# Binary classification

- ★ There will be output 0 or 1
- ★ But it dosent mean it always given you True or not.
- ★ It will be used to differentiate between pictures, words, to translate mean wider applications - other than regression which include numbers

$$(x, y) \qquad x \in R^{n_x}, \quad y \in \{0, 1\}$$

$$X = \begin{bmatrix} x^{(1)}, & X, \ldots . X^{m} \end{bmatrix}$$

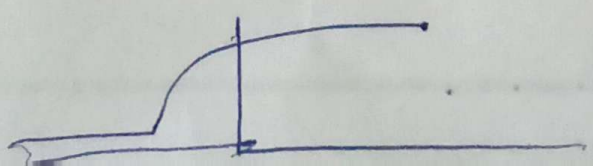
★ Logistic Regression (Supervised) (used to predict cast or not like task)

~~Give~~ X O

Given $x$, want $\hat{y} = P(y=1 \mid x)$

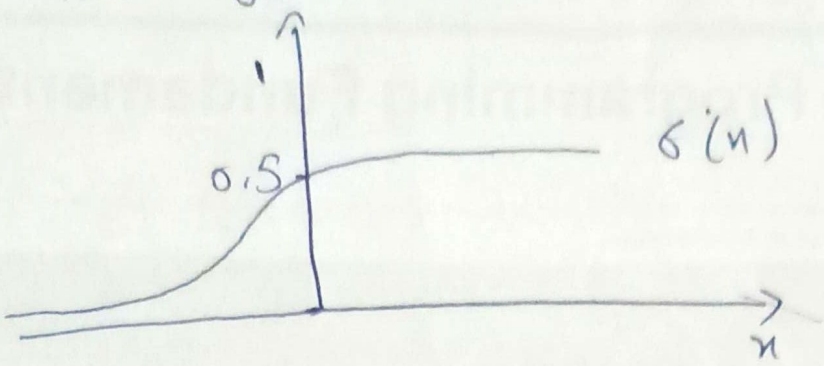
Parameter $w \in R^{n}$, $b \in$ Real num

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



sigmoid

this will form yes or no




$\sigma$ = sigmoid

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

★ If z large $\sigma(z) \approx \frac{1}{1+0} = 1$

★ If z is small $= \frac{1}{1+e^{-z}} \approx \frac{1}{1+\text{Bignum}} \approx 0$

$$\hat{y} = \sigma(\theta^T x)$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{Aix} \end{bmatrix} \} b \longleftrightarrow$$

Just ~~for~~ understan

# Logistic Regression Cost Function

$$\hat{y} = \sigma(w^T x + b) \quad \text{where} \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

---

loss error $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

$$\ell(\hat{y}, y) = -(y \log \hat{y} + (1-y)(\log)(1-\hat{y}))$$

Condition

If $y = 1$: $\ell(\hat{y}, y) = -\log \hat{y}$ ← want $\log \hat{y}$ large

If $y = 0$: $\ell(\hat{y}, y) = -\log(1-\hat{y})$ ← Want $\log(1-\hat{y})$ large

---

cost funtion $= \frac{1}{m} \sum_{i=1}^{m} \ell(\hat{y}^{(i)}, y^{(i)}) = \left[ \frac{-1}{m} \sum^{m} (y^{(i)} \log \hat{y} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}) \right]$

$$= -\frac{1}{m} \sum_{i=1}^{m} \left[ y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}) \right.$$

~~Gradient Descent~~

# Gradient Desent

✱ Cost and loss function difference

The loss function computes the error for single training example, ✱

✱ Cost function

The loss function computes the ~~error~~ average of loss function of the entire training set

Imean, Cost (Σ Loss)

## Gradient descent

$$\text{Loss} = \hat{y} = \sigma(w^T x + b), \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{cost function} = J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = \frac{-1}{m}\sum y^{(i)} \log \hat{y}^{(i)} + (1-y)\log (1-\hat{y}^{(i)})$$

~~Gradient Descent~~

# Gradient Desent

★ Cost and loss function difference

The loss function computes the error for Single training example, ★

★ Cost function

The loss function computes the ~~average~~ average of loss function of the entire training set

