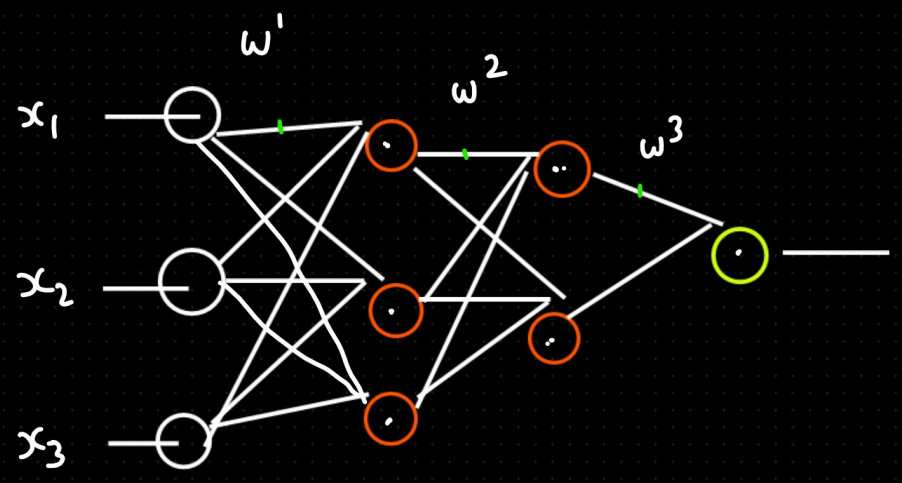
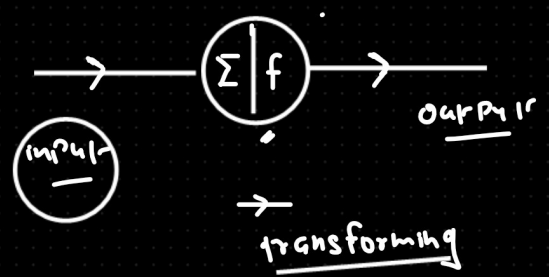
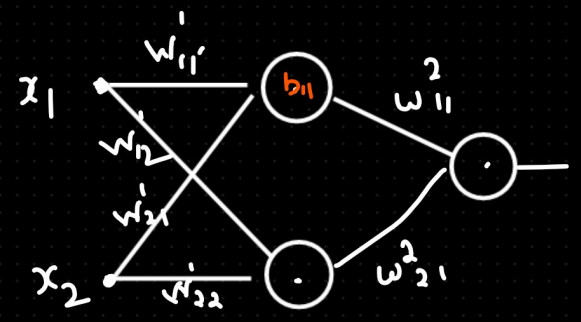


Act fn

→ To achive a non linearity in the data

→ To make sure that my neuron is activated or not activated



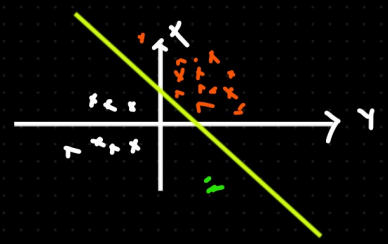


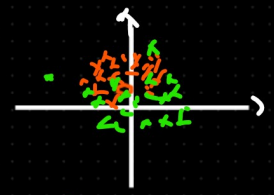
Forward Propagation

1. Weighted Sum

$$\text{Act}\left(x_1 w^1_{11} + x_2 w^1_{21} + b_{11}\right)$$

$$ax + by + c$$





= Linearseparable.

