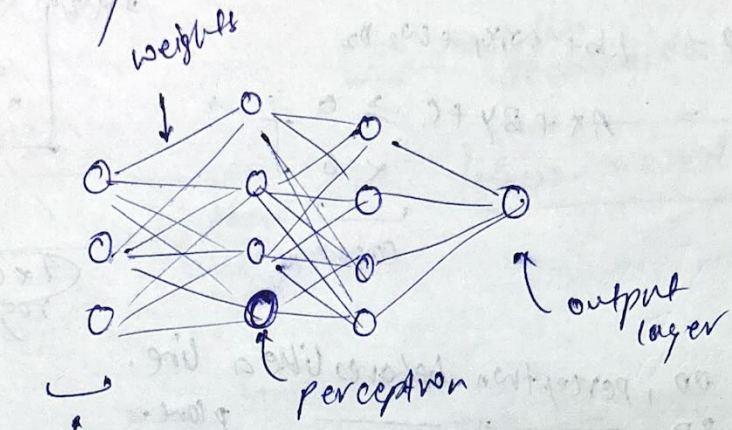
Deep learning

machine learning → uses statistics
deep learning → uses neurons



Neural Networks

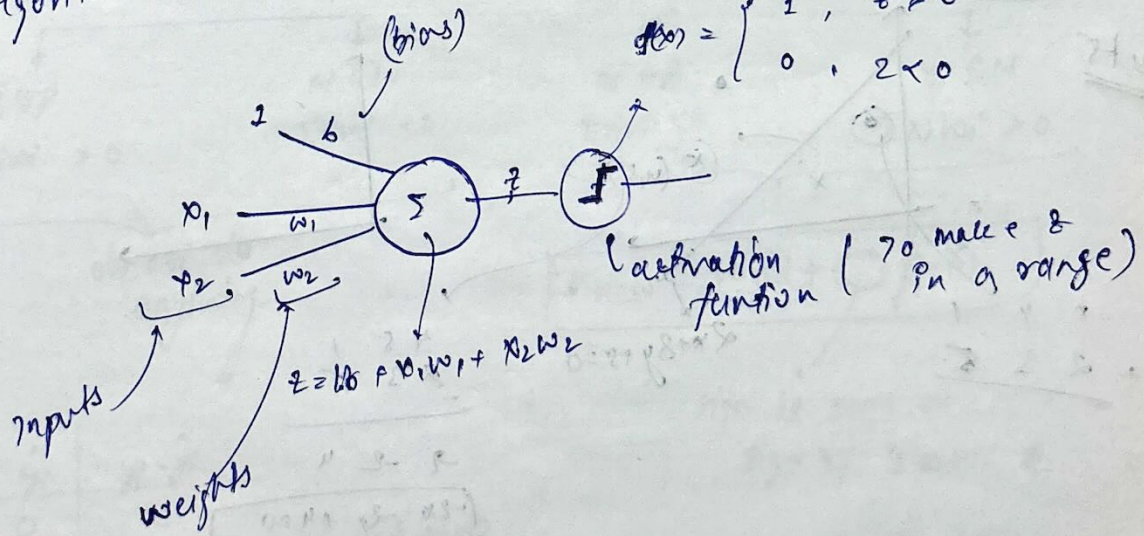
ANN, CNN, RNN, GAN
CNN → image
RNN → speech text
GAN → generate image

Lect 4

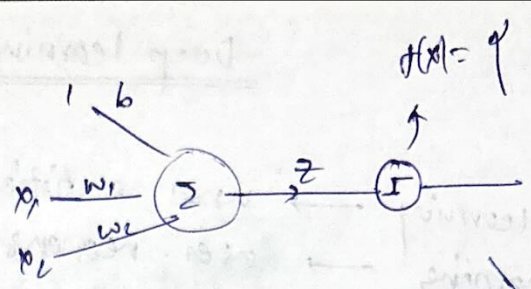
Perceptron

↓

Algorithm (mathematical model / function)



cgpa	IQ	salary
x_1	x_2	

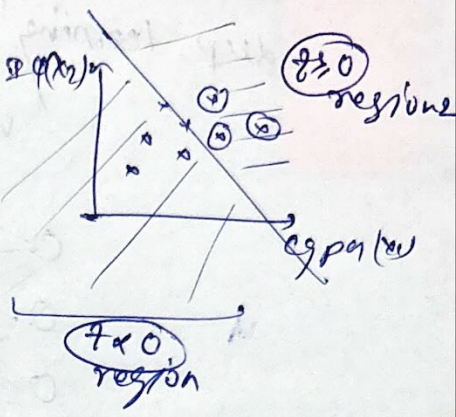


$z = 1b + w_1x_1 + w_2x_2$

$= Ax + By + C \geq 0$

< 0

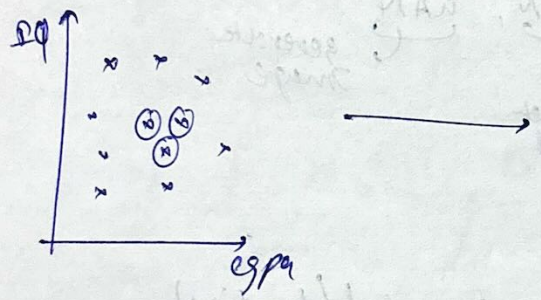
cases



Note] (1) in 2D, perceptron behaves like a line.
(2) 3D ——→ —— plane.
(3) ND ——→ —— hyperplane

× → unplaced
⊗ → placed.

But

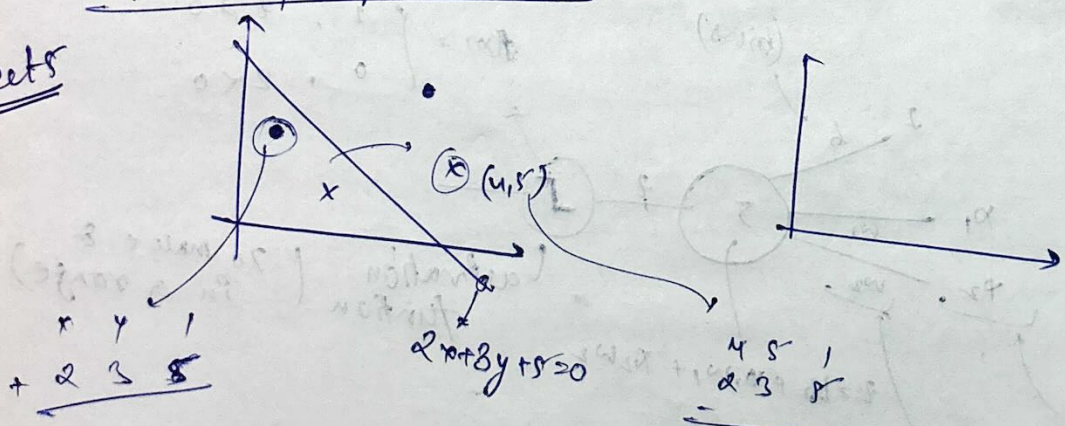


→ here, accuracy ↓ perceptron is not much good.

data should be linearly separable.

