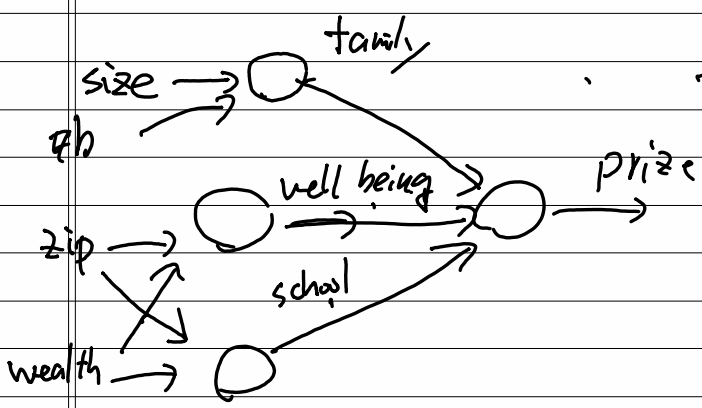
Coursera.

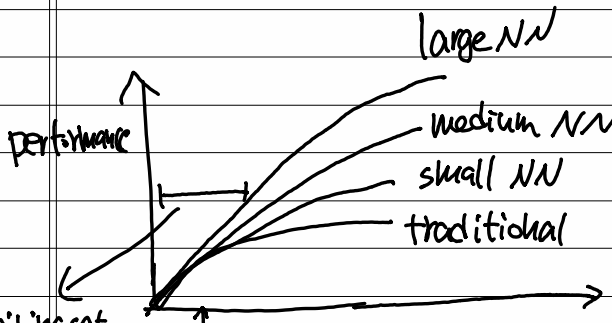
good intuition



but usually full-connected

structured data / unstructured data
table image, video, audio...

DNN outperforms



Data
computation
Algorithm

W2

m training samples $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})\}$

$$X = \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix} \; n_x \text{ features}$$

m = # training sample

$X \in \mathbb{R}^{n_x \times m}$

$X.shape = (n_x, m)$

$$Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(m)} \end{bmatrix}_{1 \times m}$$

$Y \in \mathbb{R}$

★ stack in column for easier implementation

Logistic regression

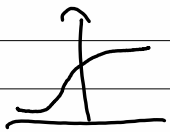
Given X, want $\hat{y} = P(y=1|x)$

$X \in \mathbb{R}^{n_x \times 1}$

parameter: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

sigmoid function σ

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



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

$z \nearrow \; \sigma(z) = \frac{1}{1+0} = 1$

$z \searrow$ negative $\; \sigma(z) = \frac{1}{1+\text{big}} = 0$

COST FUNC

cost func

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

want $\hat{y}^{(i)} \simeq y^{(i)}$

loss:

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

loss func

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

$y=1 \rightarrow L(\hat{y}, y) = -\log(\hat{y})$ want $-\log(\hat{y})$ ↓

$\log(\hat{y})$ ↑

$\hat{y}$ large

$y=0 \rightarrow L(\hat{y}, y) = -\log(1-\hat{y})$ want $\log(1-\hat{y})$ ↑

$\hat{y}$ small

$\hat{y} = (0, 1)$

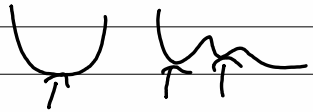
cost func $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

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

GRADIENT DESCENT

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

convex non-convex



$J(w)$

$$w := w - \underbrace{\alpha}_{\text{learning rate}} \frac{dJ(w)}{dw}$$

$J(w,b)$

$$w := w - \alpha \frac{d\,J(w,b)}{dw} \qquad \frac{\partial J(w,b)}{\partial w}$$

