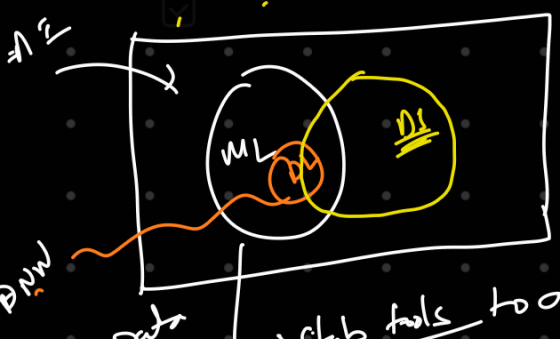
AI Vs ML Vs DL Vs DS



→ Stab tools to analyse
Preprocess, prediction, forecasting

AI application

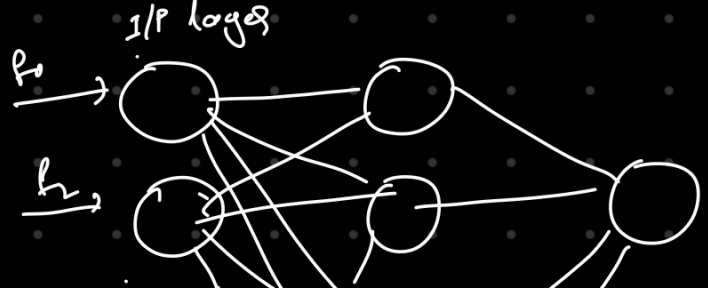
1. Medical
2. Self Driving
3. Amazon
4. youtube

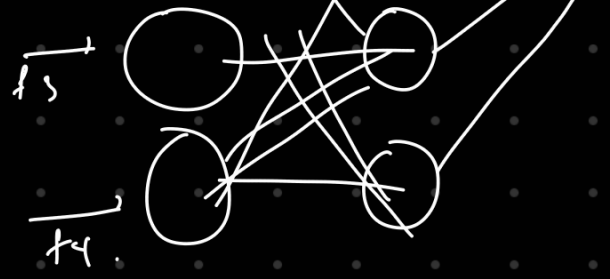
1. Deep learning
   - i) ANN ← Artificial Neural NW ← Tabular Kind of dataset.
2. CNN ← Convolutional neural network ← Images, Videos.
   - a) Image classification ← CNN, Transfer Learning
   - b) Object detection ← RCNN, FAST RCNN, FASTER RCNN, SSD, Yolo, Detectron
   - c) Segmentation
   - d) Tracking
   - e) GAN
3. RNN ← Recurrent Neural NW ← I/P → Text, Time series, sequential Data
   - → RNN
   - → LSTM RNN
   - → Bidirectional LSTM RNN
   - → Encoders Decoders
   - → Transformers
   - → BERT
   - → GPT., GPT2, GPT3.

NLP → Text → Vectors

Bow
TFIDF
word2Vec
embedding.

→ first NW → Perceptron → not effective
→ Jeffrey Hinton → proposed back tracking → become effective.

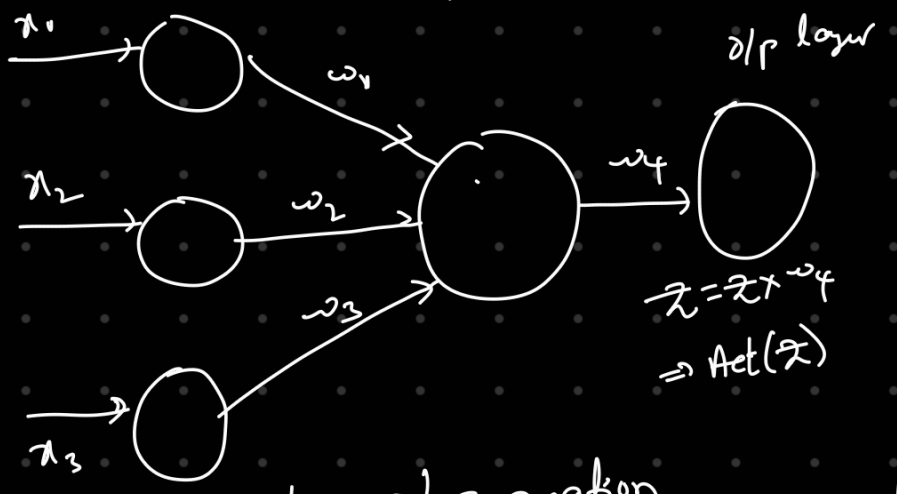




How nlw works?