Soln with perceptron trick (method 1)

leets



x	y	1
2	3	5

4	5	1
2	3	5

→ 2 2 4

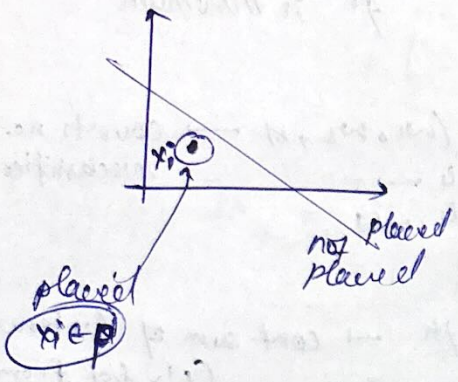
$-2x - 2y + 4 \geq 0$

$$Coef_{new} = Coef_{old} - p(i)(coord)$$

$z = w_1 x_1 + w_2 x_2 + w_0 \cdot x_0$

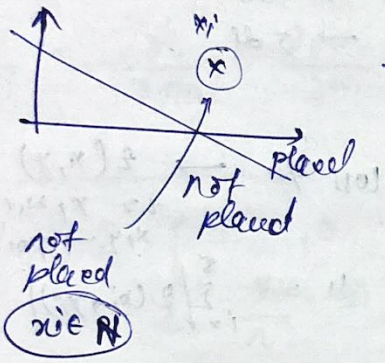
$x_0 = 1$

$z = \sum_{i=0}^{2} w_i x_i$



→ move line downwards

$w_{new} = w_{old} - \eta \cdot x_i$



→ move line upwards

$w_{new} = w_{old} + \eta x_i$

Finally

x_i , $\sum w_i x_i$

total cases

$x_i \in P$	$x_i \in N$	$x_i \in P$	$x_i \in N$
$\sum x_i w_i > 0$	$\sum x_i w_i < 0$	$\sum x_i w_i < 0$	$\sum x_i w_i > 0$
do not do anything		$w_{new} = w_{old} + \eta \cdot x_i$	

sign is same as sign of $\sum x_i w_i$

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
0	0	0
1	1	0
0	1	-1
1	0	1

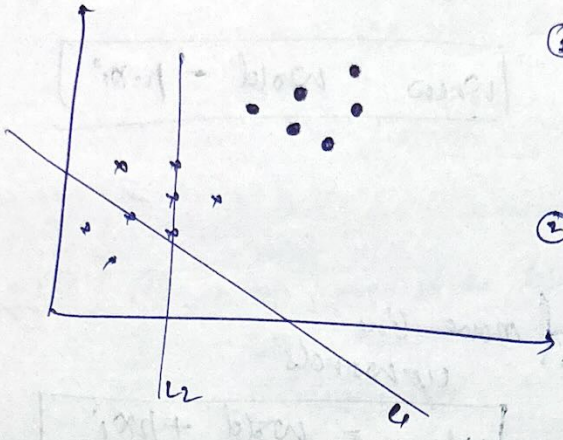
for all cases

$w_{new} = w_{old} + \eta (y_i - \hat{y}_i) \cdot x_i$

Lect 6 soln with loss function (method 2)

loss fn = $f(w_1, w_2, b)$

w_1, w_2, b → for which fn is minimum



① f(w1, w2, b) → counts no. of misclassified points
L1 → 5
L2 → 1

② loss fn → cont sum of distance (L1's dist from line)
L1 → Σdi
L2 → Σdi

③ loss fn → z(x, y)
z = x1 w1 + x2 w2 + b
x, y → point
L1 = $\sum_{i=1}^{5} |z(x_i, y_i)|$

scikit-learn Impl

$$\text{Loss} = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

$f(x_i) = w_1 x_1 + w_2 x_2 + b$

	y_i	$\hat{y}_i$	$\max(0, -y_i f(x_i))$
good points	1	1	0
	0	0	0
misclassified points	1	0	$y_i \cdot f(x_i) = \lvert f(x_i)\rvert$
	0	1	$y_i \cdot f(x_i) = \lvert f(x_i)\rvert$

only for handling sign.

solving with gradient descent

$$\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial b}$$

$$\text{Loss fn} = \frac{1}{n}\sum \underbrace{\max(0, -y_i f(x_i))}$$

where, $f(x_i) = w_1 x_1 + w_2 x_2 + b$

$$= \begin{cases} 0 & , -y_i f(x_i) < 0 \\ -y_i f(x_i) & , -y_i f(x_i) \geq 0 \end{cases}$$

(L) $$\text{Loss fn} = \sum \begin{cases} 0 & , -y_i f(x_i) \leq 0 \\ \frac{-y_i f(x_i)}{n} & , -y_i f(x_i) > 0 \end{cases}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f(x_i)} \cdot \frac{\partial f(x_i)}{\partial w_1}$$

$$\frac{\partial L}{\partial w_1} = \sum \begin{cases} 0 & , -y_i f(x_i) < 0 \\ \frac{-y_i}{n} \cdot x_{i1} & , -y_i f(x_i) > 0 \end{cases}$$

$$\frac{\partial L}{\partial w_2} = \sum \begin{cases} 0 & , \text{---} \\ \frac{-y_i}{n} \cdot x_{i2} & , \text{---} \end{cases}$$

$$\frac{\partial L}{\partial b} = \sum \begin{cases} 0 & , \text{---} \\ \frac{-y_i}{n} & , \text{---} \end{cases}$$

for i in epochs:

$$w_2 = w_2 - \eta \cdot \left(\frac{\partial L}{\partial w_2}\right)$$

$$w_1 = w_1 - \eta \left(\frac{\partial L}{\partial w_1}\right)$$

$$b = b - \eta \left(\frac{\partial L}{\partial b}\right)$$

Note If activation fn is sigmoid

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

where, $z = w_1 x_1 + w_2 x_2 + b$

and loss fn is binary cross entropy

$$L = -y_i \log \hat{y}_i + (1-y_i) \cdot \log(1-\hat{y}_i)$$

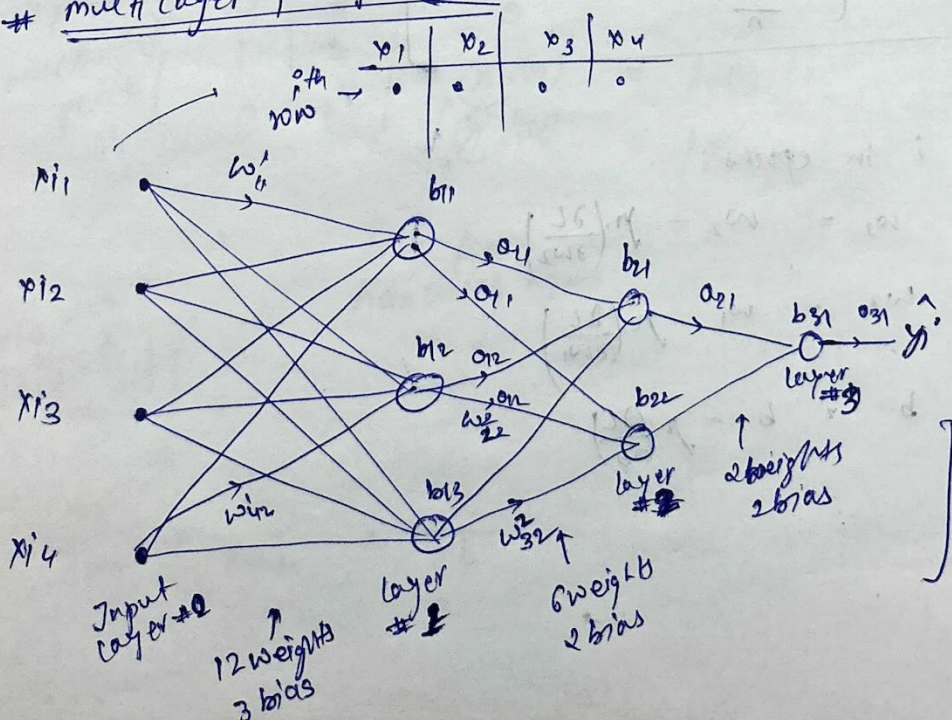
output → probability.

Here, perceptron = logistic regression

Note

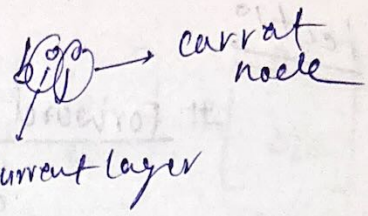
loss fn	activation fn	output
Hinge loss	step	perceptron (0, 1)
log-loss (binary cross entropy)	sigmoid	Logistic reg (0, 1) → Range Probability.
categorical cross entropy	softmax	(0, 1)
mse	linear	linear regression.