(partial derivative symbol

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

∂ : partial (for multiple var)

d : derivative

DERIVATIVE

$$\text{slope} = \frac{h}{w}$$

$f(a) = a^2$

$\frac{df(a)}{da} = 2a$

increase a by 0.1

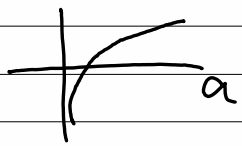
$y \uparrow 2 \times 0.1 = 0.2$

$f(a) = 3a$

$\frac{df(a)}{da} = 3$

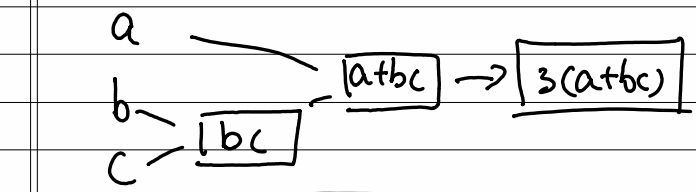
nudge $a \rightarrow$, y increase $2a$

$f(a) = \log(a)$ $\frac{d}{da} f(a) = \frac{1}{a}$

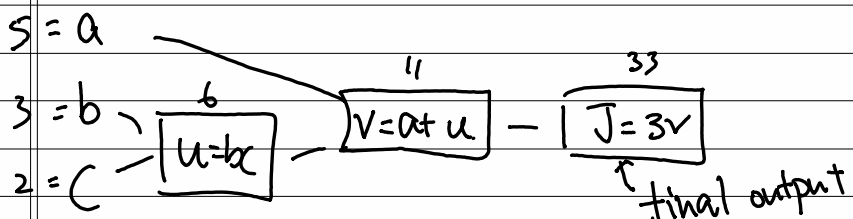


Computation Graph

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



Derivatives with computation graph



$\frac{dJ}{dv} = 3$

$V = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

diff: $0.001 \times 3 = 0.003$

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

$a = 5 \rightarrow 5.001$

$V = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$\rightarrow \frac{dv}{da} = 1$

$$\frac{dJ}{da} = \frac{dJ}{dv}\frac{dv}{da}$$ chain rule

Coding convention

$\frac{d\,\text{finaloutputvar}}{d\,\text{var}} \rightarrow$ dvar

~ derivative of final output var.

backpropagation

$\frac{dJ}{da} = da = 3$

$5 = a$

$3 = b$

$2 = c$

$u = bc$ (6), $du = 3$

$v = a + u$ (11), $dv = 3$

$J = 3v$ (33)

$$\frac{dJ}{du} = \frac{dJ}{dv} \cdot \frac{dv}{du} = 3 \cdot 1 = 3$$

$$\frac{dJ}{db} = \frac{dJ}{dv} \cdot \frac{dv}{du} \cdot \frac{du}{db} = 3 \cdot 1 \cdot 2 = 6$$

$$\frac{dJ}{dc} = \frac{dJ}{du} \cdot \frac{du}{dc} = 3 \cdot 3 = 9$$

$u = 6 \rightarrow 6.001$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$b = 3 \rightarrow 3.001$
$u = 6 \rightarrow 6.002$
$v = 11 \rightarrow 11.002$
$J = 33 \rightarrow 33.006$

Logistic regression gradient descent

$z = w^T x + b$

$\hat{y} = a = \sigma(z)$

single sample — $L(a, y) = -(y \log(a) + (1-y)\log(1-a))$

$z = w_1 x_1 + w_2 x_2 + b \rightarrow a = \sigma(z) \rightarrow L(a, y)$

$da = \frac{dL(a,y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$

$dz = \frac{dL}{dz} = \frac{dL(a,y)}{dz} = \frac{dL}{da} \cdot \frac{da}{dz}$

$= a - y$ ($a(1-a)$ — sigmoid derivative)

$\frac{dL}{dw_1} = \frac{dL}{dz} \cdot \frac{dz}{dw_1} = x_1 \cdot \frac{dL}{dz}$

m traing example

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

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

to calculate → $dw_1^{(i)}, dw_2^{(i)}, db^{(i)}$

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

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

$\frac{\partial}{\partial w_2} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial w_2} (a^{(i)}, y^{(i)})$

