Agenda

1) Problems in Training NN -
2) Weight initialization & activation func" basics
3) Act function.
4) Transfer learning
5) Batch Normalization

Remaining topic

(1) Optimizers
(2) Regularization
(3) Loss function

1)

i) Vanishing & Exploding Gradients

$W = W - \eta \underbrace{\nabla C}_{\uparrow\uparrow}$

Gradients ↓↓ → $\nabla C \downarrow\downarrow$

Grad ↑↑ → $\nabla C \uparrow\uparrow$

unstable Gradients => you'll never reach to a good solution

⇓

Solution will not converge

ii) NN requires lot of data to train

2M parameters → MNIST. 2 6 6 6 1 0 —

Sol": Transfer learning or data augmentation.

iii) For complex problem statement:

Increase size of NN => increase in no of hidden layers

=> slow training

Sol": Choose a better optimizer & activation f".

iv) Risk of overfitting.

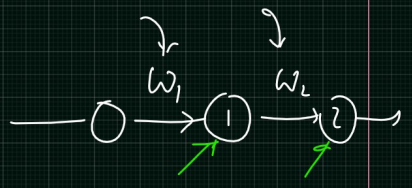
→

solution : {Regularization → Dropout}

Observation :-

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} \quad -(1)$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \quad -(2)$$



$z_2 = w_2 a_1 + b$

These are product of ratios

1st case :- all the ratio $\ll 1$

eqn. (1) $\frac{\partial e}{\partial w_2} = 0.1 \times 0.2 \times 0.3 \simeq \underline{0.006}$

$$w_2 = w_2 - \eta \times 0.006$$

$$w_2 = w_2 - \underbrace{0.1 \times 0.006}_{\simeq 0}$$

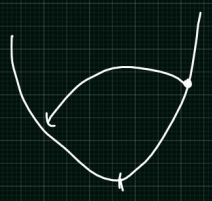
⇒ Vanishing gradient ⇒ negligible weight update

⇒ leads to train lower layer very slow

2nd case : product terms $\gg 1$

$$\frac{\partial e}{\partial w_2} = 10 \times 20 \times 30 = 6000$$

$$w_2 = w_2 - \eta\, 6000$$



⇒ Exploding gradient issue

⇒ solution will diverge

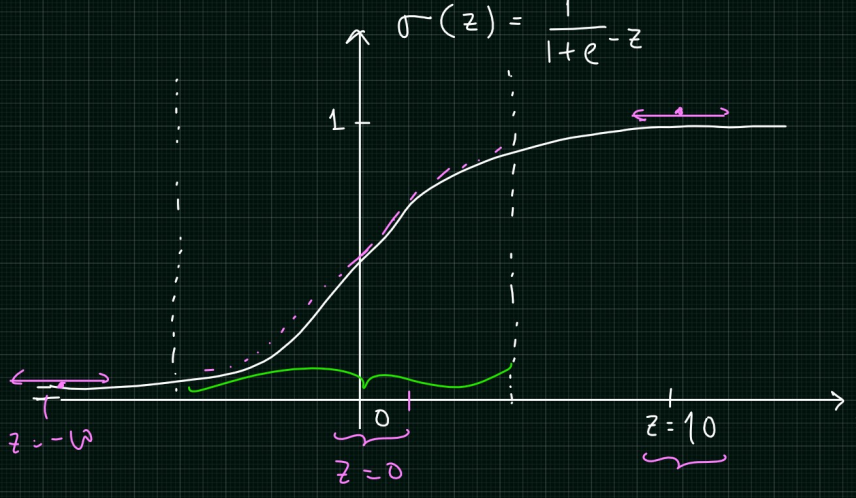
for Vanishing & Exploding gradient.

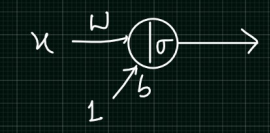
it depends on choice of activation funct.ⁿ
&
Weight initialization technique

in 2010 by
Xavier Glorot & Yoshua Bengio
in their paper
"Understanding the difficulty of training deep feed forward NN"

⇓

each layer learn at different speed.