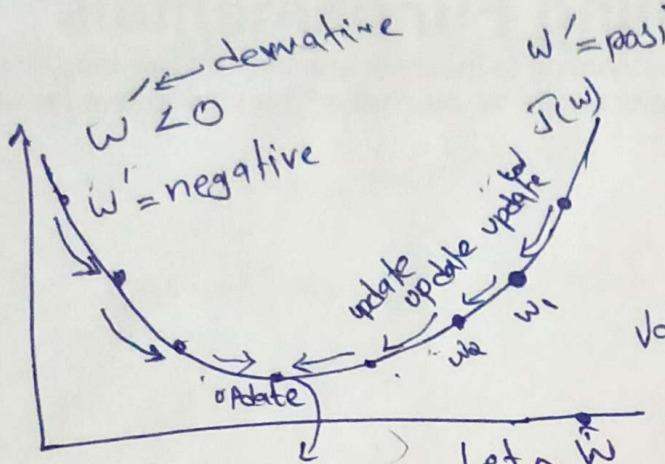
Imean, Cost (Σloss)

## Gradient descent

$$\text{loss} = \hat{y} = \sigma(w^T x + b), \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{cost function} = J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = \frac{-1}{m}\Sigma y^{(i)} \log \hat{y}^{(i)} + (1-y)\log (1-\hat{y}^{(i)})$$

# Gradient Descent




Repead

learning rate

$$w := w - \alpha \frac{d\bar{J}(w)}{dw}$$

value ← w

update

derivative

Deepest point Gibbal on Graph minima

$$w := w - \alpha \frac{d\bar{J}(w)}{dw}$$

⇒ Hence, we get the accurate value by reaching global minima.

$$b := b - \alpha \frac{d\bar{J}(w,b)}{db}$$

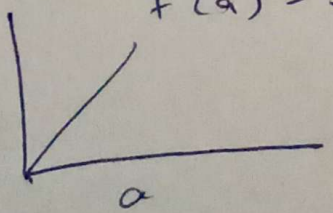
← partial derivative

---

learning derivative

$f(a) = 3a$

If $a = 2 \Rightarrow f(a) = 6$



$\Rightarrow$ Slope = $\frac{\text{height}}{\text{width}}$ of triangle

$\Rightarrow$ derivatives are small values 0.00000000001 it should be tiny

---

Example : 2

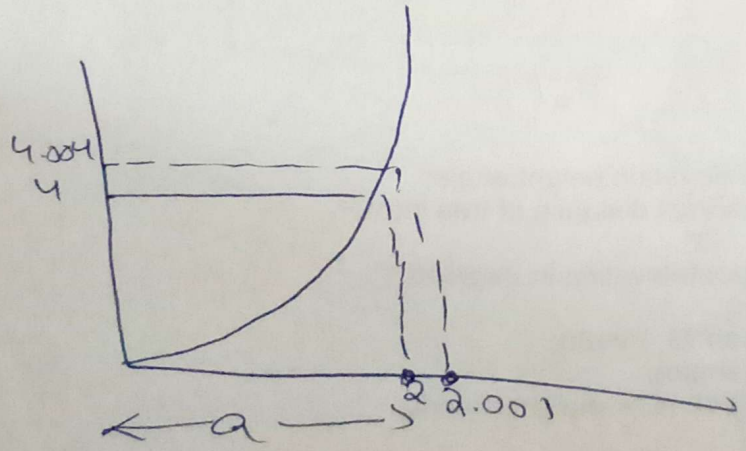
$f(a) = a^2$

$a = 2$

$a = 2.001$

$f(a) = 4$

$f(a) = 4.004$




$\frac{d}{da} a^2 \Rightarrow 2a$

$a = 5 \quad f(a) = a^2 \Rightarrow 25$

$a = 5.001 \Rightarrow f(a) \Rightarrow 25.010 \quad 2(5)$

we got the derivative 25.010 value in 2nd decimal

$f(a) = a^3$

$\frac{d}{da} fa = 3a^2$

let $a = 2$

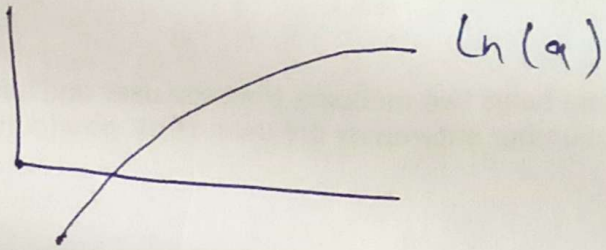
$a = 2.0001$

$f(a) = 2.0012$

---

$f(a) = \log_e(a)$

$= \ln(a)$




$a = 2$

$a = 2.001$

$\frac{d}{da} f(a) = \frac{d}{da} \log a$

$df = \frac{1}{a}$

$f(a)' = \frac{1}{2} = 0.5$

$F(a) = 2.0005$

2.