Recall # multilayer perceptron (mlp) Notation



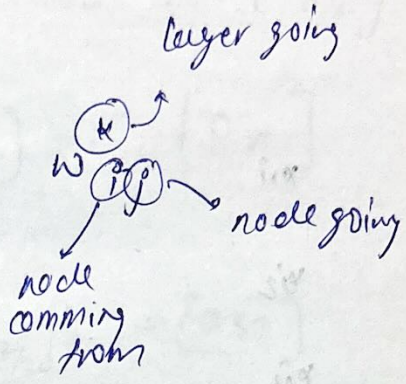
o_{ij}, b_{ij} → i = current layer
j = current node

o_{ij} → output
b_{ij} → bias

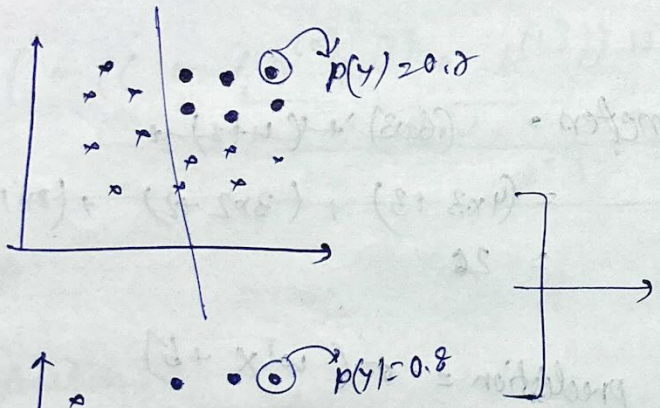


~~Weight~~ ~~i = current node~~
~~j = current node~~
~~k = incoming layer.~~

w^k_{ij} → i = current node
j = going node
k = going layer.

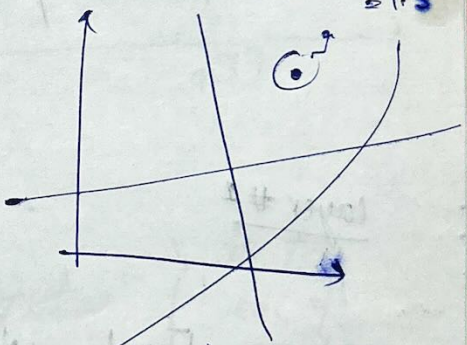


Lect 9] MLP



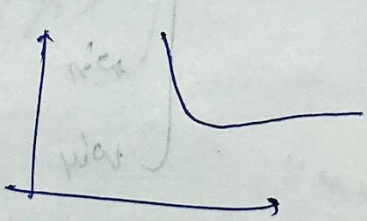
can also use weighted sum

$p(y) = 0.7 + 0.8 = 1.5$



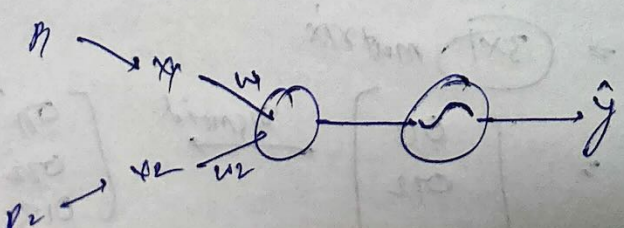
↓ smoothening

can also give bias.



sigmoid

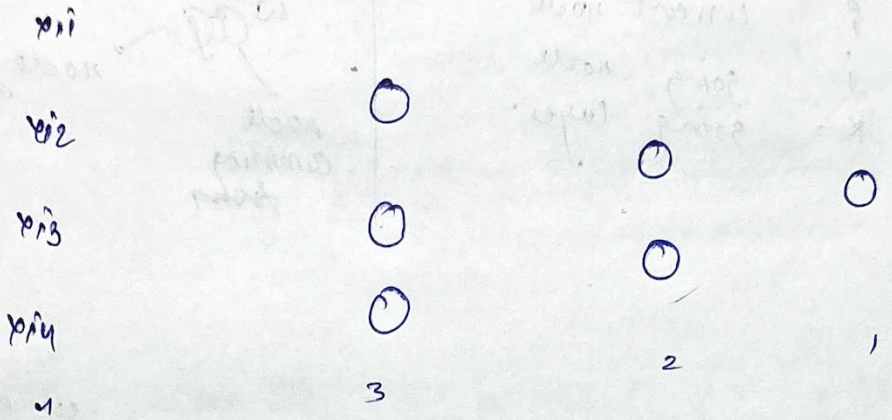
$\frac{1}{1+e^{-1.5}} = 0.82$



lect 10

Forward propogation

cgpa	IQ	10th m	12th m	placed



trainable parameters = ~~(4×3)+(3×2)+~~

$= (4\times3+3) + (3\times2+2) + (2\times1+1)$

$= 26$

layer #1

$$\text{prediction} = \sigma(W^T X + b)$$

$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{43} \end{bmatrix}^T \cdot \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$(4\times3)^T \longrightarrow (3\times4)$ $\quad 4\times1$ $\quad (3\times1)$

$(3\times1) + (3\times1)$

$= (3\times1)$ matrix

$$= \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} \xrightarrow{\text{sigmoid}} \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix}$$

layer #2

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix}^T \cdot \begin{bmatrix} o_{21} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

sigmoid

$$\begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

$$\begin{bmatrix} w^3_{11} & \\ w^3_{21} & \end{bmatrix}^T \cdot \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix} = \begin{bmatrix} o_{31} \end{bmatrix}$$

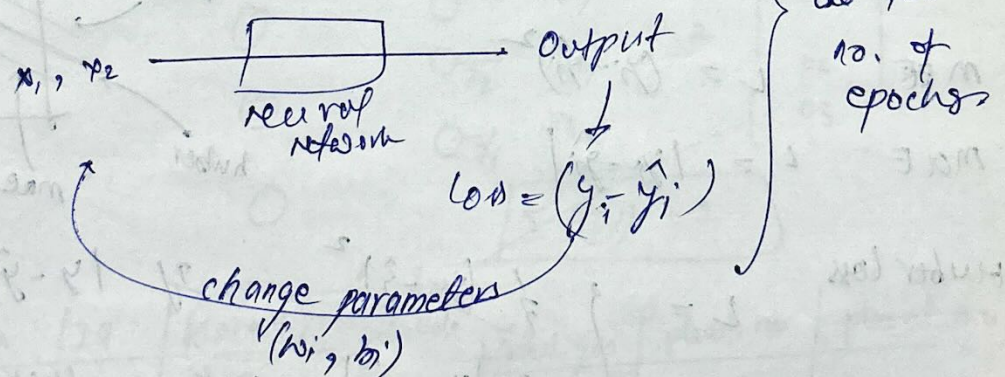
$$\therefore \quad \sigma\left(\begin{bmatrix} o_{11} \end{bmatrix}\right) = \begin{bmatrix} o_{31} \end{bmatrix}$$

$$\text{output} = \sigma\Big(\sigma\big(\sigma\big(\underbrace{\sigma\big(\underbrace{a^{[0]} w^{[1]} + b^{[1]}}_{a^{[1]}}\big)}_{a^{[2]}} w^{[2]} + b^{[2]}\big) w^{[3]} + b^{[3]}\big)\Big)$$

$a^{[3]}$

lecture 11/12/13/14

~~Loss function~~



same as linear regression.

Loss fn in DL

Regression
- mse (L_2 loss)
- mae
- number loss

classification
- binary cross entropy
- categorical — " —
- hinge loss

GAN
- discriminator loss
- min max gan loss

object detection
- focal loss

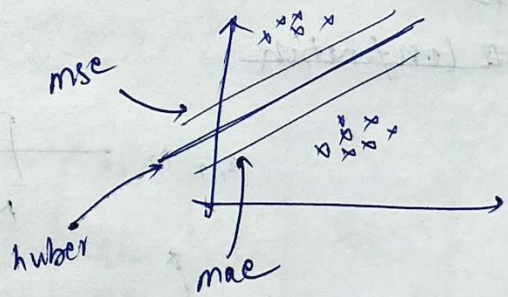
Auto encoders
- KL divergence

Embedding
- Triplet loss.

loss v/s cost fn

$(y_i - \hat{y}_i)^2$ ⟶ for single data] loss / error fn

$\sum \frac{(y_i - \hat{y}_i)^2}{n}$ ⟶ for whole data] cost fn



MSE $L = (y_i - \hat{y}_i)^2$

MAE $L = |y_i - \hat{y}_i|$

Huber loss

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

Binary cross Entropy (log loss)

→ Classification (two classes → 0/1)

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

→ output must be sigmoid activation.