Linear Act fn

= Linear Fn

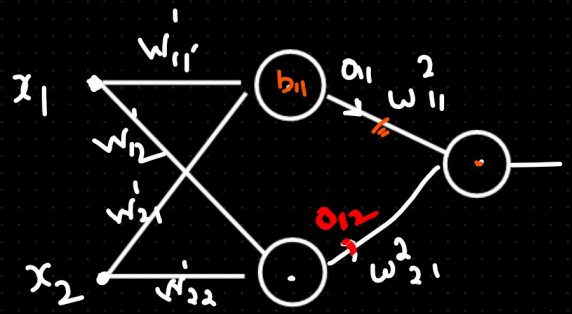
$\boxed{y = f(x)}$ $\quad x=1,\ x=2$

$y = f(1) = 1$

$y = f(2) = 2$

Non-linear Act fn

1. Sigmoid fn
2. tanh fn
3. Relu
   - Leaky relu
   - Parametric relu
   - ELU
   - SELU
4. Softmax act

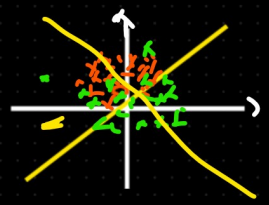


$$\text{Act}\left(x_1 w^1_{11} + x_2 w^1_{21} + b_{11}\right) = o_{11}$$

$$ax + by + c$$

$$\text{Act}\left(x_1 w^1_{12} + x_2 w^1_{22} + b_{12}\right) = o_{12}$$

$$O_{11} \times w_{11}^2 + O_{12} \times w_{21}^2 + b_{21} = O_{21}$$



$$Act(x_1 w_{11}^1 + x_2 w_{21}^1 + b_{11}) w_{11}^2 + Act(x_1 w_{12}^1 + x_2 w_{22}^1 + b_{12}) w_{21}^2 + b_{21} = \hat{Y}$$

Linear activation fn

$$\boxed{f(x) = x}$$

$$\underbrace{(x_1 w_{11}^1 + x_2 w_{11}^1 + b_{11})}_{A_0} w_{11}^2 + \underbrace{(x_1 w_{12}^1 + x_2 w_{12}^1 + b_{12})}_{A_1} w_{21}^2 + b_{21}$$

$f(x) = x$

is this a linear eq. ← $\boxed{A_0 w_{11}^2 + A_1 w_{21}^2 + b_{21}}$ = output

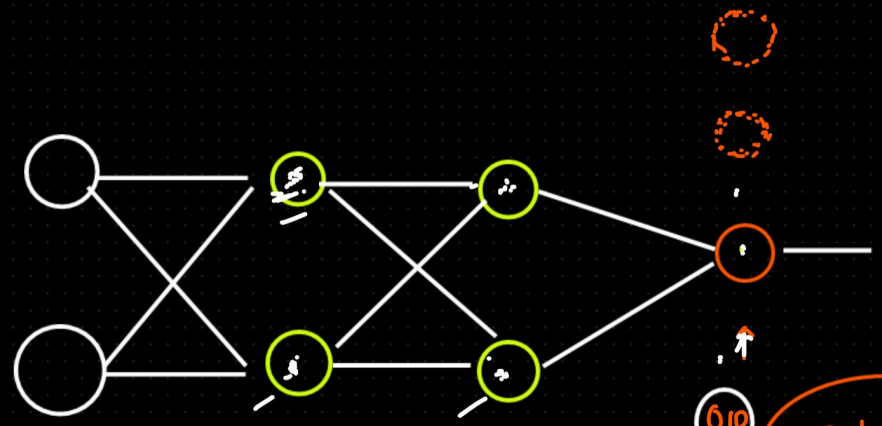
notation notation

Something I want that can provide non linearity w.r.t my data

NonLinear act fn

Regression

Binary classification



it Depends on req.

O/P

activation

Multiclass classification

vanishing gradient

~~Sigmoid fn~~

~~tanh fn~~

Dying relu

relu

Prelu

ELU

SELU

Leaky relu

linear n/c

sigmoid fn

tanh fn

relu

Softmax

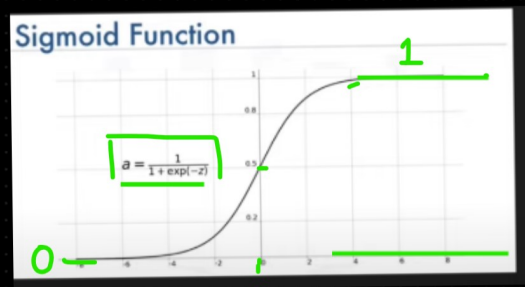
Act fn

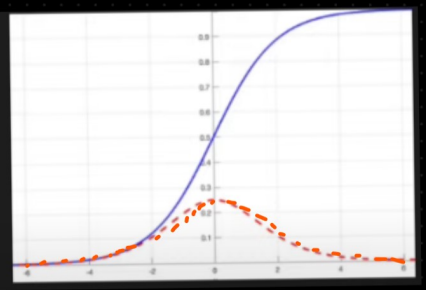
1. Non-linearity
2. Differentiable
3. Computationally inexpensive
4. Zero Centred × (0,1)
5. Non-Saturating → non-squeezed