# Computation Graph

$J(a,b,c) = 3(a+bc) = 33$

$u = bc = 6$, $a=5, b=3, c=2$

$v = a+u = 11$,

$J = 3v = 33$,

$\because J = 3v$ $J' = 3$

$\frac{dJ}{v} = ?$

$v \rightarrow v = 11.001$

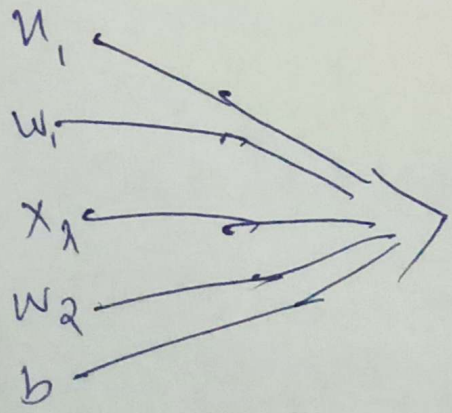
$J = 33.003$

Example $a = 5.001$

$\frac{dJ}{da} =$ $J' =$

dvalue $= a + a'$

## Use case




$Z = w_1 x_1 + w_2 x_2 + b \Rightarrow a = \sigma(z) \rightarrow \mathcal{L}(a,y)$

$da = d\mathcal{L}(a,y)$

$\frac{}{da}$

$= -\frac{y}{a} + \frac{1-y}{1-a}$

$$dz = \frac{df}{dz} = \frac{d\mathcal{L}(a,y)}{dz}$$

$$= a - y$$

$$= \frac{d\mathcal{L}}{da} \cdot \frac{da}{dz} \qquad a(1-a)$$

$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

$dw = x^{(i)}, y^{(i)})$

---

$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y^{(i)})$$

$$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b$$

$$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})$$

$$dw_1^{(i)} = (x^{(i)}, y^{(i)})$$

$J = 0,\ dw_1 = 0,\ dw_2 = 0,\ db = 0$

For $i = 1$ to $m$

$$z^{(i)} = w^T x^{(i)} + b$$

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

$$J_t = -[y^i \log a^i + (1 - y^{(i)}) \log(1 - a^{(i)})]$$

$dz^{(i)} = a^{(i)} - y^{(i)}$

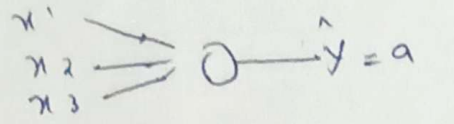
$dw_1 += x_1^{(i)} dz^{(i)}$

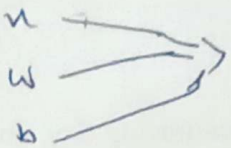
$dw_2 += x_2^{(i)} dz^{(i)}$

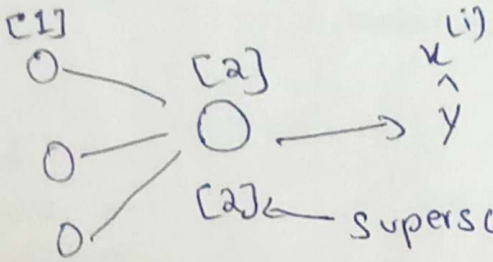
$db += dz^{(i)}$

---

## Shallow + Neural network



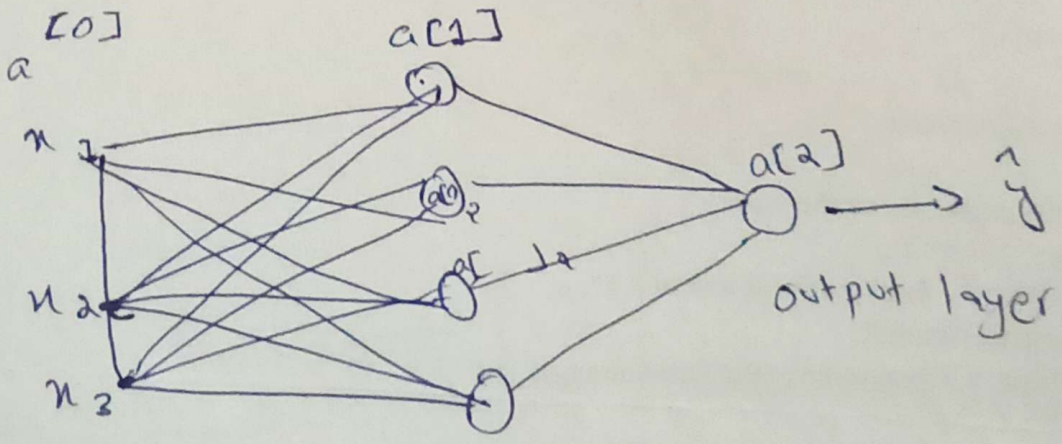




[2] ← superscript brackets.

