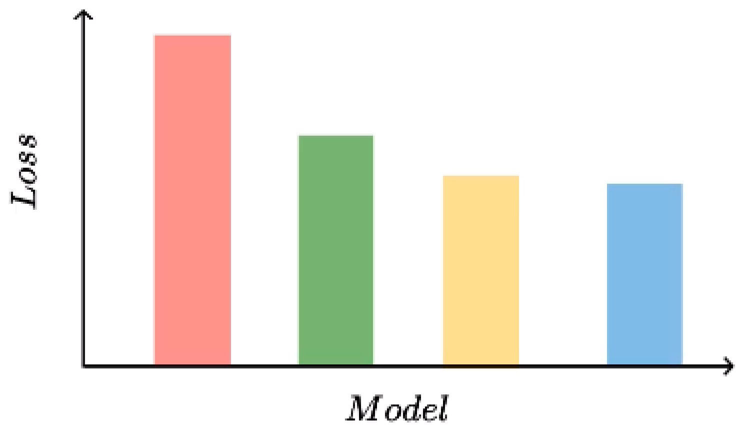

$$h = [\ h_1 \quad h_2 \quad h_3 \quad h_4\ ]$$

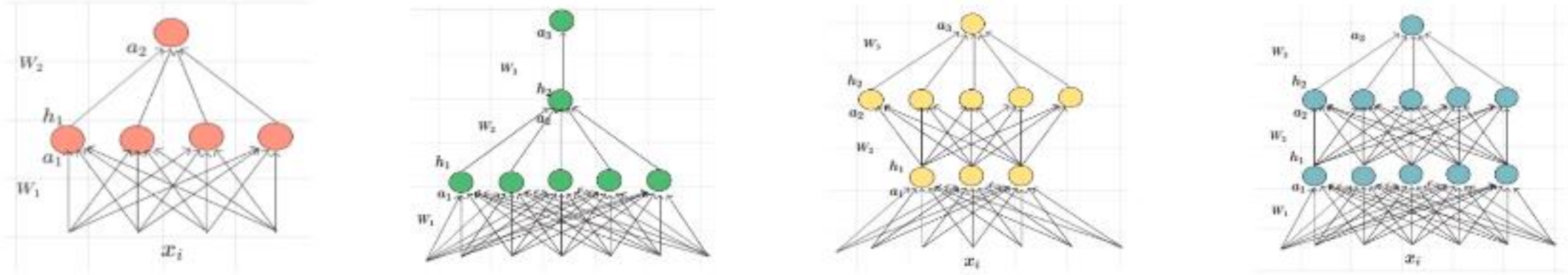
$$softmax(h) = [softmax(h_1) \quad softmax(h_2) \quad softmax(h_3) \quad softmax(h_4)]$$

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

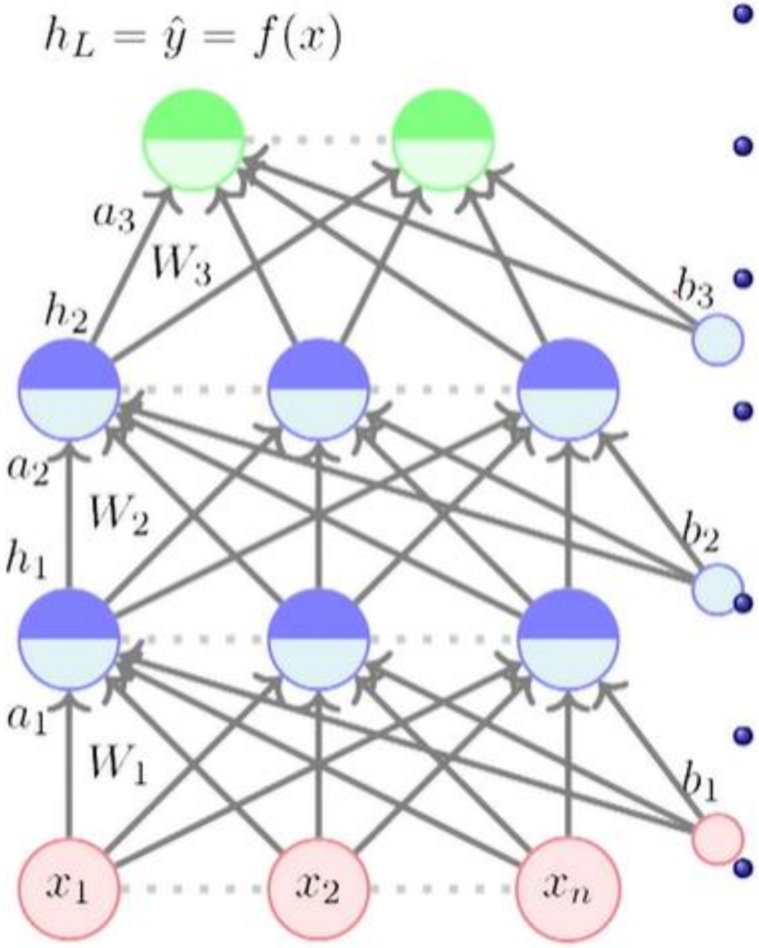
The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1.

# Try different models and check the loss



# Feed Forward Neural Networks




- The input to the network is an **n**-dimensional vector
- The network contains **L − 1** hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation ($a_i$ and $h_i$ are vectors)
- The input layer can be called the 0-th layer and the output layer can be called the ($L$)-th layer
- $W_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$ are the weight and bias between layers $i-1$ and $i$ $(0 < i < L)$
- $W_L \in \mathbb{R}^{n \times k}$ and $b_L \in \mathbb{R}^k$ are the weight and bias between the last hidden layer and the output layer ($L = 3$ in this case)

# Namah Shivaya