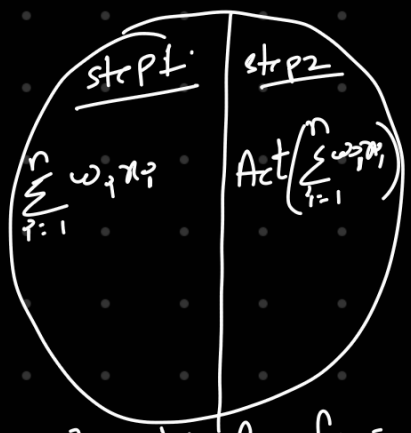
$y = w_1x_1 + w_2x_2 + w_3x_3 + bias$

$z = Act(y)$



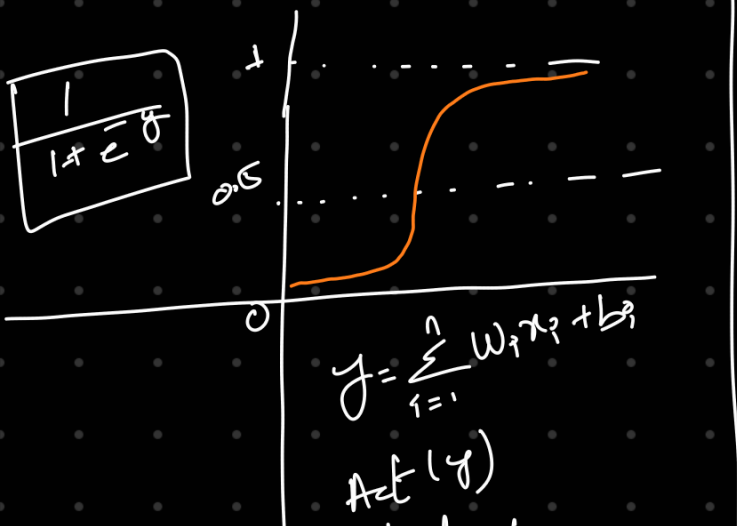
forward propagation.

Neuron



Sigmoid Activation fn = $\frac{1}{1+e^{-z}}$

Activation functions

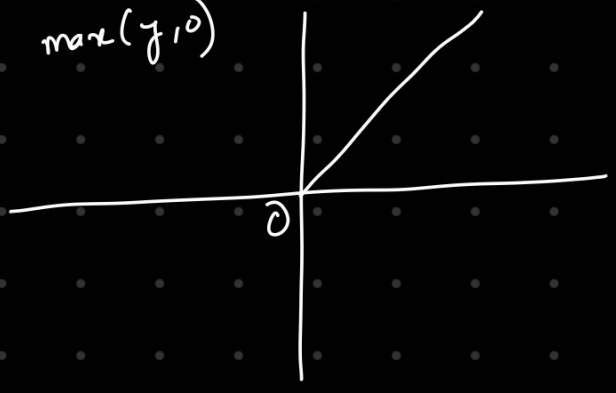


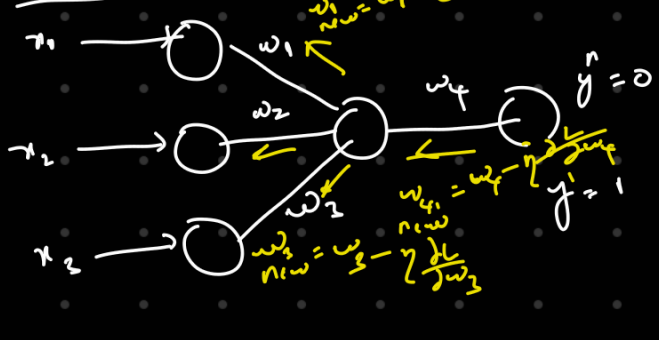
Sigmoid Activation fn

→ output 1 means the neuron is activated.