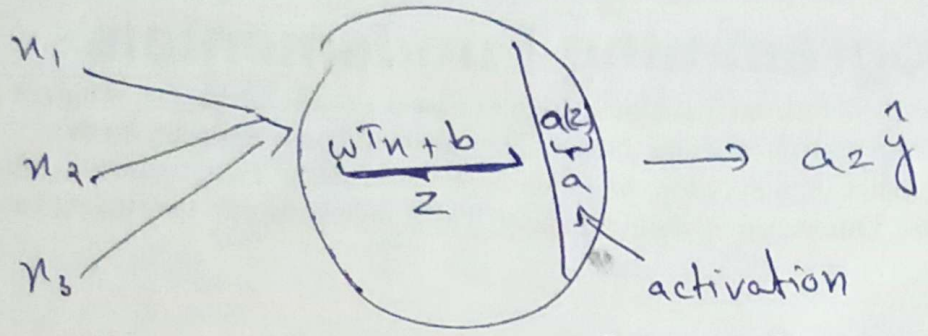
$z^{[1]} = w^{[1]} x + b^{[1]} \rightarrow a^{[1]} = \sigma(z^{[1]}) \rightarrow z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$

## Neural Network representation



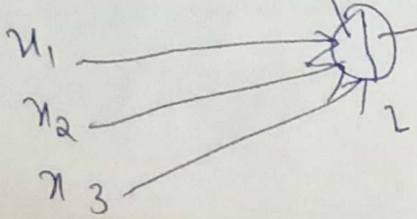

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix}$$

# Representative of Neural network



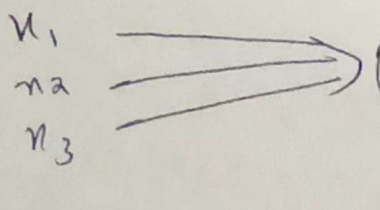

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$




$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$




transpose

$$W = \begin{bmatrix} w_1^{[1]T} x + b \\ w_2^{[2]} \\ w_3^{[1]T} x + b \\ w^{[1]T} x + b \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{bmatrix}$$

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}^{(4,3)} = \sigma(z)^{[1]}$$

## Vectoring across multiple example.

Layer working on example

$a^{[2](i)}$

Layer → $a^{[2]}$ working on example $(i)$

for i = 1 to m:

$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$

$a^{[1](i)} = g(z^{[1](i)})$

$z^{[2](i)} = W^{[2]} a^{[1](i)} + b$

$a^{[2](i)} = g(z^{[2](i)})$

a → activation

---

$z^{[1](1)} = W^{[1]} x^{(1)} + b^{(1)}$

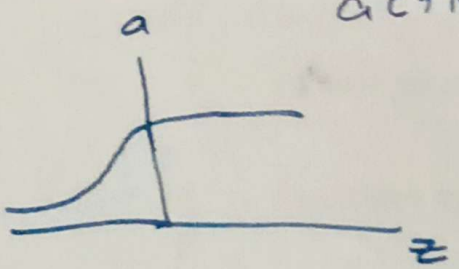
for activation

$A^{[1]} = g(Z^{[1]})$

$z^{[1](2)} = W^{[1]} a^{[1](1)} + b$

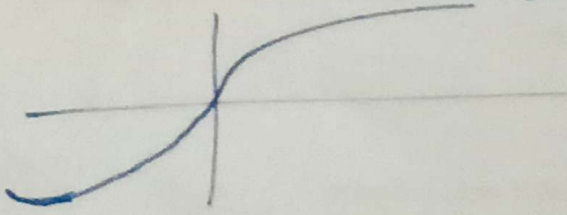
---

One hidden activation layer. Choice of activation function:-



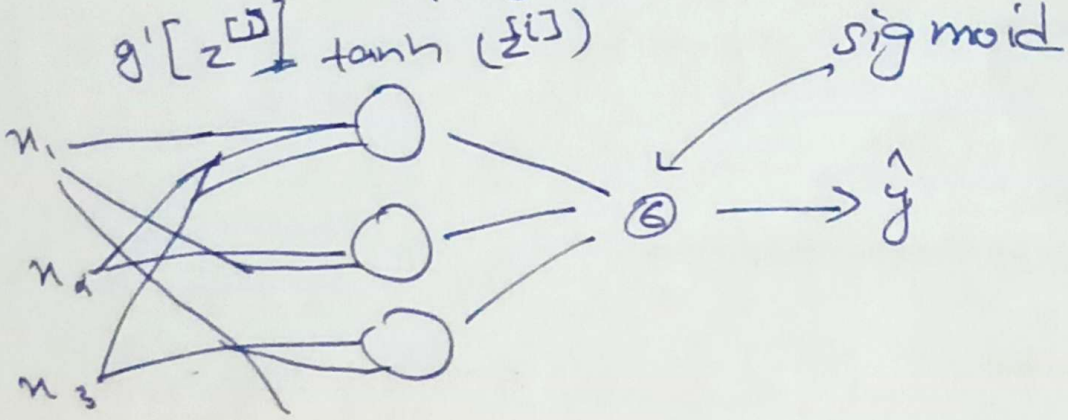

$a = \frac{1}{1+e^{-z}}$

$a = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



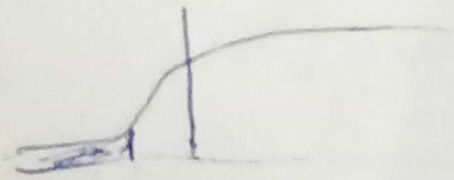
$g(z^{[i]}) \Rightarrow = \tanh(z^{[i]}$ is superior than $a = \frac{1}{1+e^{-a}}$

$g'[z^{[1]}]$ tanh $(z^{[i]})$




Limitation of 6 function and tan func

$\Rightarrow$ When value is very Large or very small it cause gradient very large or small

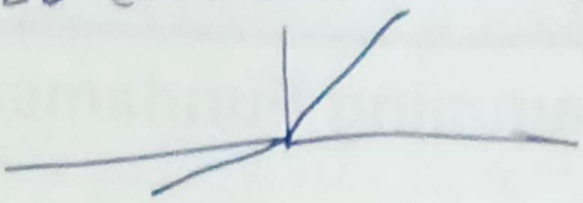


Solution:- Rectify Linear Unit (RLU)

$z^+ \Rightarrow$ ~~when~~ $z = 1$ $\quad \therefore a = max(0, z)$

$z^- \Rightarrow z = 0$

Leaky RLU (another formula)



$\max(0.01, z)$

Q Why we need activation function in neural network

$$a = g(z^{[1]})$$

$$\Rightarrow a^{[2]} = w^{[2]}\left(w^{[1]}x + b^{[1]}\right) + b^{[2]}$$

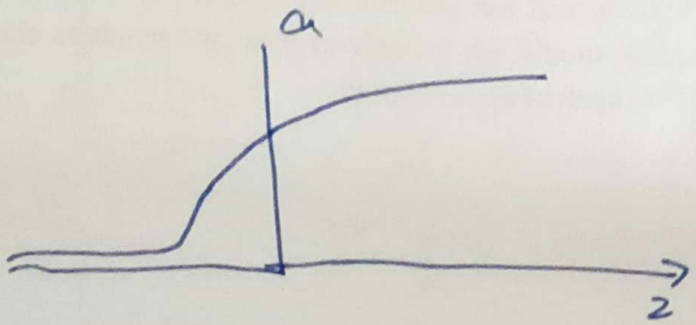
$$\Rightarrow (w^{[2]}w^{[1]})x + \underbrace{(w^{[2]}b + b^{[2]})}_{b}$$