↳ hidden layer activation can be anything.

categorical cross entropy

→ Classification (multi class)

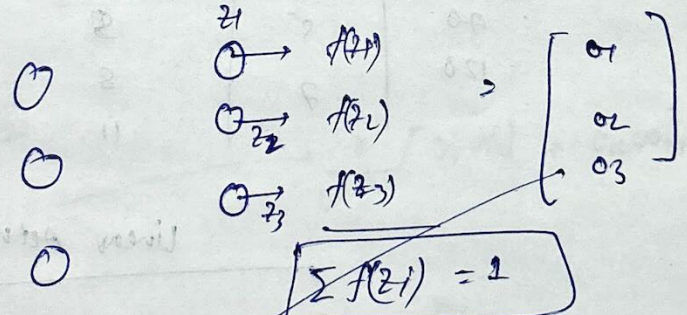
$$L = -\sum_{j=1}^{K} y_i \log(\hat{y}_i) \quad , \quad K = \text{is no. of classes.}$$

→ output must be softmax activation

$$f(z_i) = \frac{e^{z_i}$$

$$\sum e^{z_i}$$

↳ hidden layer Act can be anything



Ex

cgpa (x_1)	IQ (x_2)	placed
		yes
		no
		maybe

one hot encode

placed_yes	placed_no	placed_maybe
0	0	1
1	0	0
0	1	0

$[0.2 \quad 0.3 \quad 0.5]$
$[1 \quad 0 \quad 0]$ → loss = L

cgpa	IQ	placed
7	70	yes
8	80	No
6	60	maybe

categorical cross entropy

placed → one hot encode

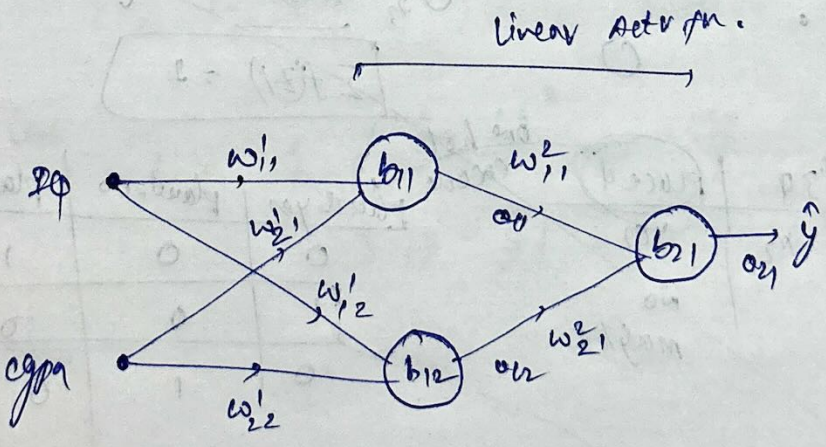
Sparse categorical cross entropy

placed → Ordinal Encode → (1,2,3)

Part 15

Back propagation

IQ	cgpa	LPA
80	8	3
60	9	
70	5	5
120	7	8
		11



steps

→ 1) Initialize w_i, b_i, anything

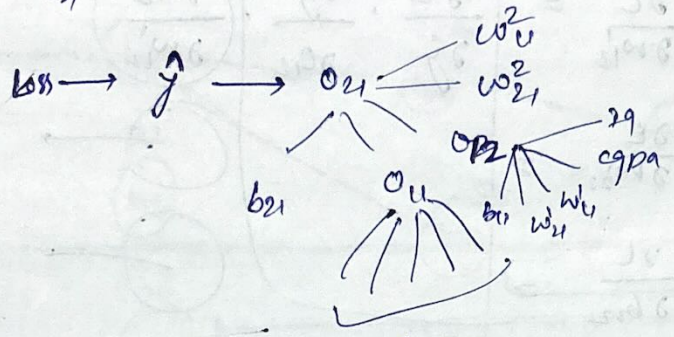
$\boxed{w \to 1,\ b \to 0}$

2) select a student

3) predict (cgpa) → forward prop

4) choose a loss fn. (mse)

$$o_{21} = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$$



5) update weights and bias

↳ gradient descent

$$\boxed{w_{new} = w_{old} - \eta\left(\frac{\partial L}{\partial w_{old}}\right)}$$

$\frac{\partial L}{\partial w_{11}^2}, \frac{\partial L}{\partial w_{21}^2}, \frac{\partial L}{\partial b_{21}}$

$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{11}^2}$$

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

$$\frac{\partial}{\partial w_{11}^2}\left[o_{11} w_{11}^2 + o_{12} w_{21}^2 + b_{21}\right] = o_{11}$$

$$\boxed{\frac{\partial L}{\partial w_{11}^2} = -2(y-\hat{y}) o_{11}}$$

$$\boxed{\frac{\partial L}{\partial b_{21}} = -2(y-\hat{y}) \cdot 1}$$

$$\boxed{\frac{\partial L}{\partial w_{21}^2} = -2(y-\hat{y}) o_{12}}$$

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{11}^1} \rightarrow x_{i1} = x_{i1}$$

$$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial b_{11}} \rightarrow 1$$

$$\frac{\partial L}{\partial w_{21}^1} = \rightarrow \text{copy} = x_{i2}$$

$$\frac{\partial L}{\partial w_{12}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial w_{12}^1} \rightarrow x_{i1} = x_{i1}$$

$$\frac{\partial L}{\partial w_{22}^1} = \rightarrow \text{copy} = x_{i2}$$

$$\frac{\partial L}{\partial b_{12}} = \rightarrow 1$$

$$\frac{\partial \hat{y}}{\partial o_{11}} = \frac{\partial}{\partial o_{11}}\left[w_{11}^2 o_{11} + w_{21}^2 o_{21} + b_{21}\right] = w_{11}^2$$

$$\frac{\partial \hat{y}}{\partial o_{12}} = \frac{\partial}{\partial o_{12}}[\quad\quad] = w_{21}^2$$

↓

now update the parameters and repeat for n no. of epochs.