Case 1. Initialized weight

$W = 10$ assume $\rightarrow x = 1$

$z = W \cdot x = 10 \times 1 = 10$

$$\sigma(z=10) = \frac{1}{1+e^{-10}}$$

$$= \frac{1}{1+0} = 1$$

$e \simeq 2.73 \simeq 3$

$3^{-10} = \frac{1}{3^{10}} \simeq 0$

at $z = 10$

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)\{1-\sigma(z)\}$$

$$\sigma'(z=10) = \sigma(z=10)\{1-\sigma(z=10)\}$$

$$= 1\{1-1\}$$

$$= 0$$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2}^{0} \cdot \frac{\partial z_2}{\partial w_2} = 0$$

$\Rightarrow$ zero weight update

Case 2: Weight init-

$w = -10$ assume $x = 1$

$z = -10 \times 1 = -10$

$$\sigma(z=-10) = \frac{1}{1+e^{-(-10)}} = \frac{1}{1+e^{10}} \simeq 0$$

$$\sigma'(z=-10) = \sigma(z=-10)\{1-\sigma(z=-10)\}$$

$$= 0$$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2}^{0} \cdot \frac{\partial z_2}{\partial w_2} \simeq 0$$

$\Rightarrow$ zero weight update

Case 3.

$w = 0$

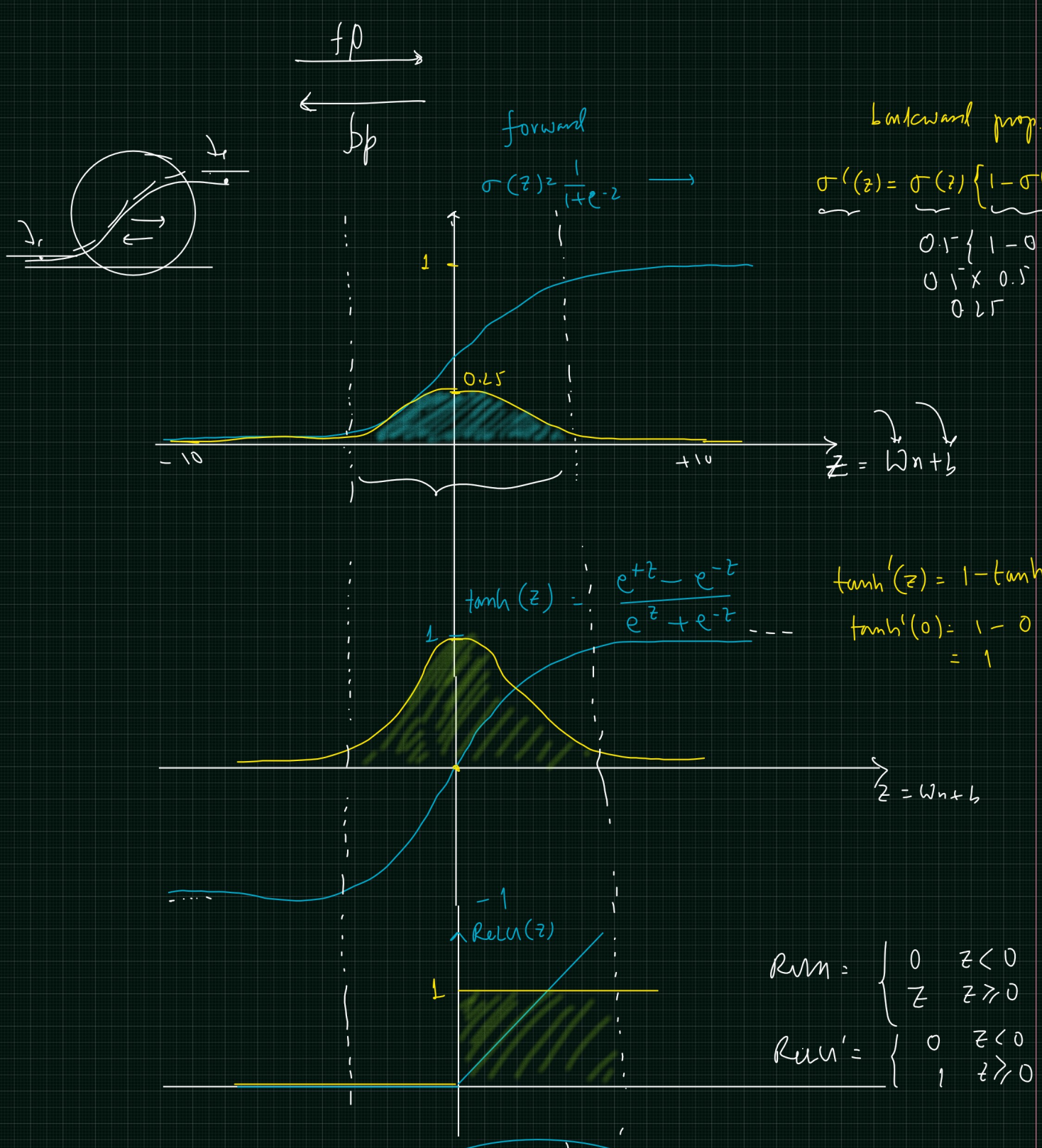
$z = 0$

$$\sigma(z=0) = \frac{1}{1+e^{0}} = \frac{1}{2} = 0.5$$

$$\square'(z=0) = \sigma(z=0)\{1-\sigma(z=0)\}$$

$$= 0.5(1-0.5)$$

$$= 0.25 \neq 0$$


fp
bp
forward
backward prop.
$\sigma(z) = \frac{1}{1+e^{-z}}$
$\sigma'(z) = \sigma(z)\{1-\sigma(z)\}$
0.1 { 1 - 0.5 }
0.1 x 0.5
0.25
1
0.25
-10
+10
$z = wx+b$
$\tanh(z) = \frac{e^{+z} - e^{-z}}{e^{z} + e^{-z}}$
$\tanh'(z) = 1 - \tanh^2(z)$
$\tanh'(0) = 1 - 0$
$= 1$
1
$z = wx+b$
-1
ReLU(z)
1
$\text{ReLU} = \begin{cases} 0 & z<0 \\ z & z \geq 0 \end{cases}$
$\text{ReLU}' = \begin{cases} 0 & z<0 \\ 1 & z \geq 0 \end{cases}$
$\sigma(z)$
$\hat{y} = \sigma(wx+b)$
z
$\sigma(wx+b)$
x
w
b
$z = wx+b$
$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w}$ - ①
