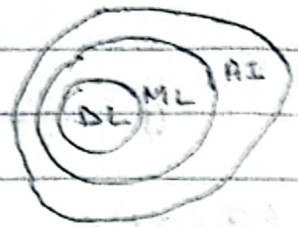
Foundation for Advancement of Science & Technology

Lecture No. 1

Deep Learning → automatically learn features (or extract)

Machine Learning → have to tell the features.

AI → learns itself and then give the output, we define rules.



•) Don't give instructions in ML

Feature Learning → automatically extracts new features.

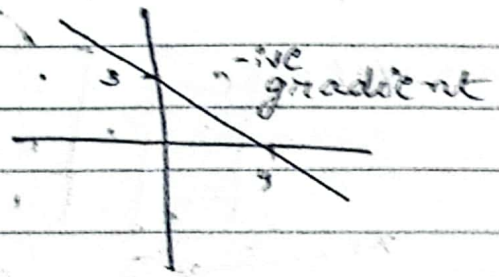
Features

- Shallow
- Hierarchical (sub-features) → combo of small features to big features
- Deep (automatically learned)

Lecture No. 2

Linear Regression → tells the linear relationship b/w dependent & independent variable.

i) slope $= \frac{\text{rise}}{\text{run}} = \frac{-3}{4}$



$b = 3$

$$y = \frac{-3}{4}x + 3$$

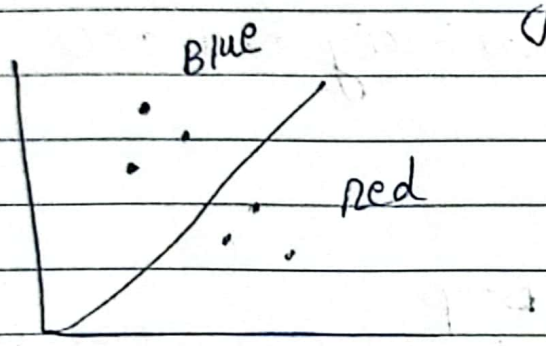
+ive gradient -ive gradient

ii) $y = \frac{5}{2}x - 2$, $c = 2$, slope $= \frac{\text{rise}}{\text{run}} = \frac{5}{2}$

iii) $y = \frac{2}{3}x + 2$, $c = 2$, $m = \frac{\text{rise}}{\text{run}} = \frac{2}{3}$

- Equation of Line : $ax + by + c = 0$
  If $c = 0$ then $ax + by = 0$ (line passes through origin)

Ex:



case 1: (6,6)
" 2: (5,0)
" 3: (3,5)

$y = x$
$val = y - x = 0$
$6 - 6 = 0$ (case 1)
$0 - 5 = -5$ (" 2)
$5 - 3 = 2$ (" 3)

if (val ≥ 0)
blue
else
red.

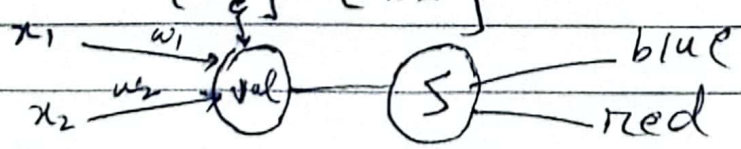
- Perceptrons are linearly seperable

bias term

$$val = \begin{bmatrix} c \\ w_1 \\ w_2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$val = \vec{w} \cdot \vec{x} = 0$

$y = w_1 x_1 + w_2 x_2 + c \cdot 1$



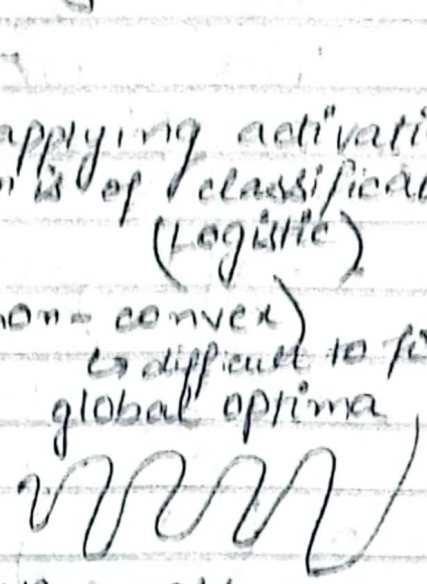
Lecture - 3

$y = mx + c$

↳ after applying activation, the problem is of classification (Logistic)

$\hat{y}$ = cost function = $\frac{1}{1+e^{-y}}$ (non-convex) ↳ difficult to find global optima

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



To make it convex, we will use cross-entropy

→ cross-entropy = $-y \log \hat{y} - (1-y)\log(1-\hat{y})$.