lect 16 → code implementation
* gradient descent

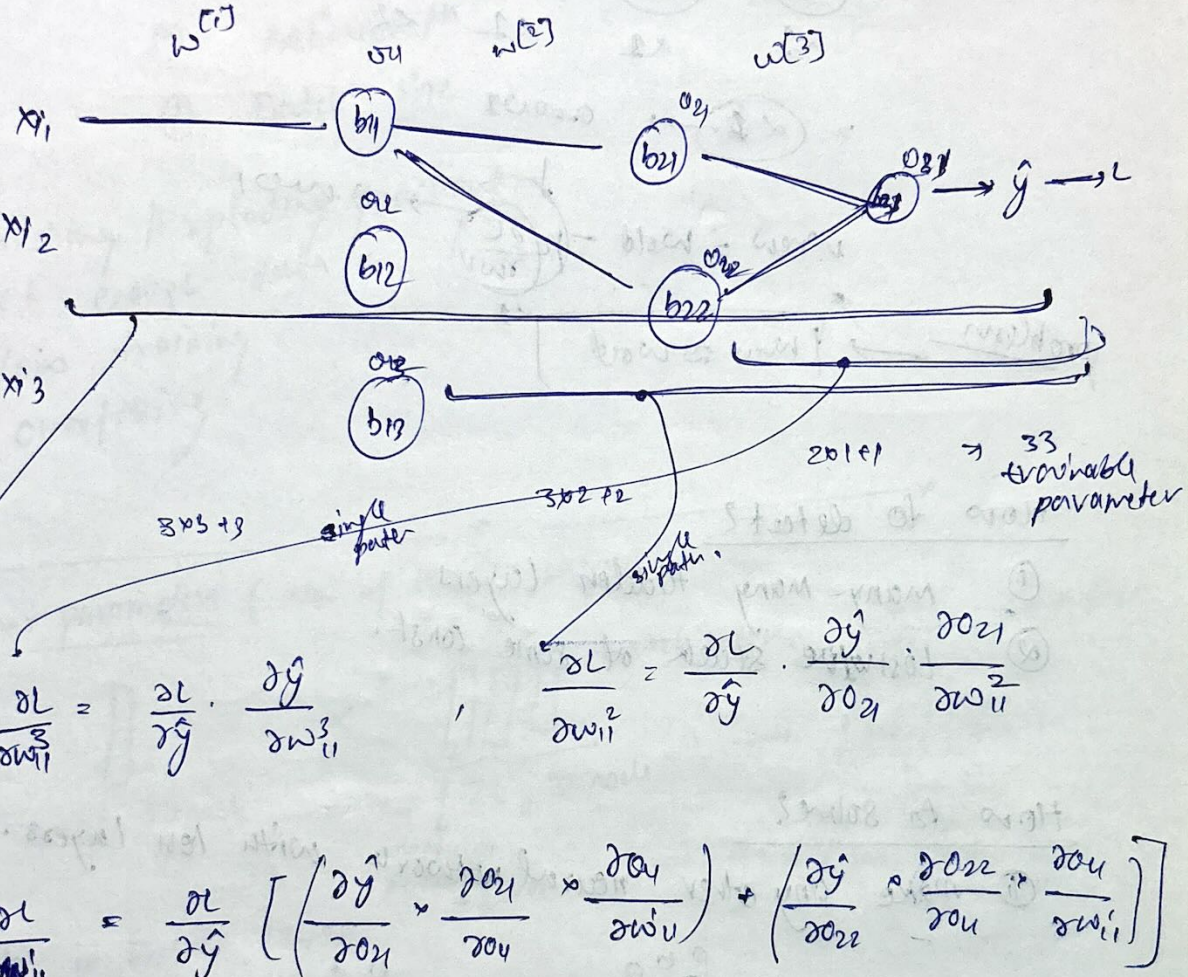
lect 17 → MLP Explained in detail

lect 18

MLP memoization

↓

It means to store the computed result for some input (solving fibonacci no with dp).



$$\frac{\partial L}{\partial w_{11}^3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{11}^3}, \qquad \frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w_{11}^2}$$

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \left[\left(\frac{\partial \hat{y}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{11}^1} \right) + \left(\frac{\partial \hat{y}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{11}^1} \right) \right]$$

Lect 20

Vanishing Gradient problem

where found?

① deep neural network like 10-15 hidden layers

② $0.1 \times 0.1 \times 0.1 \times 0.1 = 0.0001$

③ sigmoid / tanh activation fn generally.

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial a_{11}} \times \frac{\partial a_{11}}{\partial w_{11}}$$

$<1 \quad <1 \quad <1 \quad <1$

$= <1$ → a.wo1

$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{11}}$ → o.wo1

<1

problem → $w_{new} \approx w_{old}$

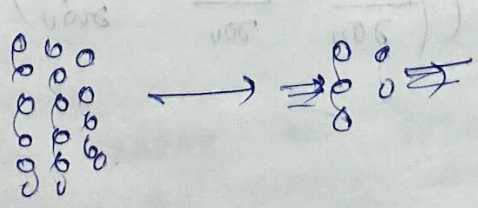
How to detect?

① many-many Hidden Layers

② Loss value stuck at some const.

How to solve?

① make any other neural network with less layers.



② using RELU activation fn.

$= \max(0, z)$

$= \begin{cases} 0, & z < 0 \\ z, & \text{otherwise} \end{cases}$

③ proper weight init → glorot / xavier

④ Batch normalization

⑤ Residual network
Building block

lecture

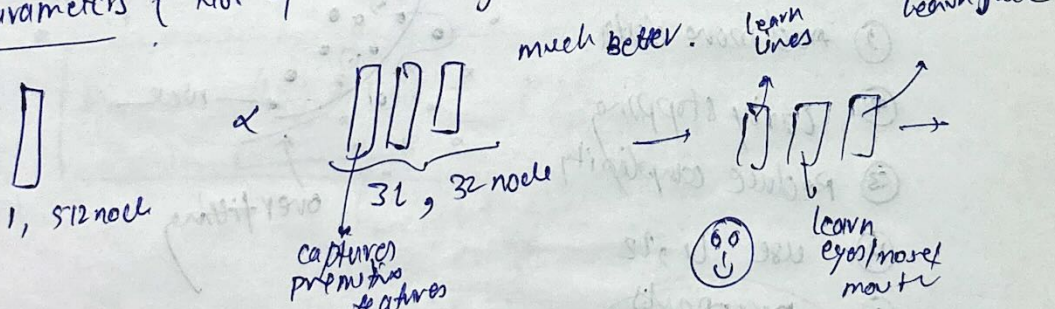
How to Improve a Artificial Neural Network (ANN)

① Fine tuning NN hyperparameters.
↳ (a) hidden layers
(b) node per layers
(c) learning rate
(d) optimizer
(e) Activation fn
(f) Batch size

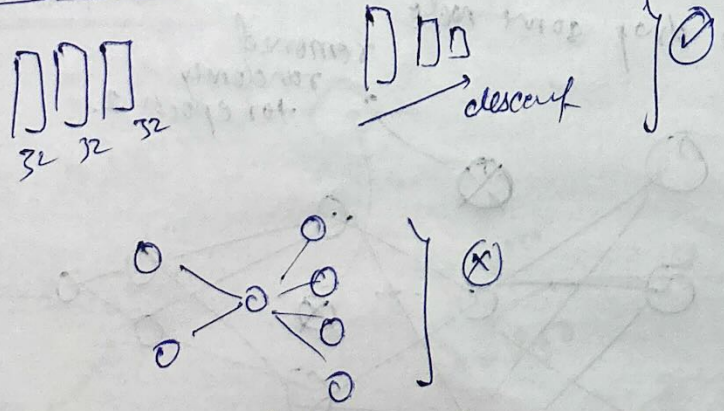
② vanishing / Exploding gradient -
③ Not enough data
④ slow training
⑤ overfitting

---

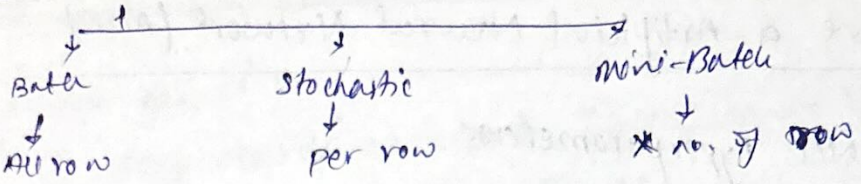
① Hyper parameters (No. of hidden layers).



② Node in hidden layers



③ Batch size

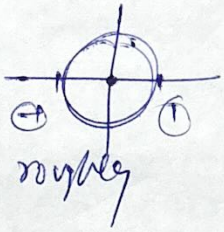


④ Epochs → use early stopping

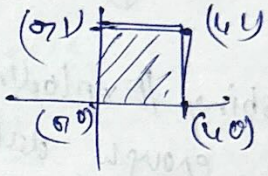
lect 23

Do scaling

standard



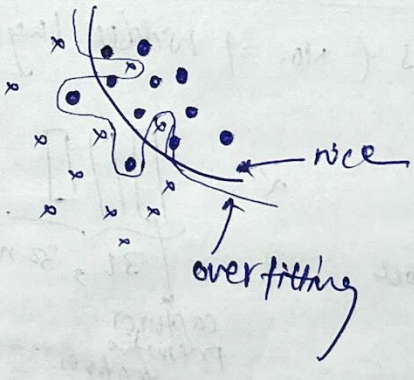
min max



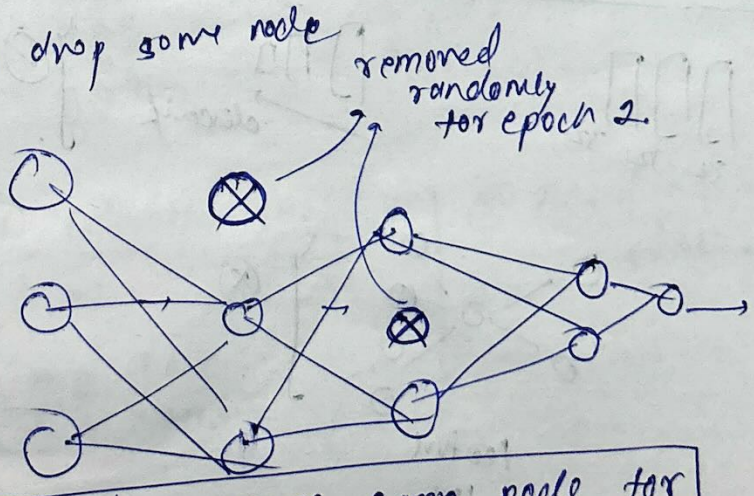
lect 24

Dropout layers

① Add more data
② Early stopping
③ Reduce complexity
④ use L1, L2
⑤ Dropouts



↓
you drop some node



→ Randomly remove some node for each epoch.