$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial b}$ - ②
$w = w - \eta \partial e/\partial w$
$b = b - \eta \partial e/\partial b$

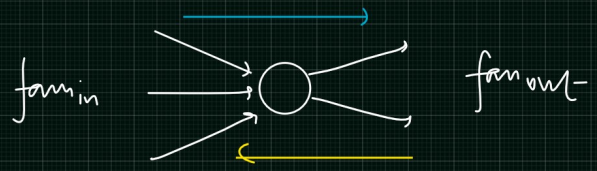
As per the paper,

for proper flow of information in forward as well as backward direction :-

info ≠ 0

fan_{in} → no. of incoming edges to a layer

fan_{out} → " " outgoing edges from ↓

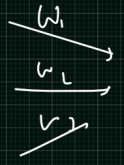


Condition to prevent vanishing & exploding gradient

$$fan_{in} = fan_{out}$$

alternative proposal

if $\sigma^2_{in} = \sigma^2_{out}$ ⇒ This can help you to maintain forward & backward info flow



↓

fanks the vanishing & exploding gradient issues

Calculate :-

$$fan_{avg} = \frac{fan_{in} + fan_{out}}{2}$$

For sigmoid,

Glorot initialization or Xavier initialization

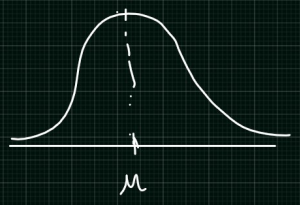
→ Normal distribution with $\mu = 0$ & variance $= \frac{1}{fan_{avg}}$

→ Uniform distribution between $-r$ & r where $r = \sqrt{\frac{3}{fan_{avg}}}$

↳ Default case of weight initialization in Keras

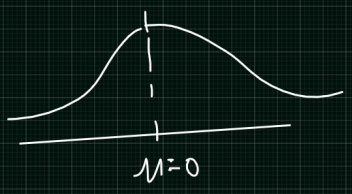
Normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Std Normal distribution $\mu = 0$, $\sigma = 1$

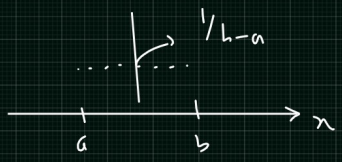
$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



Uniform distribution

a, b

$$f(x) = \frac{1}{b-a}$$



	Initialization	Regression	σ^2 (normal)
Xavier Glorot & Yoshua Bengio	Glorot-	None, tanh, sigmoid, softmax	$1/fan_{avg}$
Kaiming He	He	ReLU & its variants	$2/fan_{in}$
Yann Lecun	Le Cun	SELU	$1/fan_{in}$

activation fn

tf.keras.layers.Dense(unit, activation = "relu", kernel_initialization = "he_normal")

$W_1 \ldots W_{100}$ $\longrightarrow$ $\sigma^2 = 2/fan_{in}$