Saturating → it is not a squeezed fn

1 Sigmoid fn

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





$= (0,1)$

Differentiate

Derivation

$= (0 - 0.25)$

$\boxed{\text{relu}(x_1 w_{11}^1 + x_2 w_{21}^1 + b)}$ = O/P

$\hat{y} = L$ Sigmoid

BP

$\frac{1}{1+e^{-x}}$

$O_{11} \times w_{11}^2$

$= \frac{1}{1+e^{-(x_1 w_{11}^1 + x_2 w_{21}^1 + b)}}$

vanishing gradient

= squeed fn

saturated

$(-\infty, \infty)$

$(0,1)$

= Squeezed $(0,1)$ $\frac{\partial L}{\partial w}$

vanishing gradient

~~Sigmoid~~

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Relu

Advantage

1. [0, 1] ⇒ Probability
2. Output layer → Binary classification
3. Non-linearity → Non linear Data ← good option
4. Differentiable function

Disadvantage

1. Saturated fn (squeezed fn)
2. Computationally expensive
3. non-zero centric
4. Vanishing gradient

Slow

Back Propogation

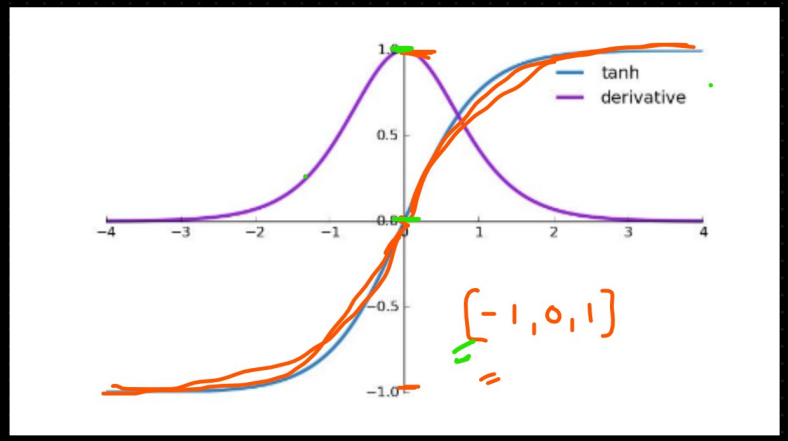
Tanh (hyperbolic tangent fn)

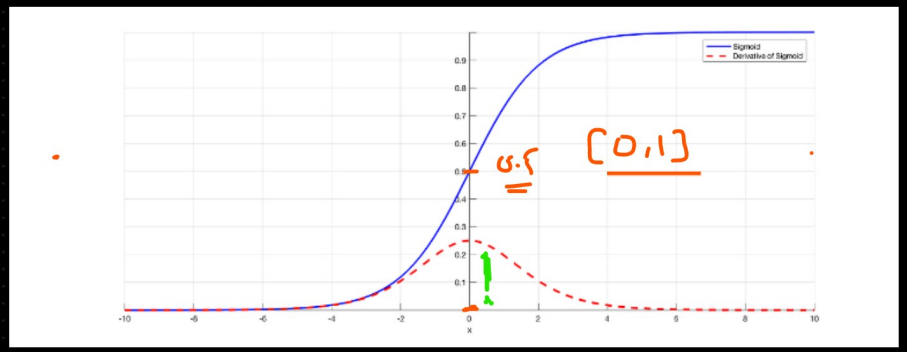
$(-\infty, +\infty) \Rightarrow (-1, +1)\ (0, 1)$