- vectorization

CODE

$$\frac{dL(a,y)}{dz} = a - y$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(a^{(i)}, y^{(i)}) \qquad \frac{dJ}{dw_1} \quad \frac{dJ}{dw_2} \quad \frac{dJ}{db}$$

for each sample

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

$dw_1 += x_1 \cdot (a - y)$

$dw_2 += x_2 \cdot (a - y)$

for $i = 1$ to m

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

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

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

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

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

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

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

$J /= m$

$dw1 /= m$

$dw2 /= m$

$db /= m$

vectorization / vectorized logistic regression

$z^{(1)} = w^T x^{(1)} + b \quad z^{(2)} = w^T x^{(2)} + b$

$a^{(1)} = \sigma(z^{(1)}) \quad a^{(2)} = \sigma(z^{(2)}) \quad \cdots$

$X \qquad X \in \mathbb{R}^{n \times m}$

$[z^{(1)} \dots z^{(m)}] = w^T X + [b \dots b]$

$1 \times m \qquad 1 \times n \quad n \times m \quad 1 \times m$

$= [w^T x^{(1)} + b \quad w^T x^{(2)} + b \quad w^T x^{(3)} + b \dots]$

$z = np.dot(w^T, X) + b$

z scalar

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

CODE

one iter of gradient descent

$z = w^T x + b$

$= np.dot(w.T, x) + b$

$a = \sigma(z)$

$dz = A - Y$

$dw = \frac{1}{m} X dz^T$

$db = \frac{1}{m} np.sum(dz)$

$w := w - \alpha dw$

$b := b - \alpha db$

Broadcasting

$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + 100 \rightarrow \begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} = \ldots$

$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100 & 200 & 300\end{bmatrix} \rightarrow \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix} = \ldots$

(m,n) (1,n) ~~~ (m,n) (m,n)

$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100\\200\end{bmatrix} \rightarrow +$

(m,n) (m,2) (m,n) - m,n = ...

(m,n) $\begin{matrix}+\\-\\*\\\div\end{matrix}$ (1,n) ~ (m,n) broadcast

(m,1) ~ (m,n) broadcasting

DEBUG

avoid "rank-1" array.

randn(5) - DON'T USE

randn(5,1) - column vector

randn(1,5) row vector

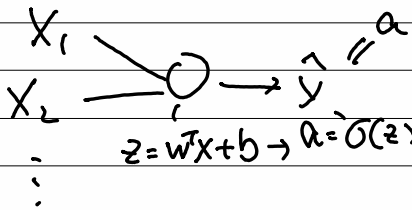
☆ → assert(a.shape = (5,1))

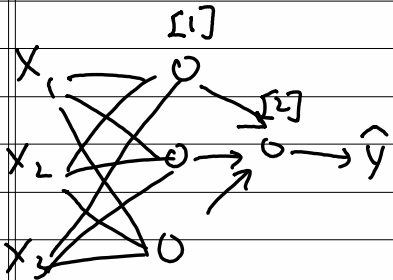
△ (5,).reshape((5,1))