② ReLU Af

$\max(y, 0)$





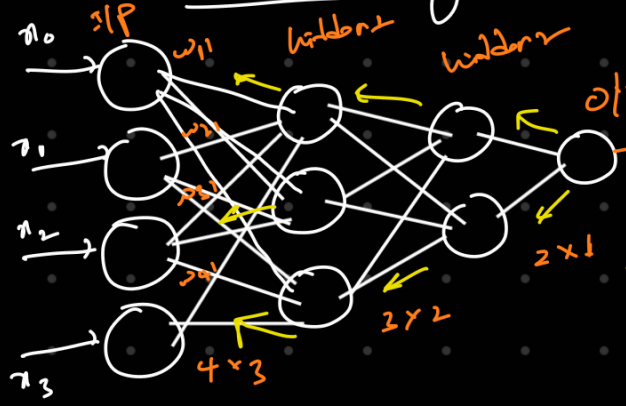
$Loss = (y - \hat{y})^2$ ↓ need to decrease this.

$= (1-0)^2$

$cost\ fn = \sum_{i=1}^{n} (y - \hat{y})^2$

minimize loss → Optimizer.

ANN Training

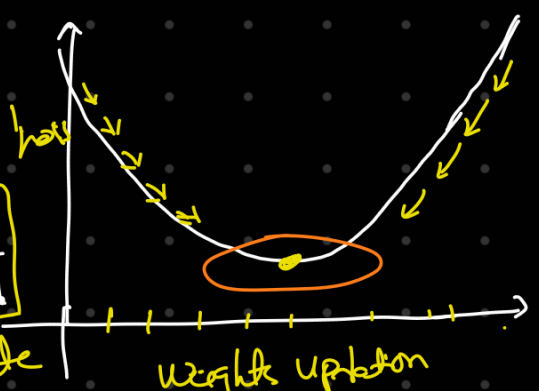


Loss $= (y-\hat{y})^2$ ↓↓

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

learning rate (0.001) small val.

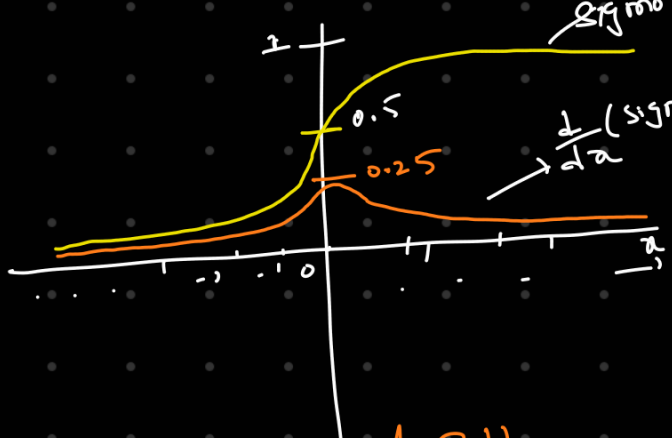
Gradient Descent



chain rule in back propagation

$$w^3_{11\,new} = w^3_{11\,old} - \eta \frac{\partial L}{\partial w^3_{11\,old}} \rightarrow \frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial O_{31}} * \frac{\partial O_{31}}{\partial O_{21}} + \frac{\partial O_2}{\partial w_{11}} + \frac{\partial L}{\partial O_3} \frac{\partial O_{31}}{\partial O_{22}} \frac{\partial O_{22}}{\partial w_{11}}$$

Vanishing Gradient Problem



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

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial O_{21}}{\partial O_{11}} * \frac{\partial O_{11}}{\partial w_{11}} \rightarrow \text{chain rule}$$

4 layers $0.20 \times 0.1 \times 0.05 \rightarrow 0.001$

$= 0.25 - 1 \times 0.04$

$= 2.46 \times \cdot$ $\simeq 2.99$

→ So, $w_{11\,new} \simeq w_{11\,old}$ weights not getting updated.