→ ANN p(10 - 50%)
RNN p(20 - 30)
CNN p(40 - 50)
↑ prop of dropout

lect 26

Regularization

more complex NN ∝ more capable is the network to make curve fit training data. (overfitting).

$$\text{Cost} = \frac{1}{n}\sum l(y_i, \hat{y}_i) + \text{penalty term}$$

$$(\text{regularization})^{L_1} = \frac{\lambda}{2n}\left[w_1^2 + w_2^2 + \cdots + w_w^2\right]$$

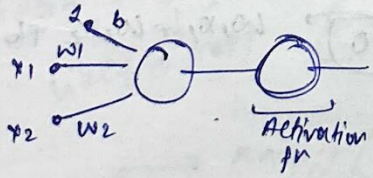
$$(\text{regularization})^{L_2} = \frac{\lambda}{2n}\sum \|w_i\|^2$$

$$w_{new} = (1 - \mu\lambda)\, w_{old} - \eta\left(\frac{\partial L}{\partial w_{old}}\right)$$

weight decay

lect 27

Activation function

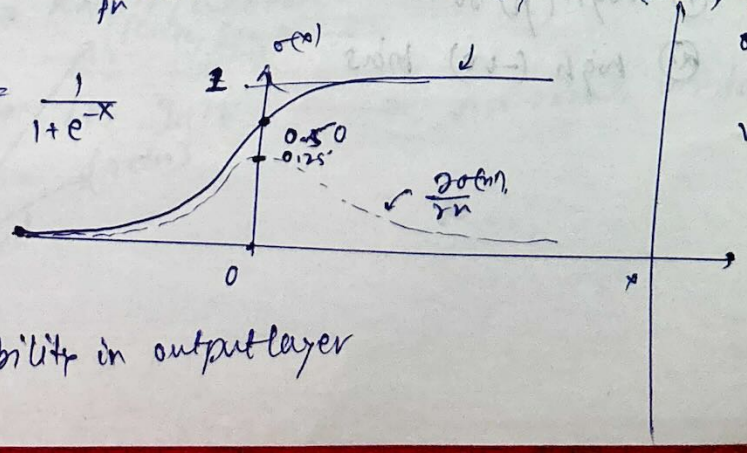


① Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

$\sigma(x) \in [0, 1]$

↓

can be used as probability in output layer



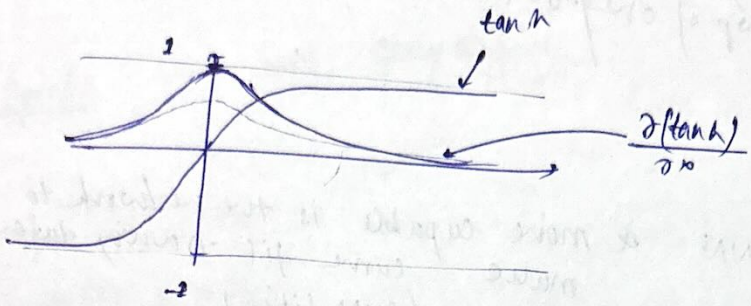
$\frac{\partial \sigma(x)}{\partial x}$

↓

vanishing GD problem occurs.

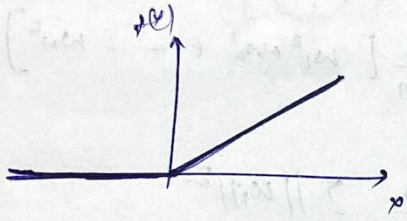
Tanh

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Relu

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



Dying relu problem

Dead neuron = Node to be output zero.

If one time it give zero, then always it will give zero.

Hence If 50% N are dead its problem.

Dead neuron → $\frac{\partial L}{\partial w} = 0$ → hence no update in parameters.

when $z < 0$ → $w_1 x_1 + w_2 x_2 + b$

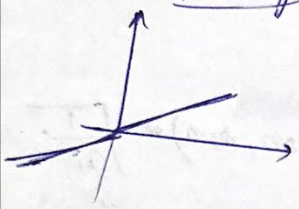
happens because of

① high (+) rate

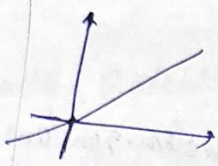
② high (-ve) bias

Relu variants

- Linear
  - leaky
  - parametric
- non-linear
  - Elu
  - selu



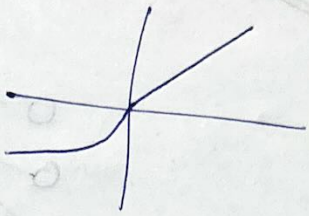
$f(x) = \max(0.01x, x)$



$f(x) = \begin{cases} x, & x \geq 0 \\ ax, & \text{otherwise} \end{cases}$

a is variable

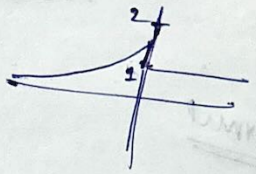
$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$



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

$ELU'(x) = \begin{cases} 1, & x > 0 \\ ELU(x) + \alpha, & x \leq 0 \end{cases}$

$\alpha = 1.673$
$\lambda = 1.0507$ } fixed parameters



$SELU'(x) = \lambda \begin{cases} 1, & x > 0 \\ \alpha e^x, & x \leq 0 \end{cases}$

Lecture

Weight initialization

1. $w_i = 0$ (Don't do it) — Sigmoid AFn
2. Non-zero const (Don't do it)
3. Init it with random values ✗
4. Random too small / Large values.
   - like (0.001)
   - (3-5)

Lecture

soln →
- Xavier / Glorot (tanh, sigmoid) → normal, uniform
- He Init (relu) → normal, uniform

Suggested to use

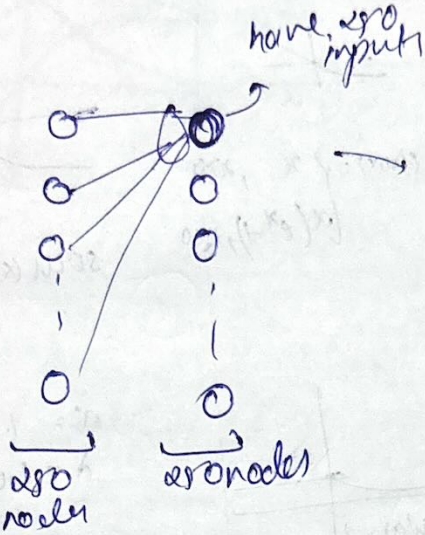
Small

np.random.randn(250, 250) * 0.01

Large

np.random.randn(250, 250) * 1.

Xavier initialization



np.random.random(250, 250) * $\sqrt{\frac{1}{n}}$

n: no. of inputs.

He Normal

weights = (⟶) * $\sqrt{\frac{2}{fan\_in}}$ ← no. of inputs.