→ <u>Gradient Descent</u>

for each sample
compute cost
compute loss
update m and c

$$m_{new} = m_{old} - \alpha \frac{\partial L}{\partial m}$$

$$c_{new} = c_{old} - \alpha \frac{\partial L}{\partial c}$$

$$\frac{\partial L}{\partial m} = -y \log \hat{y} - (1-y)\log(1-\hat{y}) \qquad \because \hat{y} = a = 1/1+e^{-z}, \quad z = mx+c$$

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

$$\frac{\partial}{\partial m}\left(-y \log\left(\frac{1}{1+e^{-z}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-z}}\right)\right) \text{ or}$$

Foundation for Advancement of Science & Technology

$$\frac{\partial L}{\partial m} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial m}$$

$$\frac{\partial L}{\partial a} = \frac{\partial}{\partial a}\left(-y \log a - (1-y)\log(1-a)\right)$$

$$= -y\left(\frac{1}{a}\right)\cdot(1) - (1-y)\cdot\frac{1}{1-a}(-1)$$

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

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

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

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

$$\frac{\partial z}{\partial m} = \frac{\partial}{\partial m}(mx+c) = x$$

putting values

$$\frac{\partial L}{\partial m} = \left(\frac{-y}{a} + \frac{1-y}{1-a}\right) \cdot a(1-a) \cdot x$$

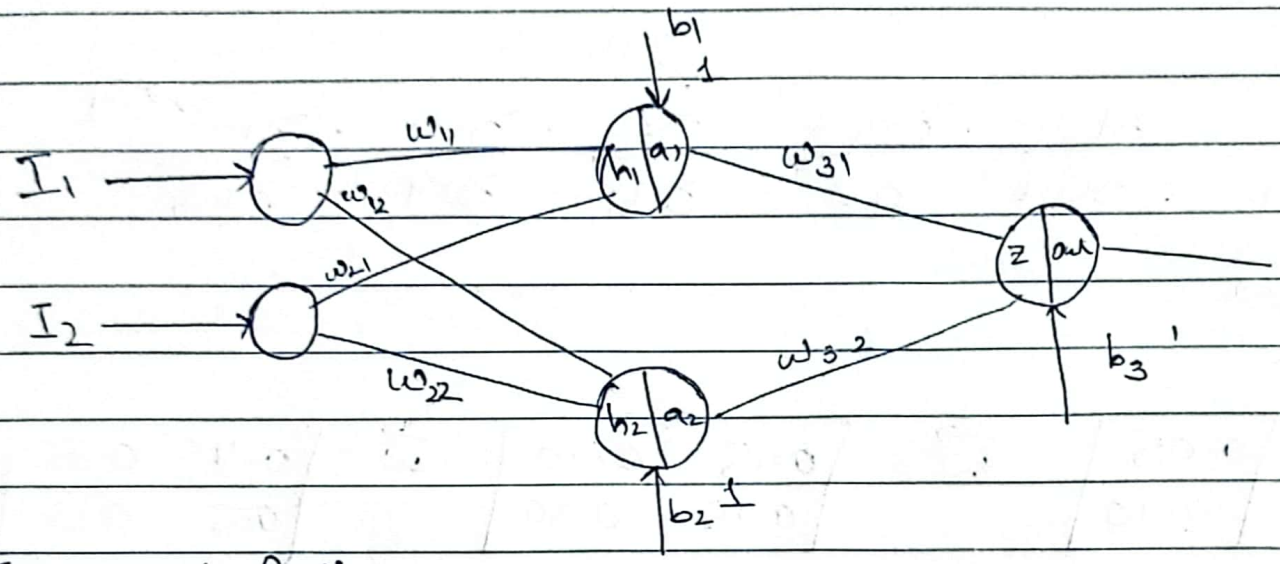
$$\frac{\partial L}{\partial c} = \left(\frac{-y}{a} + \frac{1-y}{1-a}\right) \cdot a(1-a) \cdot 1$$