Exploding Gradient Problem

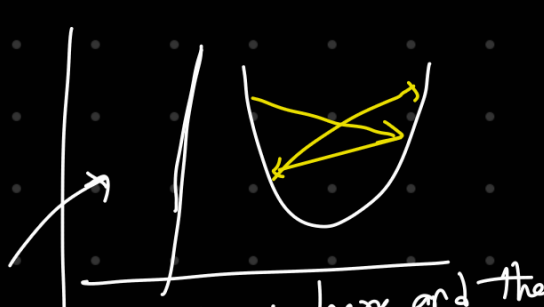
$$\frac{\partial O_{21}}{\partial O_{11}} = \frac{\partial \phi(z)}{\partial z} * \frac{\partial z}{\partial O_{11}} \rightarrow \frac{\partial (O_{11} \cdot w_{21} + b_2)}{\partial O_{11}} = w_{21}$$

z

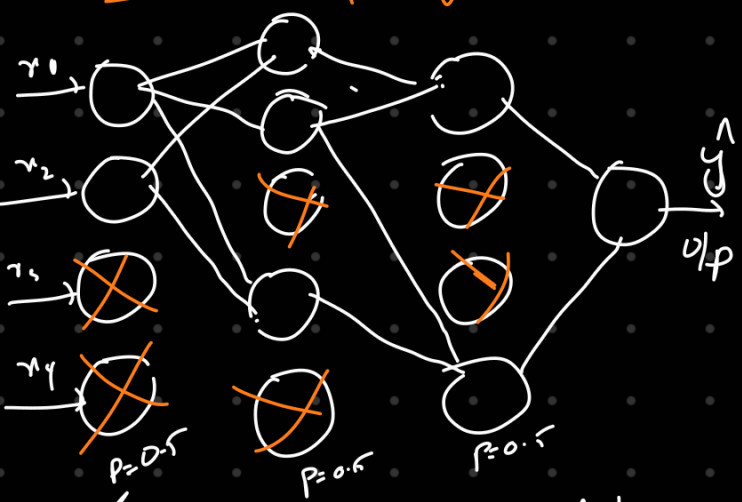
$\Rightarrow [0 \leq \downarrow \leq 0.25] * w_{21}$

$\Rightarrow 0.25 \times 500 = 125$

If wts are initialized with higher values, then only exploding gradient problm happens.



Dropout & Regularization



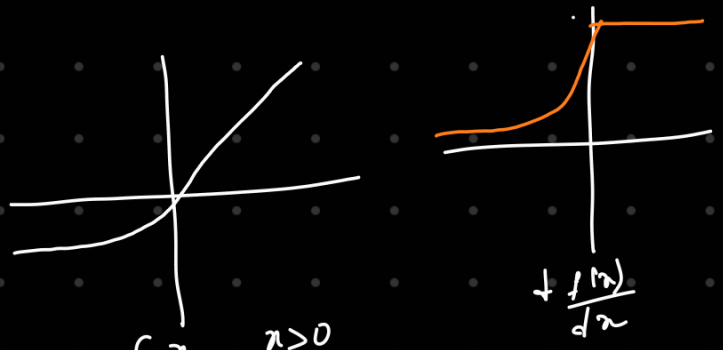
↓
50% of neurons are deactivated.

Dropout rate

$0 \leq p \leq 1$

→ to reduce overfitting.

ELU {Exp Linear Units}



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

↓
→ disadvantage is computationally exp.

PRelu (Parameterised ReLU)

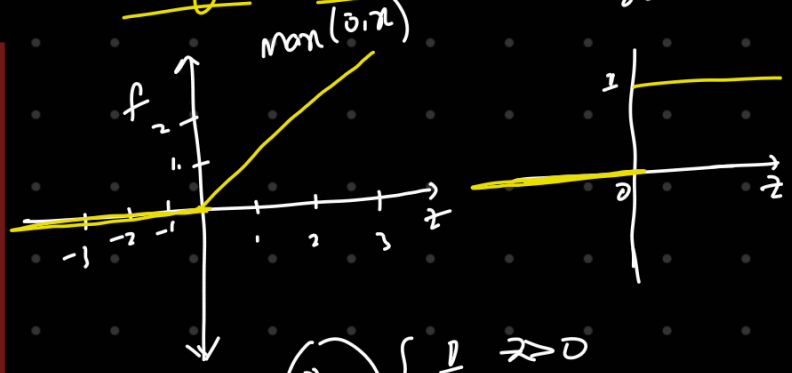
$$f(x) = \begin{cases} x & ; x > 0 \\ \alpha x & ; x <= 0 \end{cases}$$