$(-\infty, +\infty) \Rightarrow (0, 1)$

$(0, 0.25)$

$= \frac{1}{1+e^{-x}}$





$$= \boxed{f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}} \leftarrow (0, 1)$$

$$= \boxed{f(x) = \frac{1}{1+e^{-x}}} \leftarrow (0, 0.25)$$

~~output~~

output (0,1)

$$\frac{dx}{dx} = \frac{d\left(\frac{1}{1+e^{-x}}\right)}{dy} = \quad =$$

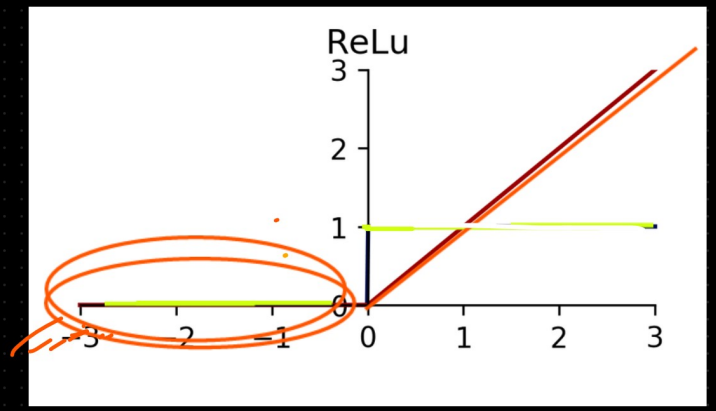
Advantage

1. Non-Linearity
2. Differentiable
3. Zero centric ⇒ training will be faster

Disadvantage

1. Saturated fn → squeezed fn ⇒ Vanishing gradient Problem
2. computational expensive
3. Don't use tanh is your output layer

3 Relu



$f(x) = \max(0, x)$

$f(-5) = 0$

$f(-2) = 0$

$f(4) = 4$

$f(2) = 2$

~~Squeezing~~

1

vanishing gradient problem

1. Non-linear
2. Non-saturated in +ve region
3. Computational inexpensive
4. (Convergence will be faster)

= (Batch normalization)

1 or 0

0

Sigmoid

tanh

relu

faster training

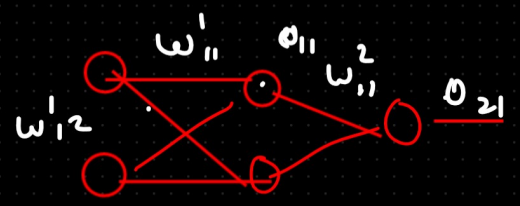
Disadvantage

Differentiation

$x \le 0 \rightarrow 0$

$x > 0 \rightarrow 1$

$\Rightarrow$ dying Neuron



$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W}$$

$W_{new} = W_{old}$

no updation

$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w^2_{11}} \times \frac{\partial w^2_{11}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w^1_{11}}$$

$L \rightarrow \hat{y} \rightarrow w^2_{11} \rightarrow o_{11} \rightarrow w^1_{11}$

negative

$\Rightarrow 0$

relu

dying Neuron

Problem

1. Vanishing gradient
2. exploding gradient
3. Batch normalization
4. Regularization
5. Dying neuron

⇒ how to increase the performance of NN

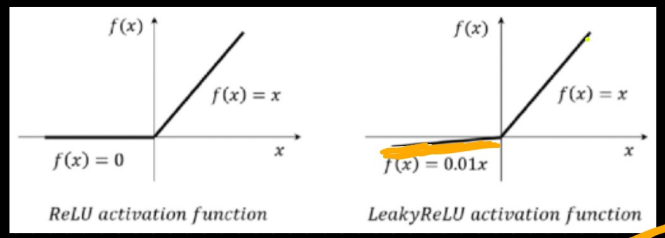
Leaky Relu

(LRELU)

Parameter = 0.01

Parametric relu

(PRELU)