Xavier uniform

weights range

$[-limit, limit]$

$limit = \sqrt{\frac{6}{(fan\_in + fan\_out)}}$

He Uniform

$[-limit, limit]$

$limit = \sqrt{\frac{6}{fan\_in}}$

Lect 31

Batch Normalization → mean = 0, $\sigma = 1$

↓

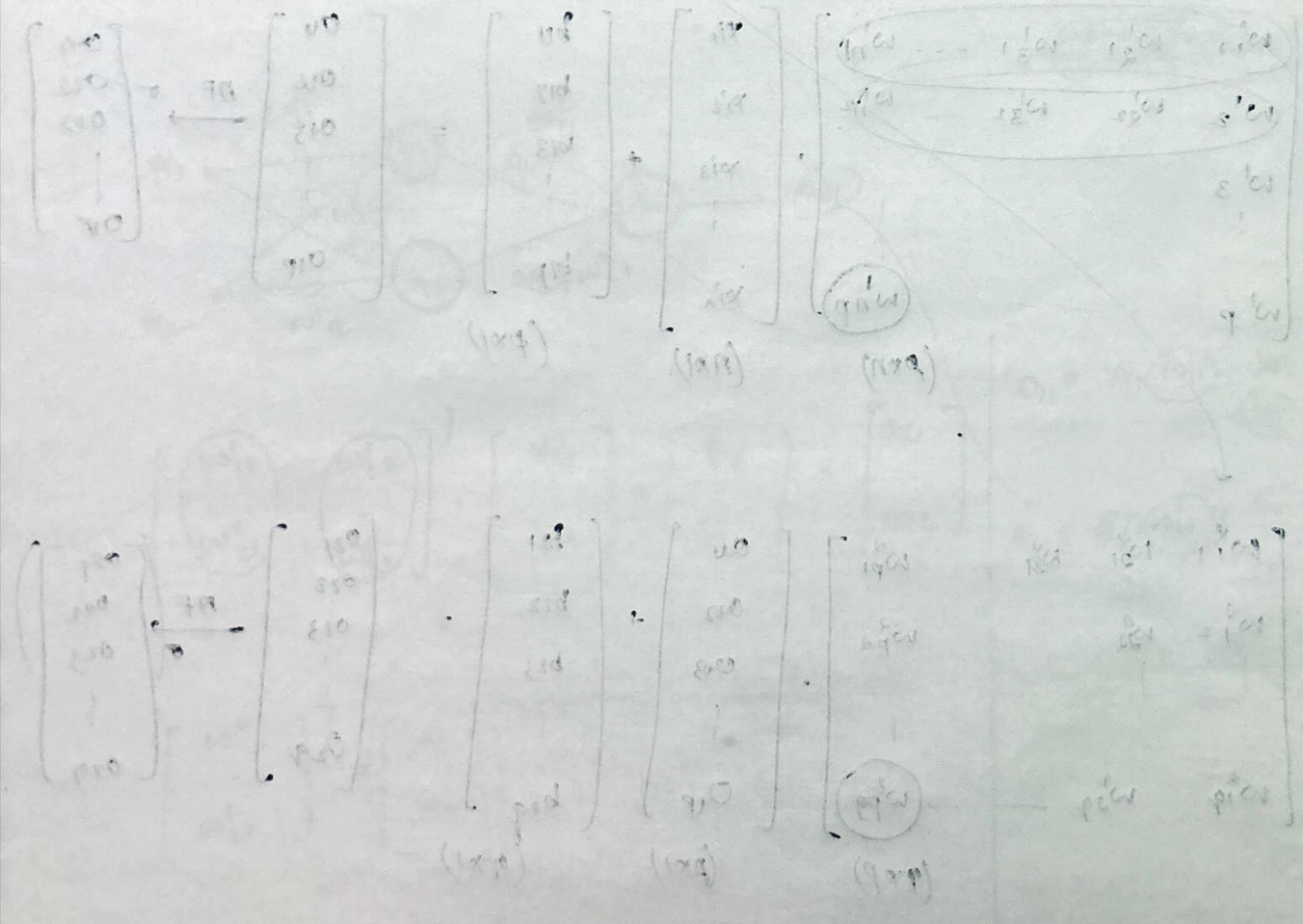
you also normalize the inputs of hidden layer which were output of other hidden layer.

model.add(BatchNormalization())

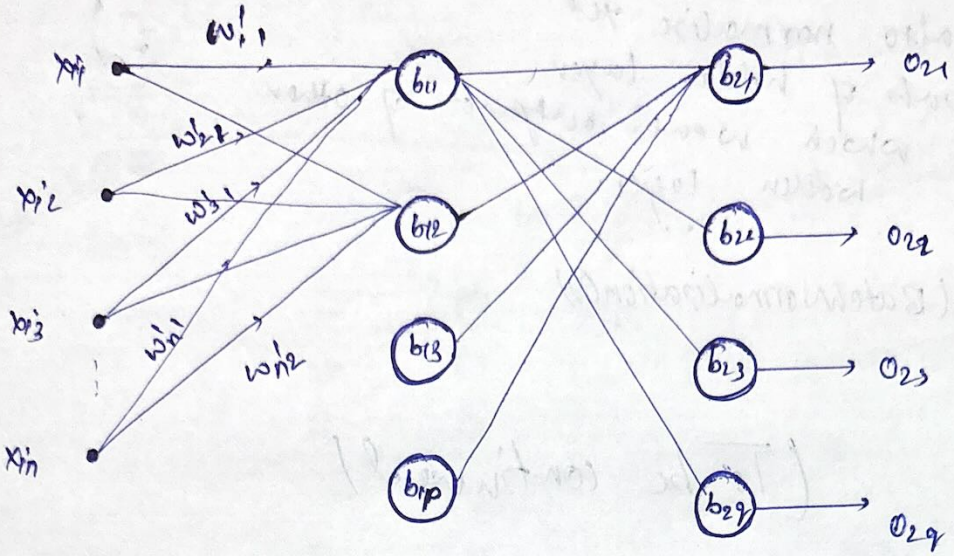
Lect 32

optimizers

(To be continued)



Forward propagation (FAINI project)



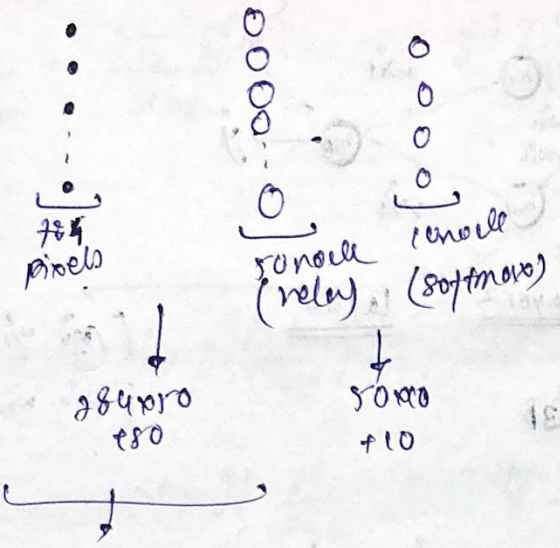
n = no. of nodes
Layer #0

p = no. of nodes
AFn = RELU
Layer # 1

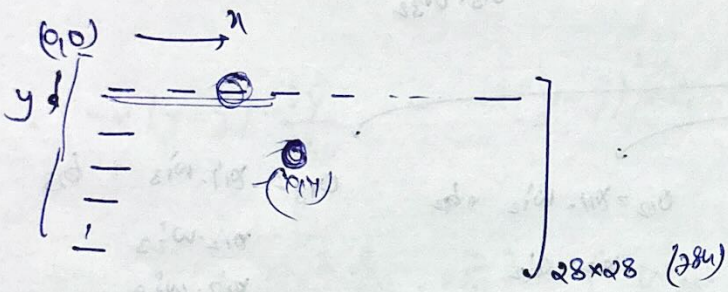
q = no. of nodes.
AFn = softmax
Layer # 2