↓
learning parameter.

if $\alpha = 0.01$ → leaky relu

$\alpha = 0$ → Relu.

Rectified Linear Unit $\frac{\partial f}{\partial x} = \begin{cases} 0 \\ 1 \end{cases}$

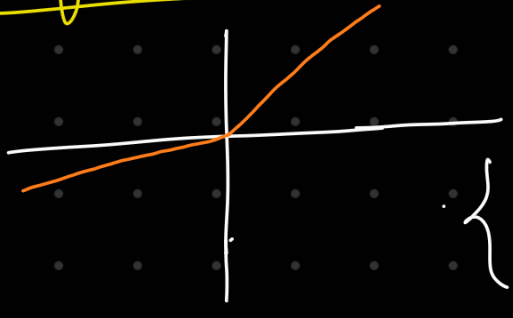


$$W_{new} = W_{old} - \eta \left(\frac{\partial L}{\partial W_{old}} \right) \begin{cases} 1 & x > 0 \\ 0 & x <= 0 \end{cases}$$

↓
$0 \times 1 \times 1$

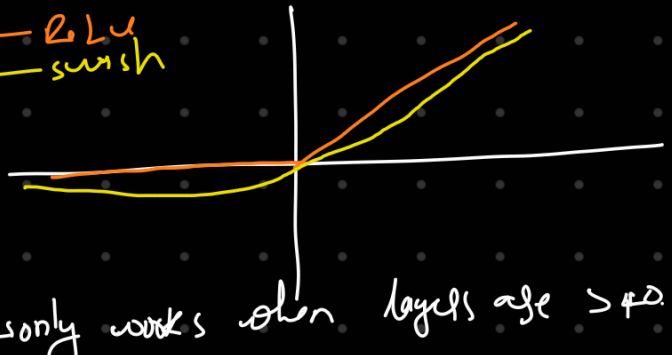
↓
if $x <= 0$ → 0 creates dead neuron. during back propagation.

→ Leaky Relu



$$\begin{cases} x > 0 \\ 0.01(x) & x <= 0 \end{cases}$$

Swish (A self-Gated) Function



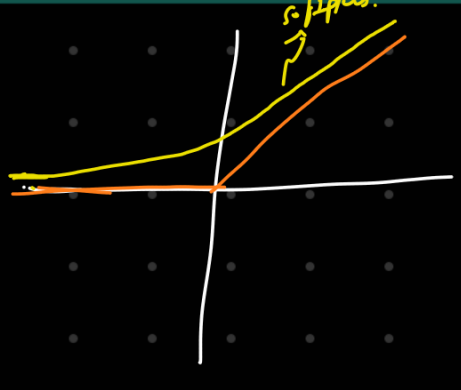
→ only works when layers are > 40.

→ $y = x * \text{sigmoid}(x)$

→ Computationally exp.

Softplus

$f(x) = \ln(1 + e^x)$



Weight Initialization

1. weights should be small
2. weights should not be same
3. weights should have good variance

① Uniform Distribution

$$W_{ij} \sim \text{Uniform}\left[\frac{-1}{\sqrt{fan\_in}}, \frac{1}{\sqrt{fan\_out}}\right]$$

(a, b)

② Xavier / Glorot

→ Xavier Normal

$$W_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{fan\_in + fan\_out}}$$

→ Xavier Uniform

$$W_{ij} \sim U\left[\frac{-\sqrt{6}}{\sqrt{fan\_in + fan\_out}}, \frac{\sqrt{6}}{\sqrt{fan\_in + fan\_out}}\right]$$