CamScanner

Foundation for Advancement of Science & Technology

EX: NN

$$I = \begin{bmatrix} 0.35 \\ 0.9 \end{bmatrix} \quad \overset{1}{\vec{w}} = \begin{array}{cc} w_{11} & w_{12} \\ \end{array} \begin{bmatrix} 0.1 & 0.4 \\ 0.8 & 0.6 \end{bmatrix} \begin{array}{c} \\ w_{21} \quad w_{22} \end{array} \quad \overset{2}{\vec{w}} = \begin{bmatrix} 0.3 \\ 0.9 \end{bmatrix}$$



→ Forward Pass:

$h_1 = I_1 w_{11} + I_2 w_{21} + b_1$; $h_2 = I_1 w_{12} + I_2 w_{22} + b_2$

$a_1 = \sigma(h_1)$ $\quad a_2 = \sigma(h_2)$

$z = a_1 w_{31} + a_2 w_{32} + b_3$

$\text{Output} = \sigma(z)$

→ Backward Propogation:

$$(\text{loss})\ L = -y \underbrace{\log \hat{y}}_{out} - (1-y) \underbrace{\log(1-\hat{y})}_{out} \quad (\text{cross-entropy})$$

$$\frac{\partial L}{\partial w_{31}} = \frac{\partial L}{\partial out} = \frac{\partial out}{\partial z} \cdot \frac{\partial z}{\partial w_{31}}$$

$$\frac{\partial L}{\partial out} = -y \underbrace{\left(\frac{1}{out}\right)} - (1-y) \cdot \underbrace{\frac{1}{1-out}} \cdot (-1)$$

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

$$\frac{\partial z}{\partial w_{11}} = a_1$$

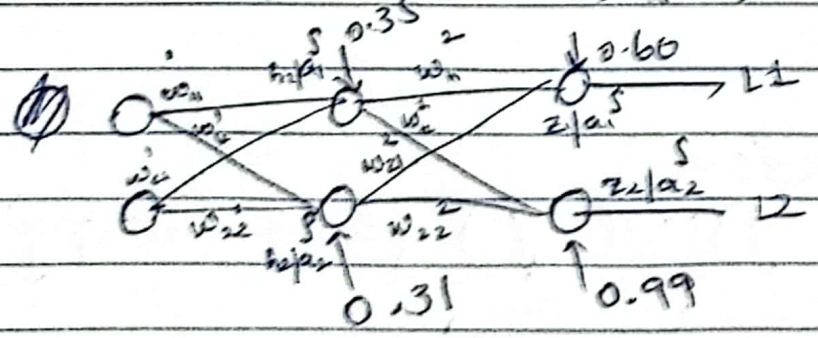
$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial out} \cdot \frac{\partial out}{\partial z} \cdot \overset{w_{21}}{\frac{\partial z}{\partial a_1}} \cdot \frac{\partial a_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_{11}} \rightarrow I_1$$

Assignment

$$I = \begin{bmatrix} 0.05 \\ 0.10 \end{bmatrix} \quad \overset{1}{\vec{w}} = \begin{bmatrix} 0.15 & 0.20 \\ 0.25 & 0.30 \end{bmatrix} \quad \overset{2}{\vec{w}} = \begin{bmatrix} 0.4 & 0.45 \\ 0.5 & 0.55 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} 0.35 \\ 0.35 \end{bmatrix} \quad b_2 = \begin{bmatrix} 0.60 \\ 0.99 \end{bmatrix} \quad y = \begin{bmatrix} 0.01 \\ 0.99 \end{bmatrix}$$

$w_{11}^2 = ?$ $\quad$ $w_{11}^1 = ?$



$\hat{y} = a = \frac{1}{1+e^{-z}}$ $\because z = mx + c$.