$$= w'x + b$$

$$= w'x + b$$

---

Derivative of activation function




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

$$\frac{d}{dt}g(z) = \text{slope of } g(x) \text{ at } z$$

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

$$= g(z)(1 - g(z))$$

$= g(z)(1 - g(z)) \Rightarrow z = 10 . g(z) \approx 1$

$\frac{d}{dz} g(z) \approx 1(1-1) = 0$

---

$z = -10 \; g(z) \approx 0$

---

$\frac{d}{dz} g(z) = 0.(1-0) = 0$

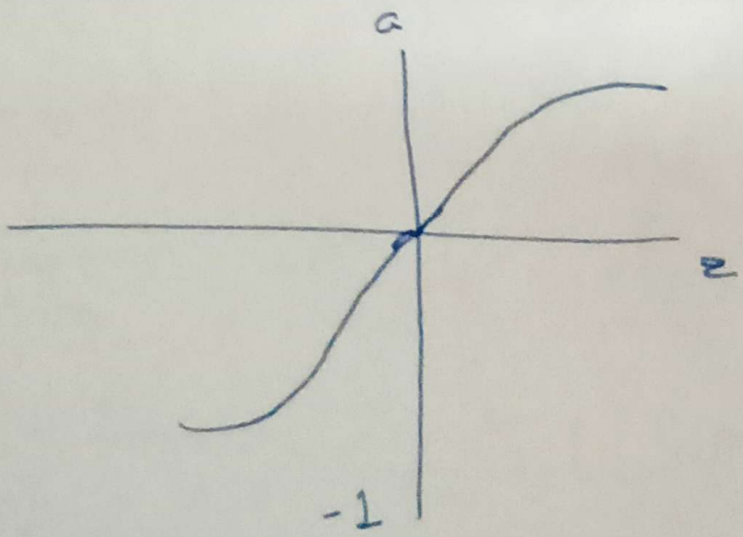
---

$z = 0 \quad g(z) \quad \frac{1}{2}$

$\frac{d}{dz} g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$

---

derivative of tan(h) activation function:-




$\dot{g}(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z$

$= 1 - (\tan(z))^2$

# ReLU and Leaky RELU

$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

ReLU

Leaky ReLU

$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases}$$

---

Topic:- Gradient descent for neural network

Parameters : $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}$

Cost function = $\frac{1}{m} \sum L(\hat{y}, y)$ (loss function)

$\hat{y} \rightarrow a^{[2]}$

Repeat { $(\hat{y}^{(i)}, i = 1 \ldots n)$

$dw = w^{[i]} - \alpha\, dw^{[i]}$

$b = b^{[1]} - \alpha\, db^{[2]}$

formula for propagation in Python

$$z^{[1]} = W^{[1]} x + b^{[1]}$$

$$A^{[2]} = g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$$

$$dz^{[2]} = A^{[2]} - Y$$

$$dw^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dz^{[2]}, axis = 1, keepdims = True$$

---

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$$

---

Backpropagation (Topic) (goes backward to find $w, b$ from $L(a,y)$)

$x$, $w$, $b$ $\rightarrow$ $z = w^T x + b \rightarrow a = \sigma(z) \rightarrow L(a,y)$ (loss)

chain rule

$dz = da \cdot g'(z)$

$\frac{dL}{dz} = \frac{dL}{da} \cdot \frac{da}{dz}$

"$dz$" = "$da$" $\cdot$ d

$\frac{d}{da} L(a,y) = -y \log a - (1-y) \log(1-a)$

$= \frac{-y}{a} + \frac{1-y}{1-a}$

Q# Back propagation

- $L(a, y)$

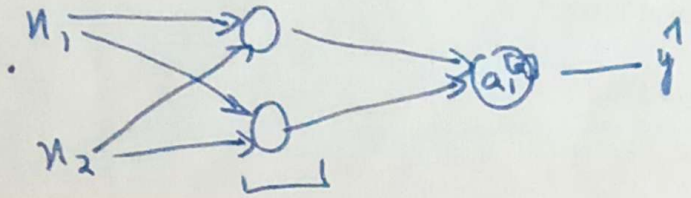
$$dz^{[2]} = a^{[2]} - y$$

$$dw = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

---

One hidden layer




hidden layer

---

$$Z = wx + b$$

where $w$ is randomly initialized

$w = np.random.rand(n\text{-outputs}, n\text{-inputs}) \times 0.01$

# Deep neural network

★ more than 3 hidden layers

Shallow neural layer

★ 2 hidden layer

## Top) Forward propagation

first layer $z^{[1]} = w^{[1]} x + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$