③ He initialization

(a) He Normal

$$W_{ij} \approx N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{fan\_in}}$$

(b) He Uniform

$$W_{ij} \approx U\left[\frac{-\sqrt{6}}{\sqrt{fan\_in}}, \frac{\sqrt{6}}{\sqrt{fan\_out}}\right]$$

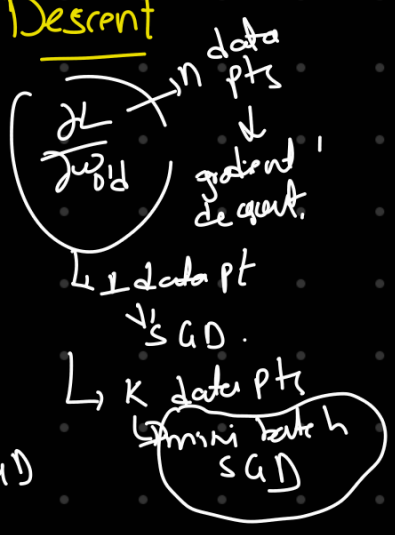
Stochastic Gradient Descent

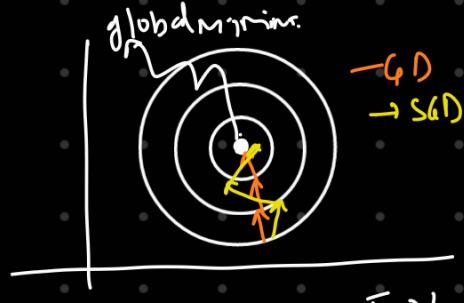
$$W_{new} = W_{old} - \eta \times \frac{\partial L}{\partial W_{old}}$$

$$Loss = \sum_{i=1}^{n} (y - \hat{y})^2$$

$\sum_{i=1}^{n} (y - \hat{y})^2$ → GD

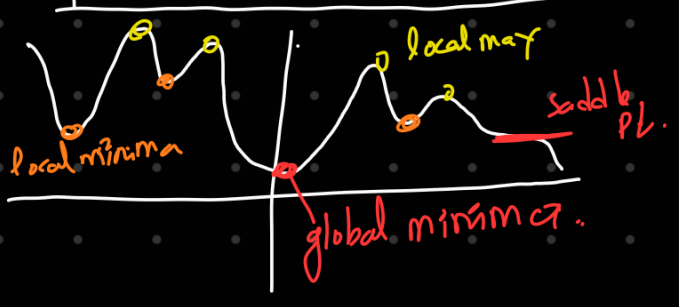
$(y - \hat{y})^2$ → SGD



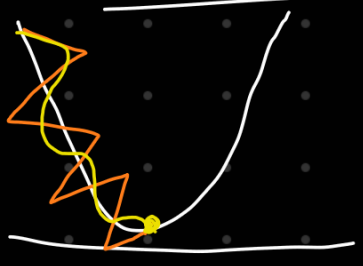


$$\left[\frac{\partial L}{\partial W_{old}}\right]_{\text{mini batch sample}} \approx \left[\frac{\partial L}{\partial W_{old}}\right]_{GD \rightarrow \text{population}}$$

Global Minima V/s Local Minima



SGD with momentum



t: t_1 t_2 t_3 ... t_n

b: b_1 b_2 b_3 ... b_n

$t = 1$ $V_{t_1} = b_1$

$V_2 = \gamma * V_1 + b_2$

$V_3 = \gamma V_2 + b_3$

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

$$= W_{old} - \left[\gamma V_{t-1} + \eta \frac{\partial L}{\partial W_{old}}\right] \quad \gamma = 0.9$$

$$V_{t-1} = 1 * \left[\frac{\partial L}{\partial W_{old}}\right] + \gamma \left[\frac{\partial L}{\partial W_{old}}\right]_{t-1} + \gamma^2 \left[\frac{\partial L}{\partial W_{old}}\right]_{t-2} \cdots$$