Foundation for Advancement of Science & Technology

Lecture - 4

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

Forward Pass:

$h_1 = I_1 w_{11}^1 + I_2 w_{21}^1 + b_1^1$

$a_1 = \sigma(h_1)$

$h_2 = I_1 w_{12}^1 + I_2 w_{22}^1 + b_2^1$ $\qquad w_{11}^2 = ?$

$a_2 = \sigma(h_2)$ $\qquad w_{12}^1 = ?$

$z_1 = a_1 w_{11}^2 + a_2 w_{21}^2 + b_2^1$

$O_1 = \sigma(z_1)$

$z_2 = a_1 w_{12}^2 + a_2 w_{22}^2 + b_2^2$

$O_2 = \sigma(z_2)$

Backward Pass/Propogation:

$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L_1}{\partial w_{11}^2} = \frac{\partial L_1}{\partial O_1} \times \frac{\partial O_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_{11}}$$

$$\frac{\partial L_1}{\partial w_{11}^2} = \frac{-y}{O_1} + \frac{1-y}{1-O_1} \cdot O_1(1-O_1) \cdot a_1$$

$$\frac{\partial L}{\partial w_{12}^1} = \frac{\partial L_1}{\partial w_{12}^1} + \frac{\partial L_2}{\partial w_{12}^1}$$

$$\frac{\partial L_1}{\partial w_{12}^1} = \frac{\partial L_1}{\partial O_1} \times \frac{\partial O_1}{\partial z_1} \times \frac{\partial z_1}{\partial a_2} \times \cancel{\frac{\partial w_{21}^2}{\partial a_2}} \times \frac{\partial a_2}{\partial h_2} \times \frac{\partial h_2}{\partial w_{12}^1}$$

$$\cancel{\frac{\partial L_1}{\partial w_{12}^1}} = \cancel{\frac{-y}{O_1} + \frac{1-y}{1-O_1}} \cdot \cancel{O_1(1-O_1)} \cdot \cancel{a_2} \cdot \cancel{a_2(1}$$

$$\frac{\partial L_1}{\partial w_{12}^1} = \frac{\partial L_1}{\partial O_1} \cdot \frac{\partial O_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial a_2} \cdot \frac{\partial a_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_{12}^1}$$

$$= \frac{-y}{O_1} + \frac{1-y}{1-O_1} \cdot O_1(1-O_1) \cdot w_{21}^2 \cdot a_2(1-a_2) \cdot I_1$$

$$\frac{\partial L_2}{\partial w_{12}^1} = \frac{\partial L_2}{O_2} \cdot \frac{\partial O_2}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_{12}^1}$$

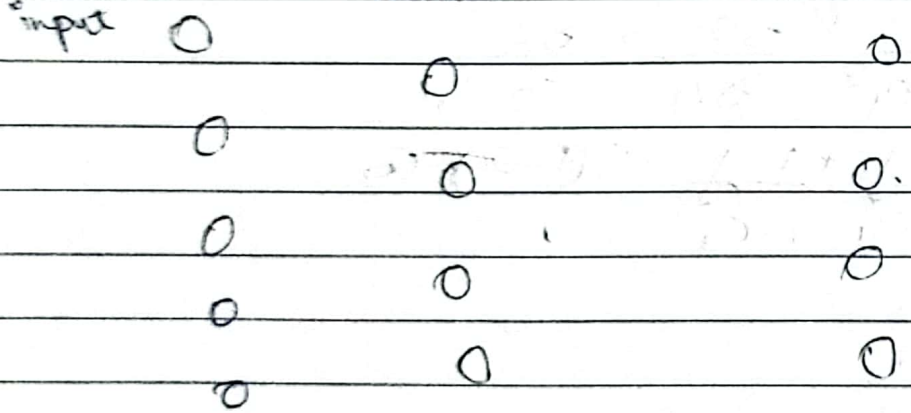
$$= \frac{-y}{O_2} + \frac{1-y}{1-O_2} \cdot O_2(1-O_2) \cdot w_{22}^2 \cdot a_2(1-a_2) \cdot I_1$$

DropOut Layer

of iterations
epochs

# of hidden layer
# of neurons per hidden layer.

ANN



Dimensions of Data = No. of neurons in the input layer.

To improve acc and avoid overfitting, we use early stopping.

Foundation for Advancement of Science & Technology

Lecture-5

Dropout Layer

- complexity reduce
- different types of neural n/w train through Ensemble Learning
  ↳ Random Forest → Decision Trees.
- no. of combinations = 2^ no. of neurons except output neuron
- no overfitting, solved vanishing gradient problem.
- During testing, multiply weight of every neuron × (1-p) (p is the probability of the time in which the neuron will remain off).

→ want to increase the speed of NN
(i) ↳ weight initialization
(ii) ↳ batch normalization
(iii) ↳ activation function
(iv) ↳ optimizer.