2nd layer = $z^{[2]} = w^{[2]} x^{[1]} + b^{[2]}$

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

$z^{[4]} = w^{[4]} x^{[3]} + b^{[4]}$

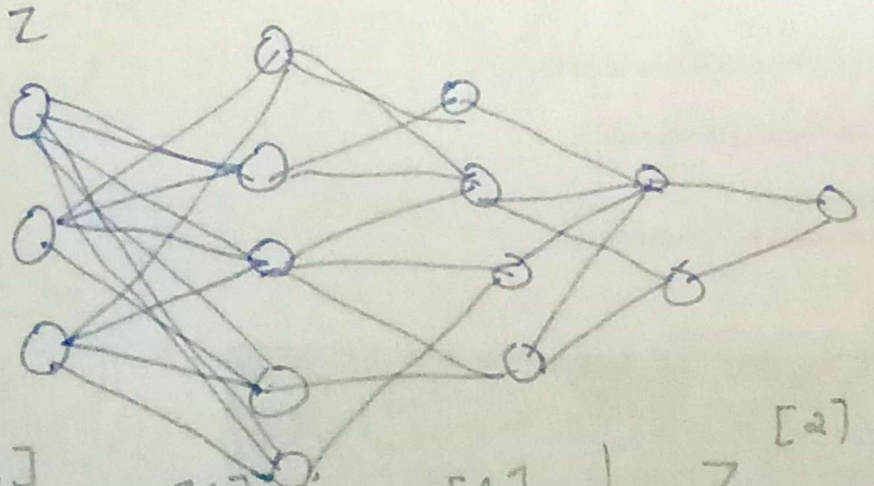
$a^{[4]} = \sigma[z^{[4]}]$

$z^{[l]} = w^{[l]} x^{[l-1]} + b^{[l]}$

$a^{[l]} = g^{[l]}(z^{[l]})$

---

## Top) ~~Dimension~~ Getting or dimension right.




$z^{[1]} = w^{[1]} x + b^{[1]}$ → [3,1]

$(3,1) = (3,2)\ (2,1)$

$[\vdots]\ (n,2)$

$Z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$

[5,3], ↑

$= [3,1]$

$$w^{[l]} = (n^{[l]}, n^{[l-1]})$$

$$b^{[l]} = [n^{[l]}, 1$$

---

# Deep Neural Networks Representation

Forward propagation:-

Input $a^{[l-1]}$

output $a^{[l]}$, cache ($z^{[l]}$

→ use to store intermediate computation of one layer in back propagation

Back propagation

Input $da^{[l]}$

output $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$

$$\Rightarrow dz^{[l]} = da^{[l]} * g^{[l]}(z^{[l]})$$

$$\Rightarrow dw^{[l]} = dz^{[l]} \cdot a^{[l-1]^T}$$

$$\Rightarrow db^{[l]} = dz^{[l]}$$

$$\Rightarrow da^{[l-1]} = w^{[l]^T} \cdot dz^{[l]}$$

$$\Rightarrow dz^{[l]} = w^{[l+1]^T} dz^{[l+1]} * g^{[l]}(z^{[l]})$$

$$dz^{[l]} = dA^{[l]} * g^{[l]'}(z^{[l]})$$

$$dw^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T}$$

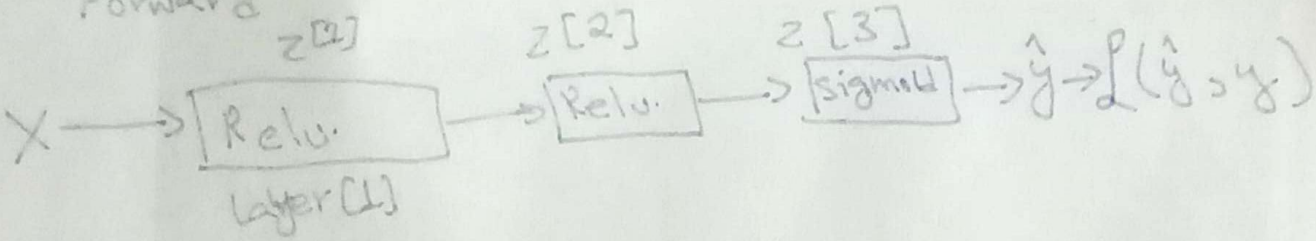
$$db^{[l]} = \frac{1}{m} np.sum(dz^{[l]}, axis=1, keepdims=True)$$

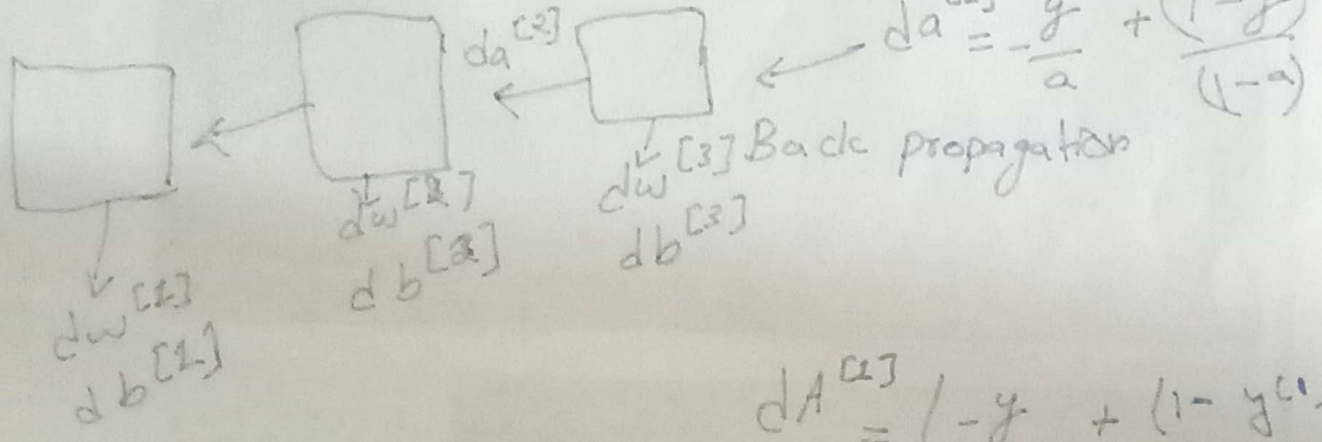
$$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