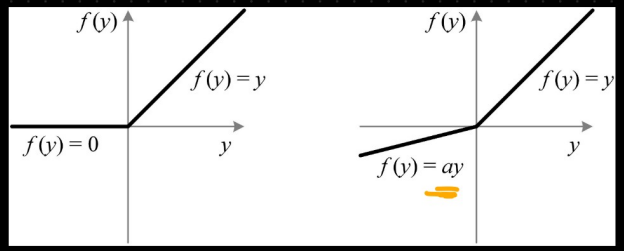
0.01
0.02
0.03

$f(x) = \max(0.01 \cdot x, x)$

$f(-1) = -0.01$

$f(2) = 2$

$f(-3) = -0.03$



ay, y

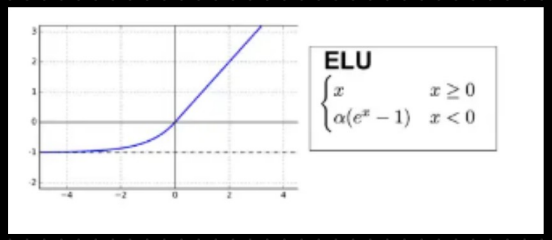
$\max(ax, x)$

learning Parameter

0.01

(ELU)

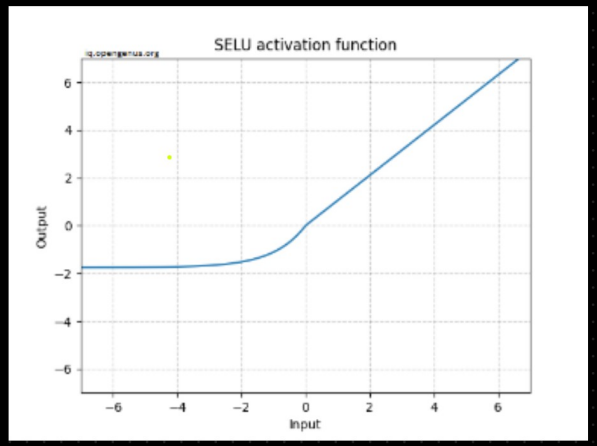
Exponential linear unit



$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

(SELU)

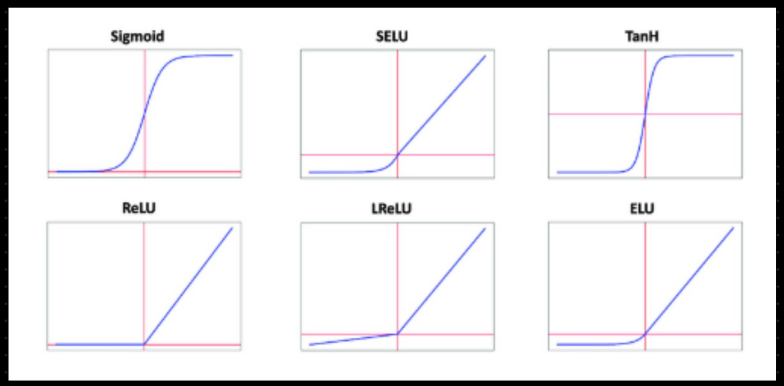
Scale & exponential linear unit



very famous, very useful

- Relu $\Rightarrow$ $\max(0, x)$
- Leaky Relu $\Rightarrow$ $\max(0.01x, x)$
- Parametric Relu $\Rightarrow$ $\max(\alpha x, x)$

$$ELU = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



$$SELU = \lambda \begin{cases} x & x \geq 0 \\ \alpha e^x - \alpha & x < 0 \end{cases}$$

recent research

$\lambda \approx 1.0507009873554804934193349852946$

$\lambda \approx 1.0507009873554804934193349852946$

λ = Scaling factor

α =

NAME	Function range	derivation range
Linear fn	$f(x) = x$	1
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$ $(0,1)$	$(0, 0.25)$
tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ $(-1,+1)$	$(0,1)$
Relu	$\max(0, x)$ $(0, x)$	0 or 1 $x \le 0$ 0 $x > 0$ 1
Leaky Relu	$\max(0.01x, x)$ $(0.01x, x)$	0.01 or 1
PRelu (Parametric)	$\max(\alpha x, x)$ $(\alpha x, x)$	α or 1

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \qquad \begin{cases} 1 & x > 0 \\ (0,1) & x \leq 0 \end{cases}$$

SELU

$$\lambda \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \qquad \begin{cases} 1 & x > 0 \\ (0,1) & x \leq 0 \end{cases}$$

$\lambda \approx 1.0507009873554804934193349852946$

Practical on Softmax

Act

ANN

Weight updation

Optimizer

Problems

Practical