- If the updated weights are 0, it means that ~~there~~ no learning / trainning is done.
- If the taken random weights are larger, the problem of exploding gradient may occur.
  ~~If the tak~~ saturate the activation function.
- If the taken random weights are small, the problem of vanishing gradient may occur.

CamScanner

- Dense layer: Every neuron is connected to another neuron.

- Glorant / Xavier = instead of multiplying every neuron with random number, it will be multiplied by variance.

- fan_{in} is avoiding saturation and fan_{out} is used for overcoming / avoiding vanishing gradient.

Batch Normalization

- Normalizing neurons of output ~~layer too~~ of hidden layer too.
- different distribution of same classes in trainning & testing do not give accurate results. also known as covariate shift (name ko confirm karna)

- After normalization, we do scale and shift, β & Gamma "γ term is used.

$$BN = \gamma hi + \beta \rightarrow Mean$$

values of γ & β will also be updated like weights and biases.

Foundation for Advancement of Science & Technology

Lecture NO. 7
Gradient Descent

$w_{t+1} = w_t - \alpha \nabla w_t$

→ Momentum:

$w_{t+1} = w_t - v_t \Rightarrow w_{t+1} = w_t - (\beta v_{t-1} + \eta \nabla w_t)$

$v_t = \beta v_{t-1} + \eta \nabla w_t$

→ NAG: (Nesterov ~~Adaptive~~ Accelarated Gradient) → To minimize oscillation

$w_{t+1} = w_t - v_t$

$v_t = \beta v_{t-1} + \eta \nabla w_{la} \Rightarrow v_t = (w_t - w_{la}) + \eta \underline{w_{la}}$ → look ahead term

$w_{la} = w_t - \beta v_{t-1} \Rightarrow \beta v_{t-1} = w_t - w_{la}$

→ AdaGrad: (Adaptive Gradient) → RMS prop

$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} \nabla w_t$ $\qquad$ $w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} \nabla w_t$

$v_t = v_{t-1} + (\nabla w_t)^2$ $\qquad$ $v_t = \beta v_{t-1} + (1-\beta)(\nabla w_t)^2$

→ Bias Correction:

$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ → epoch

$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

→ Adam

$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} + \hat{m}_t$

moment um

$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla w_t$

Ada Grad ↔ $v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla w_t)^2$

→ To speedup Neural Network trainning

- weight initialization
- batch normalization
- activation function
- optimizer

CamScanner

Foundation for Advancement of Science & Technology

•) velocity is always 0 in 1st iteration.

•) We added term "momentum" for speed.

→ Challenges

- learning rate
- learning rate schedules
- multiple weight optimal find
- saddle poin → curvature → local minima

→ Exponential weighted mean average

$w_t = \beta w_{t-1} - (1-\beta) w_t$

previous history

•) Pichle wale ko zayada importance, aage wale ko km.

$w_0 = 0$

$w_1 = 0.95 w_0 + 0.05 w_1$

$w_2 = 0.95 w_1 + 0.05 w_2$

$= 0.95(0.95 w_0 + 0.05 w_1) + 0.05 w_2$

$= 0.9025 w_0 + 0.0475 w_1 + 0.05 w_2$

Previous History is added & then new weight is updated.

→ when to update, we also update the current weight.

- In 100k ahead, we find slope & direction.

speed which gives oscillations and will not let it to stuck in global minima

Lecture No. 8

→ Moment:

$w_{t+1} = w_t - v_t$

$v_t = \beta v_{t-1} + \eta \nabla w_t$

Stochastic Gradient Descent with Momentum

↳ Batch size = 1, $w = 0$, $v_0 = 0$

Sample 1 $(1.1, 39)$ $\beta = 0.95$

$\eta = 0.05$

Ex:

Year	Salary
1.1	39
1.3	46
2	43

epoch 1

$t = 1$ $\hat{y} = wx$

$\text{loss} = (y - \hat{y})^2$

$= (39 - 0)^2 = (39)^2$

$= 1521$

weight update:

$w^{new} = w^{old} - \eta \nabla w$

Epoch 1

$w_1^{new} = w_1^{old} - v_1$

$w_1^{new} = 0 - (-4.29)$

$w_1^{new} = 4.29$

$v_1 = -4.29$

$v_1 = \beta v_0 + \eta \nabla w_1$

$v_1 = \beta v_0 + \eta(-2x(y - \hat{y}))$

$v_1 = 0.95(0) + 0.05(-2(1.1)(39 - 0))$

$v_1 = -4.29$

$\frac{\partial L}{\partial w} = \frac{\partial (y - \hat{y})^2}{\partial w}$

$= 2(y - \hat{y})(-x)$ $mx + c$

$= -2x(y - \hat{y})$

Epoch 2

x y

$(1.3, 46)$

$\hat{y} = w_1 x = 0$

$\text{LOSS} = (y - \hat{y})^2$

$= (46 - (w_1 x))^2$

$= (46 - (4.29(1.3)))^2$

$= 1634$

$w_2^{new} = w_1^{old} - v_2$

$= 4.29 - (-9.33) = 13.62$

$v_2 = \beta v_1 + \eta(-2x(y - \hat{y}))$

$= 0.95(-4.29) + 0.05(-2(1.3)(46 - 5.577))$

$= \cancel{-105.11}$ (re-check)

$= \cancel{-5.25}\ -9.33$

$\cancel{\frac{\partial L}{\partial w_{la}} \cdot \frac{\partial y}{\partial w_{la}} \cdot (wx}$

→ NAG

first we find momentum and then gradient

Year	Salary
1.1	39
1.3	46

$w_{t+1} = w_t - v_t$

$v_t = \beta v_{t-1} + \eta \nabla w_{la}$

$w_{la} = w_t - \beta v_{t-1}$

• iteration 1 : (epoch = t = 1)

$w_0 = 0$, $v_0 = 0$ $\quad (1.1, 3.9)$ $\quad \beta = 0.95$

$\hat{y} = w_0 x = 0$ $\quad \eta = 0.05$

$loss = (y - \hat{y})^2 = 1521$

$\frac{\partial L}{\partial w_{la}} = \frac{\partial}{\partial w_{la}} (y - wx$

$\downarrow$

$w_t - v_t$

$\downarrow$

$w_{la} + \beta v_{t-1}$

$= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w} \cdot \frac{\partial w}{\partial w_t} \cdot \frac{\partial w_t}{\partial w_{la}}$

$= 2(y - \hat{y})(x) 1.1$

$= -2x(y - \hat{y})$

update weight

$w_1^{new} = w_1^{old} - v_1$

$v_1 = 0.95(0) + 0.05$

→ AdaGrad:

(Adaptive Gradient)

↳ used when the dataset is sparsed.

↓

mostly values in dataset are zero.

Eg

Roll No	IS CGPA 40% not
32	0
34	0
36	1

$$w_{t+1}^{new} = w_t^{old} - \frac{\eta \, \Delta w}{\sqrt{V_t + \varepsilon}}$$

∵ V_t → stores all previous rates

$$V_t = V_{t-1} + (\nabla w_t)^2$$

→ ∵ taking square bcz we only want magnitude not direction

→ Problems:

If $\frac{\eta}{\sqrt{V_t}}$ value is very large then w_{t+1} will be very small. In case of vice versa, the updation will be close to none.

→ RMS props: (mostly work on RNN's)

Advantage: •) oscillation is dampened → kam kr deta ha

•) saddle point (faster movement)

→ Adam: → works on ANN, CNN, RNN ~~cannot work on complex problems~~

→ momentum, NAG → momentum focuses on

→ AdaGrad, RMS props → learning rate

} combination = Adam.

→ Bias correction:

To ~~avoid~~ correct slow updation in the initial states. (To avoid faster oscillations → recheck)

Foundation for Advancement of Science & Technology

works good in CNN

Batch Normalization

Batch with 3 samples

					mean	std
features	x_1	1	3	8		
	x_2	3	4	3		
	x_3	5	6	2		
	x_4	7	2	1		

Normalization across mini batch independently for each features

Layer Normalization

Batch with 3 samples.

	sample 1	sample 2	sample 3
x_1	1	3	8
x_2	3	4	3
x_3	5	6	2
x_4	7	2	1
mean			
std			

works good in RNN and Transformers.

Normalization across features, independently for each sample.