NN

WEEK 3

Logistic regression





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

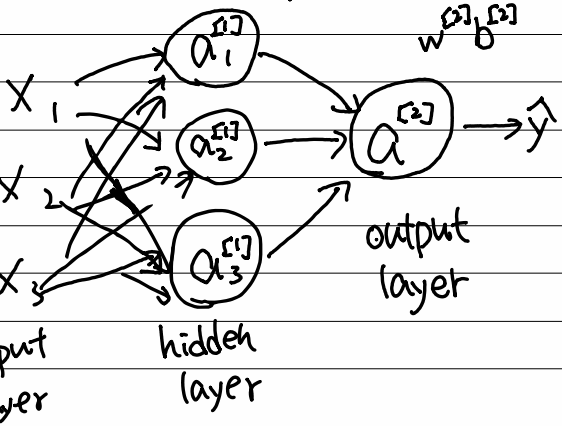
$\rightarrow a^{[2]} = \sigma(z^{[2]})$

$\rightarrow L(a^{[2]}, y)$

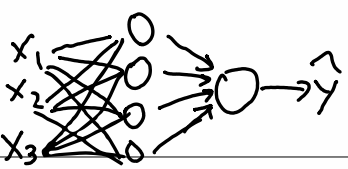
2-layer NN

layer

$a^{[0]} = x$ $\quad a^{[1]}$ $\quad w^{[1]}, b^{[1]}$ $\quad w^{[2]}, b^{[2]}$



$a_i^{[l]}$ — l: layer, i: node in layer



Computing NN output

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

neuron

vectorize

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$\begin{bmatrix} — w_1^{[1]} — \\ — w_2^{[1]} — \\ \vdots \\ — w_4^{[1]} — \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ \vdots \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \cdot x + b_1^{[1]} \\ \vdots \\ w_4^{[1]T} \cdot x + b_4^{[1]} \end{bmatrix}$$

4×3 $\quad 3 \times 1$

$$W^{[1]} \qquad b^{[1]}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

CODE

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

m samples

$$X \longrightarrow a^{[2]} = \hat{y}$$
$$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$$
$$\vdots$$
$$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$$

for $i = 1$ to m

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

$\vdots$

vectorize
m training

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$n \times m$.

$$z^{[1]} = W^{[1]} X + b^{[1]}$$
$$A^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(z^{[2]})$$

$$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

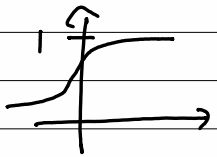
$$A^{[1]} = \begin{bmatrix} | & & | \\ a^{[1](1)} & \cdots & a^{[1](m)} \\ | & & | \end{bmatrix}$$

* different node in layers ↑
different training sample →

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

ACTIVATION

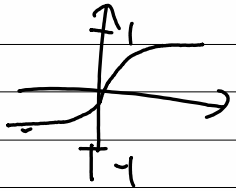
$\sigma = \frac{1}{1+e^{-z}}$



prob only for output [0,1]

$g = \tanh(z)$

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



almost always better for hidden layers

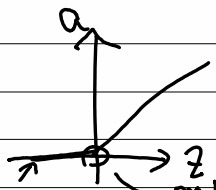
con:

- large z → small gradient → slow

DEFAULT **ReLU**

Rectified Linear Unit

$a = \max(0, z)$

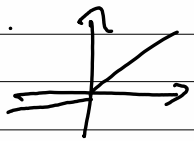


problem in theory only

pro: learn faster!

- bc not affected by small slope
- usually work w/ deep nets w/ $z > 0$

Leaky ReLU



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

Derivative of Activation Function

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

prime $g'(z)$

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

g prime of z

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

$$\frac{d}{dz} e^{-z} = -z \cdot e^{-z}$$

* Power rule applies when numerator is the power of derivcator

$$\frac{d}{dx} x^n = n \cdot x^{n-1}$$

$$\frac{d}{dx} f(x)^n = \frac{d}{df(x)} f(x)^n \cdot \frac{df(x)}{dx}$$

$$= n \cdot f(x)^{n-1} \cdot f'(x)$$

$$= -\frac{d}{dz}(1+e^{-z}) \times (1+e^{-z})^{-2}$$

$$= -e^{-z} \cdot (1+e^{-z})^{-2}$$

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

$$\frac{d}{dx} f(x)^{-1} = -f(x)^{-2} \cdot f'(x)$$

$$= \frac{-f'(x)}{f(x)^2}$$

$$\frac{d}{dx} \frac{g(x)}{f(x)}$$

$$= \frac{d}{dx} g(x) \cdot f(x)^{-1}$$

---

tanh

$$g(z) = \tanh(z)$$

$$\frac{d}{dz} \tanh(z) = 1 - (\tanh(z))^2$$

ReLU

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

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

implement NN

Forward Prop

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

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

$z^{[2]} = W^{[1]} A^{[1]} + b^{[2]}$

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

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

Back Prop

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

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

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

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

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

$db^{[1]} = \frac{1}{m} np.sum(dz^{[1]}, axi=1, keepdim=true)$

Initialization

× all w = 0

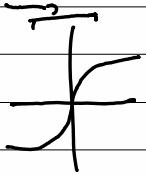
- all hidden unit will be the same

$w^{[1]} = np.random.randn(2,2) * 0.01$ → in this range

$b^{[1]} = np.zero(2,1)$

:

$b[2] = 0$



W3 assignment

planar

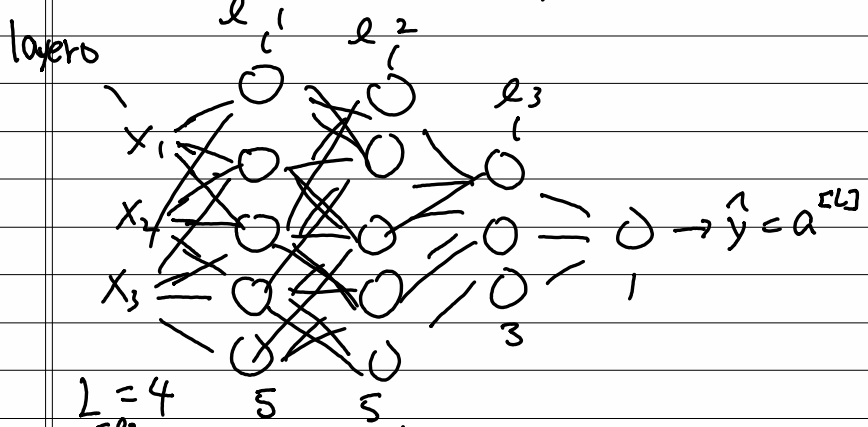
X [x_1 / x_2 ...] 2xm

Y [ ] 1xm label

W4

DNN - Deep NN

shallow vs deep



$n^{[l]}$ = # unit in layer l

$n^{[1]}=5$, $n^{[2]}=5$, $n^{[3]}=3$, $n^{[4]}=1$

$n^{[0]} = n_x = 3$

$a^{[l]}$ = activation in layer l

$a^{[l]} = g^{[l]}(z^{[l]})$ $w^{[l]}$ $b^{[l]}$ = weights for $z^{[l]}$

Forward Prop

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

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

$$\Downarrow$$

vectorize

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

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

$$\downarrow$$

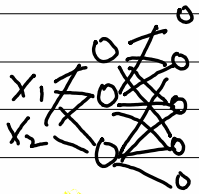
$$\hat{Y} = g(Z^{[4]}) = A^{[4]}$$

$z^{[l]}$

$\{ z^{[l](1)} \cdots z^{[l](m)} \}$

$Z^{[l]}$

DEBUG DNN



$$Z^{[1]} = W^{[1]} \cdot X + b$$

$(3,1)$ $(2,1)$

so, W should be $(3,2)$

$$\Rightarrow (n^{[1]}, 1) = (n^{[1]}, n^{[0]})(n^{[0]}, 1)$$

$$Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$$

$(5,1)$ $(3,1)$

$\Downarrow$

so. $(5,3)$

$$\vdots$$

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

$$b^{[l]} : (n^{[l]}, 1)$$

$$dw^{[l]} : (n^{[l]}, n^{[l-1]})$$

$$db^{[l]} : (n^{[l]}, 1)$$

VECTORIZED

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

$(n^{[1]}, 1)\ (n^{[1]}, n^{[0]})\ (n^{[0]}, 1)\ \ (n^{[1]}, 1)$