Adagrad Optimizer

GD, SGD, Mini SGD

$$W_t = W_{t-1} - \eta \frac{\partial L}{\partial W_{t-1}}$$

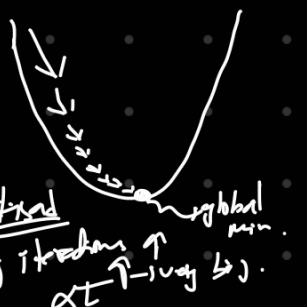
$$W_t = W_{t-1} - \left(\frac{\eta}{t}\right) \times \frac{\partial L}{\partial W_{t-1}}$$

Decreasing slowly after time.

$$\eta_t = \eta / \sqrt{\alpha_t + \varepsilon} \quad (\varepsilon: \text{small +ve val})$$

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial W_i}\right)^2$$

↳ higher no.



Adadelta & RmsProp

- Need to restrict η_t to go small beyond certain value.

Adagrad $w_t = w_{t-1} - \eta'_t \frac{\partial L}{\partial w_{old}}$

$\eta_t = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$ $\quad w_{avg_t} = 0.9 \times w_{avg_{(t-1)}} + (1-0.9)\left(\frac{\partial L}{\partial w}\right)^2_t$

$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial w_t}\right)^2$

Adam Optimizer (Adaptive Moment estimation)

- momentum → smoothing
- RMS Prop → α_t

Advantages

Algo

$V_{dw}=0 \quad V_{db}=0 \quad S_{dw}=0 \quad S_{db}=0$

on iteration t

① Compute $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}$

$V_{dw} = \beta_1 V_{dw} + (1-\beta)\frac{\partial L}{\partial w}$ } momentum

$V_{db} = \beta_1 V_{dw} + (1-\beta)\frac{\partial L}{\partial b}$

$S_{dw} = \beta_2 S_{dw} + (1-\beta)\left(\frac{\partial L}{\partial w}\right)^2$

$S_{db} = \beta_2 S_{db} + (1-\beta)\left(\frac{\partial L}{\partial b}\right)^2$ } RMS Prop

$w_t = w_{t-1} - \frac{\eta * V_{dw}}{\sqrt{S_{dw}+\epsilon}}$

$b_t = b_{t-1} - \frac{\eta * V_{db}}{\sqrt{S_{db}+\epsilon}}$

↳ Adam opt

Loss functions

MSE

$= \frac{1}{m}\sum (y-\hat{y})^2$

i) plot the quadratic eqn, we get a gradient descent with only global minima

ii) We don't get any local minima

2) The MSE loss penalizes the model for making large errors by squaring them.

→ Disadvantages

① It is not robust to outliers.

② Absolute Error Loss

① $L = |y-\hat{y}| \quad J = \frac{1}{t}\sum_{i=1}^{t} |y-\hat{y}|$ → computational

→ the MAE is more robust to outliers as compared to MSE

③ Huber Loss

① MSE ② MAE

$$\text{Loss} = \begin{cases} \frac{1}{2}(y-\hat{y})^2, & \text{if } |y-\hat{y}| \le \delta \leftarrow \text{hyperparameter} \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$$

↳ binary cross entropy

Cross Entropy → Logistic Regression ⇒ Multi class cross Entropy Loss

$$\text{Loss} = -y \times \log(\hat{y}) - (1-y) * \log(1-\hat{y})$$

Binary cross entropy, bin classification

$$\begin{cases} -\log(1-\hat{y}) & ; y=0 \\ -\log(\hat{y}) & ; y=1 \end{cases}$$

$\hat{y} = \text{Sigmoid} = \frac{1}{1+e^{-z}}$

$L(x_i, y_i) = -\sum_{j=1}^{c} y_{ij} * \log(\hat{y}_{ij})$ → categorical cross entropy.

o/p → Multiclass → good, Bad, Neutral → one hot encoded

f_1	f_2	f_3	o/p
-2	3	4	good → [1 0 0]
4	5	6	Bad → [0 1 0]
7	8	9	Neutral → [0 0 1]

y_{i1} y_{i2} y_{i3} → one hot encoded.

[1 0 0] → $\hat{y}_{ij}$ ⇒ softmax act fn.