CamScanner

Foundation for Advancement of Science & Technology

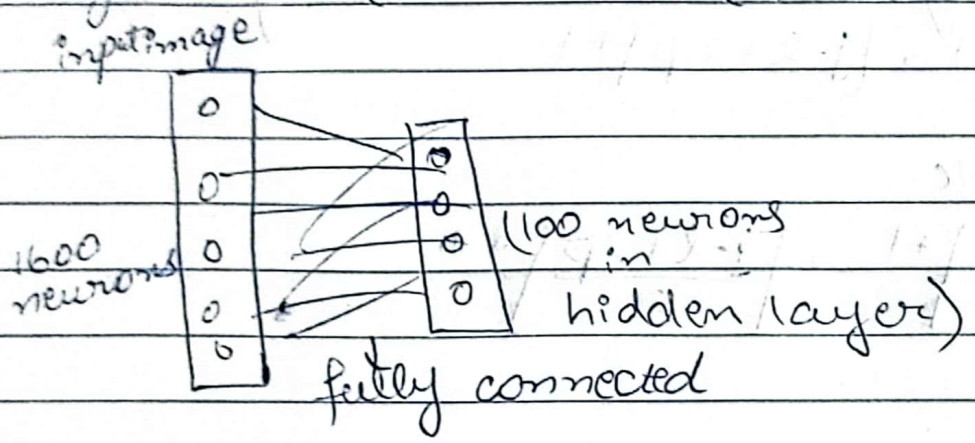
Lecture NO. 9

Convolutional Neural Network

Problems of ANN

↳ images, time series data, computational power

image : 40×40 (1600 neurons (1 pixel = 1 neuron))



Total weight = 1600 × 100 = 160,000

CNN gives better results than ANN.

↳ use filters/kernals

↓

used for feature extraction.

layers of CNN:

→ convolution

~~→ potential~~

→ pooling

→ fully connected layers.

Ex: image (5×6)

8	6	5	2	19	255
0	8	6	2	255	255
0	6	2	2	255	255
0	9	2	8	255	255
0	8	0	4	255	255

-1	-1	-1
0	0	0
1	1	1

CamScanner

• jb convolution kri n ge, agr value -ve hui to ans 0 ho jaye ga n agr 255 se value bri aa gayi to uss ko hum 255 kr dein ge.

• Convolution

image 5×6 ($i \times j$) filter/Kernal (3×3) ($k \times l$)

$[i-k]+1 , [j-l]+1$

$[5-3]+1 , [6-3]+1$

$3 , 4$

Resultant image (3×4) (rows, col)

in case of padding, above formula changes to

$[i-k+2p]+1 , [j-l+2p]+1$

after adding stride

$\left\lfloor \frac{i-k+2p}{s} \right\rfloor +1 , \left\lfloor \frac{j-l+2p}{s} \right\rfloor +1$

• Pooling:

To minimize size. Average & Max Pooling are most commonly used.

Ex:

		# of parameters	dimension
input →	(10, 10, 1) (height, width, channel)	0	10, 10, 1
conv →	(5, 3, 3) (no. of filters, height, width)	$(3 \times 3 \times 1 + 1) * 5 = 50$	$[i-k+1] \cdot [10-3]+1 = 5 \times (8 \times 8)$
activation (sigmoid)		0	$5 \times 8 \times 8$
Conv2D (2, 3, 3)		$(3 \times 3 \times 5 + 1) \times 2 = 92$	$2 \times (6 \times 6)$
activation (sigmoid)		0	$2 \times 6 \times 6$
output		$50 + 92 = 142$	$2 \times (6 \times 6)$

→ Formula to find # of parameters:
(filter width * filter height * # of filters in previous layer + 1) * # of filters in this layer

in case, agr input ke baad direct convolution aa jata ha to "# of filters in previous layer" ki jga "# of channels aaye ga.

(filter width * filter height * # of channels + 1) * # of filters in this layer

*	# of parameters	dimention
pooling	0	2x(3x3)
Fully connected Dense (64)	1216	

Formula for fully connected layer: (# of parameters)
(current layer neuron × previous layer neuron) + current layer neuron
= 64 × (2*3*3) + 64
= (64 × 18) + 64 = 1216

Hyper parameters & parameters in ~~Con~~ CNN

stride, pooling types } pooling layers ki → no parameters.

filter size, height, width, stride, padding, # of filters, activation function.

filter length, filter height, values in param formula