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

$(n^{[1]}, m)\ (n^{[1]}, n^{[0]})\ (n^{[0]}, m)\ \ (n^{[1]}, 1)$

Python broadcasting

↓

$(n^{[1]}, m)$

$Z^{[l]}, A^{[l]} : (n^{[l]}, m)$

$l = 0,\ A^{[0]} = X : (n^{[0]}, m)$

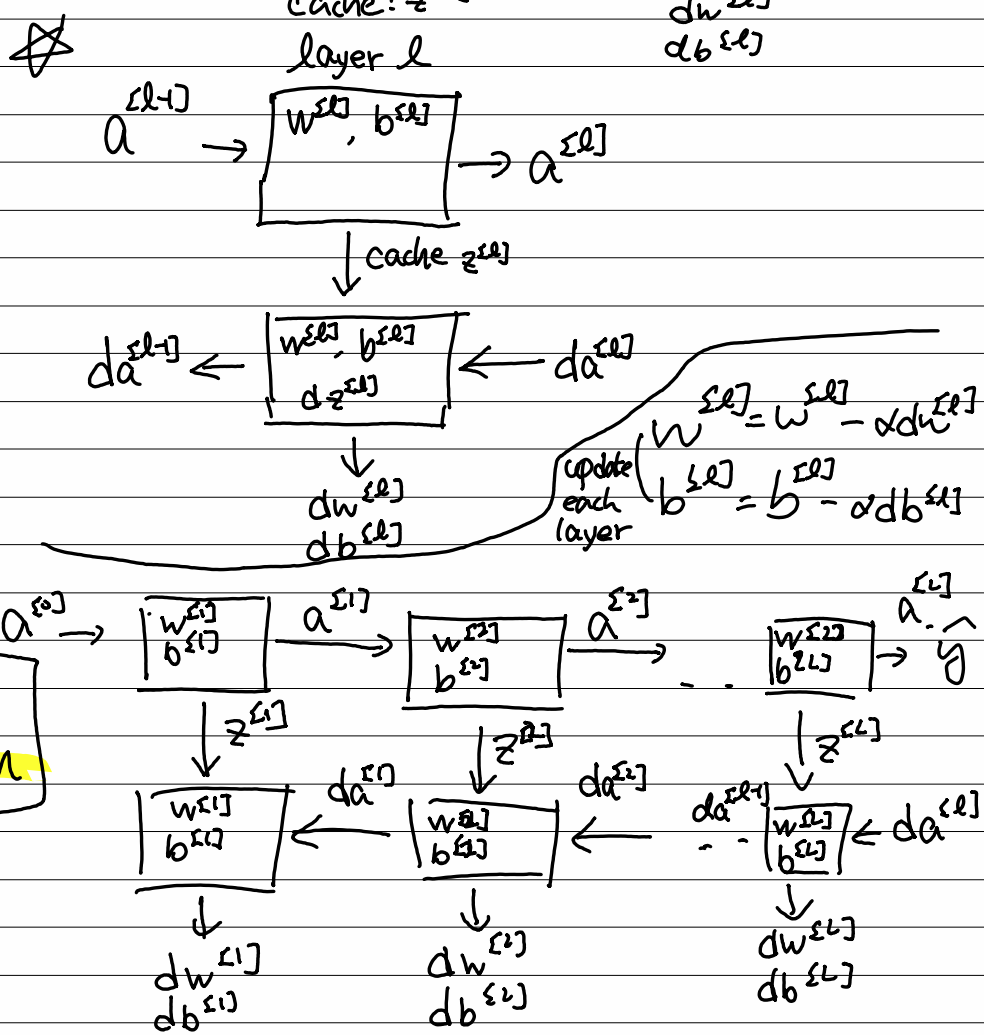
$dZ^{[l]}, dA^{[l]} : (n^{[l]}, m)$

Forward: input $a^{[l-1]}$, output $a^{[l]}$

$z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$, cache $z^{[l]}$

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

Backward: input $da^{[l]}$, output $da^{[l-1]}$



IMPLEMENTATION

BACKPROP

- input: $da^{[l]}$

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

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

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

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

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

$$dz^{[l]} = w^{[l+1]T} dz^{[l+1]} * g^{[l]'}(z^{[l]})$$

$$\Delta\ da^{[L]} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$$

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

HYPERPARAMETER

- learning rate
- iteration
- hidden layer L
- hidden unit $n^{[1]}, n^{[2]}$
- activation func

→ parameters

$W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]} \dots$

look at cost function graph to decide hyperparameter

trial-and-error.