[0 1 0]

[0 0 1] $\quad \sigma(z) = \frac{e^{z_j}}{\sum_{j=1}^{c} e^{z_j}}$

#CNN

① ANN → Regression
→ classification

② CNN

CEREBAL CORTEX
↓
visual cortex
eyes
↓
V_1 → filters
V_2, V_3, V_4, V_5, V_6
image

What is Convolution ?

0 to 255

Grey scale

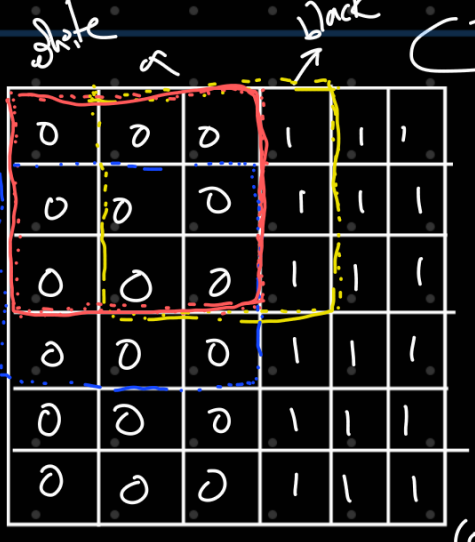
4×4

Colored (RGB)

- B channel
- G channel
- R channel

pixels 0-255

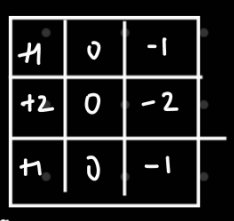
6×6×3



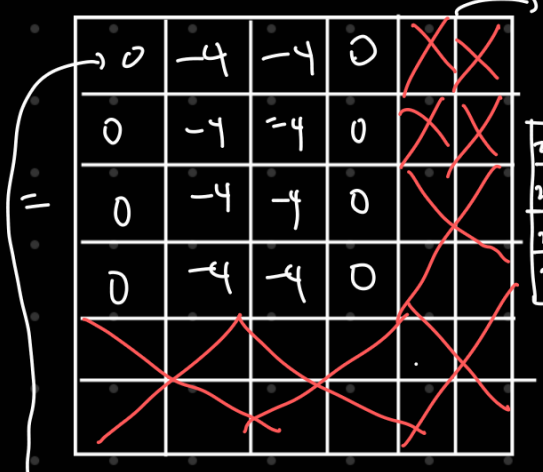
$n=6$, $f=3$ → 4×4

$n-f+1$ → changing for padding

Here stride = 1



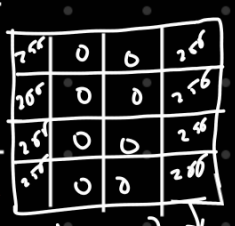
$(0\times1 + 0\times0 + 0\times-1) +$
$(0\times2 + 0\times0 + 0\times-2) +$
$(0\times1 + 0\times0 + 0\times-1) = 0$



→ then one step right

→ like one step down with filter.

Applying min max scale



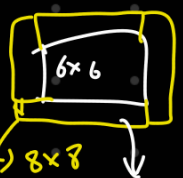
#Padding in CNN

→ We are loosing some info when filter is added.

$6\times6 \xrightarrow[3\times3]{} 4\times4$

$n-f+1 = 6$ → for not loosing anything

$n = 8$ so add extra edges



now $8\times8 \xrightarrow[3\times3]{} 6\times6$

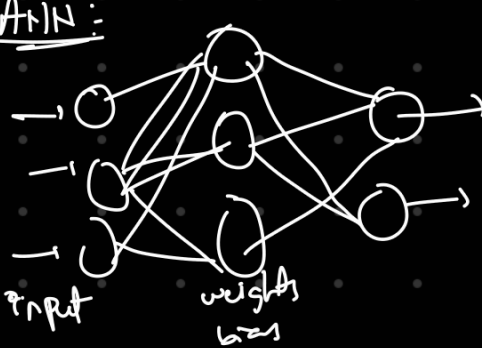
Values : ① '0's of padding
② real values

⇒ new formula :

$$n-f+1+2p$$

Operation Of CNN :

ANN :



↓ Similarly