---

Summary:

Forward




$$dA^{[1]} = \left( \frac{-y}{a^{[1]}} + \frac{(1-y^{(1)})}{1-a^{(1)}} \right)$$

$$dz^{[l]} = dA^{[l]} * g^{[l]'}(z^{[l]})$$

$$dw^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T}$$

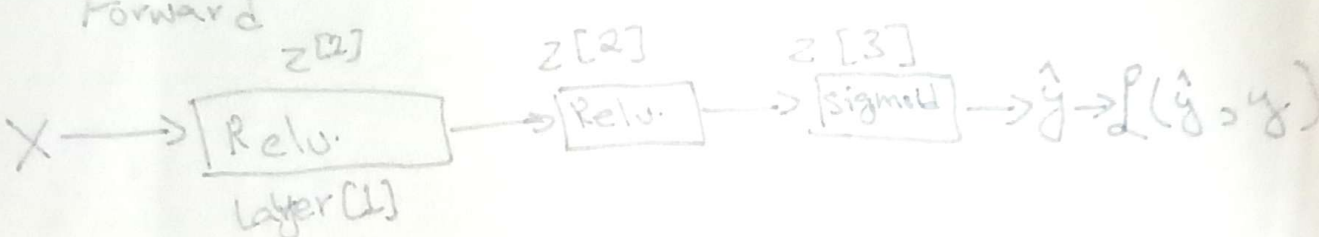
$$db^{[l]} = \frac{1}{m} np.sum(dz^{[l]}, axis=1, keepdims=$$

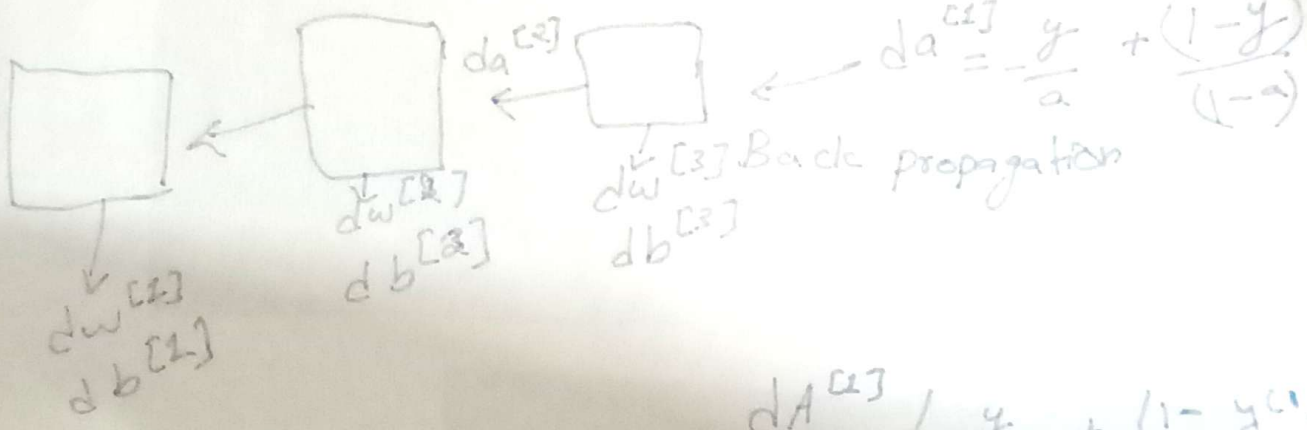
$$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

---

Summary:

Forward




$$dA^{[2]} = \left( \frac{-y}{a^{[2]}} + \frac{(1-y^{(1)})}{1-a^{[2]}} \right)$$