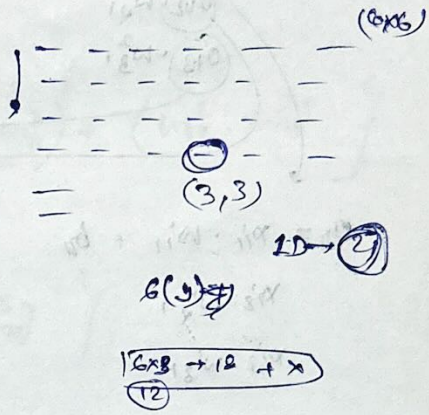
$$\begin{bmatrix} w^1_{11} & w^1_{21} & w^1_{31} & \cdots & w^1_{n1} \\ w^1_{12} & w^1_{22} & w^1_{32} & \cdots & w^1_{n2} \\ w^1_{13} & & & & \\ \vdots & & & & \\ w^1_{1p} & \cdots & & & w^1_{np} \end{bmatrix}_{(p \times n)} \cdot \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ \vdots \\ x_{in} \end{bmatrix}_{(n \times 1)} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \\ \vdots \\ b_{1p} \end{bmatrix}_{(p \times 1)} = \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \\ \vdots \\ o_{1p} \end{bmatrix} \xrightarrow{AF} \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \\ \vdots \\ o_{1p} \end{bmatrix}$$

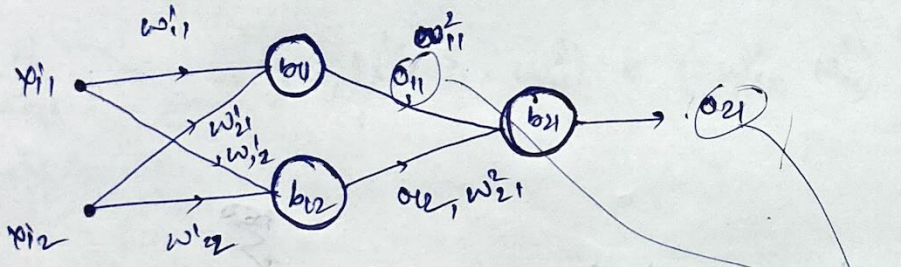
$$\begin{bmatrix} w^2_{11} & w^2_{21} & w^2_{31} & \cdots & w^2_{p1} \\ w^2_{12} & w^2_{22} & & & w^2_{p2} \\ \vdots & \vdots & & & \vdots \\ w^2_{1q} & w^2_{2q} & \cdots & & w^2_{pq} \end{bmatrix}_{(q \times p)} \cdot \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \\ \vdots \\ o_{1p} \end{bmatrix}_{(p \times 1)} + \begin{bmatrix} b_{21} \\ b_{22} \\ b_{23} \\ \vdots \\ b_{2q} \end{bmatrix}_{(q \times 1)} = \begin{bmatrix} o_{21} \\ o_{22} \\ o_{23} \\ \vdots \\ o_{2q} \end{bmatrix} \xrightarrow{AF} \begin{bmatrix} o_{21} \\ o_{22} \\ o_{23} \\ \vdots \\ o_{2q} \end{bmatrix}$$



$(50\times784)\,(784\times1) \rightarrow (50\times1) + \underbrace{(50\times1)}_{\text{Bias}}$





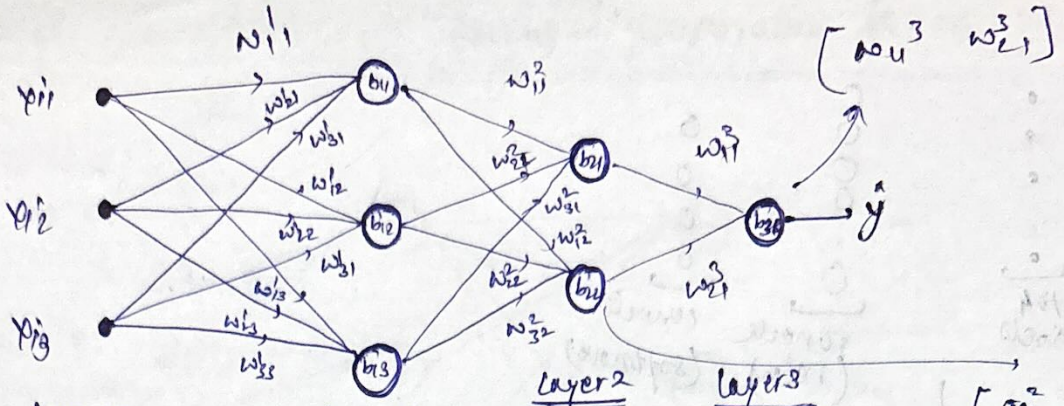


$$\begin{bmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{bmatrix}^T \cdot \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} = \begin{bmatrix} o_{11} \\ o_{12} \end{bmatrix}$$

$$\begin{bmatrix} w^2_{11} \\ w^2_{21} \end{bmatrix}^T \cdot \left(\begin{bmatrix} o_{11} \\ o_{12} \end{bmatrix} \right) + \begin{bmatrix} b_{21} \end{bmatrix} = \begin{bmatrix} o_{21} \end{bmatrix}$$

$o_{11} = x_{i1} \cdot w^1_{11} + x_{i2} \cdot w^1_{21} + b_{11}$

$o_{21} = o_{11} \cdot w^2_{11} + o_{12} \cdot w^2_{21} + b_{21}$



$$\hat{y} = o_{21} w_{11}^3 + o_{22} w_{21}^3 + b_{31}$$

$$o_{21} = o_{11} \cdot w_{12}^2 + b_{21}$$
$$o_{12} \cdot w_{21}^2$$
$$o_{13} \cdot w_{31}^2$$

$$o_{22} = o_{11} \cdot w_{12}^2 + b_{22}$$
$$o_{12} \cdot w_{22}^2$$
$$o_{13} \cdot w_{32}^2$$

$$o_{11} = x_{i1} \cdot w_{11}^1 + b_{11}$$
$$x_{i2} \cdot w_{21}^1$$
$$x_{i3} \cdot w_{31}^1$$

$$o_{12} = x_{i1} \cdot w_{12}^1 + b_{12}$$
$$x_{i2} \cdot w_{22}^1$$
$$x_{i3} \cdot w_{32}^1$$

$$o_{13} = x_{i1} \cdot w_{13}^1 + b_{13}$$
$$x_{i2} \cdot w_{23}^1$$
$$x_{i3} \cdot w_{33}^1$$

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

Layer 3

$$\frac{\partial L}{\partial w_{11}^3} = \frac{\partial (y-\hat{y})^2}{\partial w_{11}^3} = -2(y-\hat{y}) \frac{\partial \hat{y}}{\partial w_{11}^3} = -2(y-\hat{y}) \cdot o_{21}$$

$$\frac{\partial L}{\partial w_{21}^3} \longrightarrow -2(y-\hat{y}) \cdot o_{22}$$

Layer 2 (weights = 6, nodes = 2)

$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial (y-\hat{y})^2}{\partial w_{11}^2} = -2(y-\hat{y}) \frac{\partial \hat{y}}{\partial w_{11}^2} = -2(y-\hat{y}) \cdot o_{11} \cdot w_{11}^3$$

$$\frac{\partial L}{\partial w_{22}^2} = -2(y-\hat{y}) o_{12} \cdot w_{21}^3$$

Layer 1

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial (y-\hat{y})^2}{\partial w_{11}^1} = -2(y-\hat{y}) \frac{\partial \hat{y}}{\partial w_{11}^1} = -2(y-\hat{y}) \cdot \{ (w_{11}^3 \cdot w_{11}^2 \cdot x_{i1}) + (w_{21}^3 \cdot w_{12}^2 \cdot x_{i1}) \}$$

$$\frac{\partial L}{\partial w_{12}^1} = -2(y-\hat{y}) \cdot \{ (w_{11}^3 \cdot w_{31}^2 \cdot x_{i2}) + (w_{21}^3 \cdot w_{32}^2 \cdot x_{i2}) \}$$

layer 3

$$\frac{\partial L}{\partial b_{31}} = \frac{\partial (y-\hat{y})^2}{\partial b_{31}} = -2(y-\hat{y})\frac{\partial \hat{y}}{\partial b_{31}} = -2(y_1-\hat{y}).1$$

layer 2

$$\frac{\partial L}{\partial b_{21}} = -2(y-\hat{y})\left(\frac{\partial \hat{y}}{\partial b_{21}}\right) = -2(y-\hat{y}).w_{11}^3$$

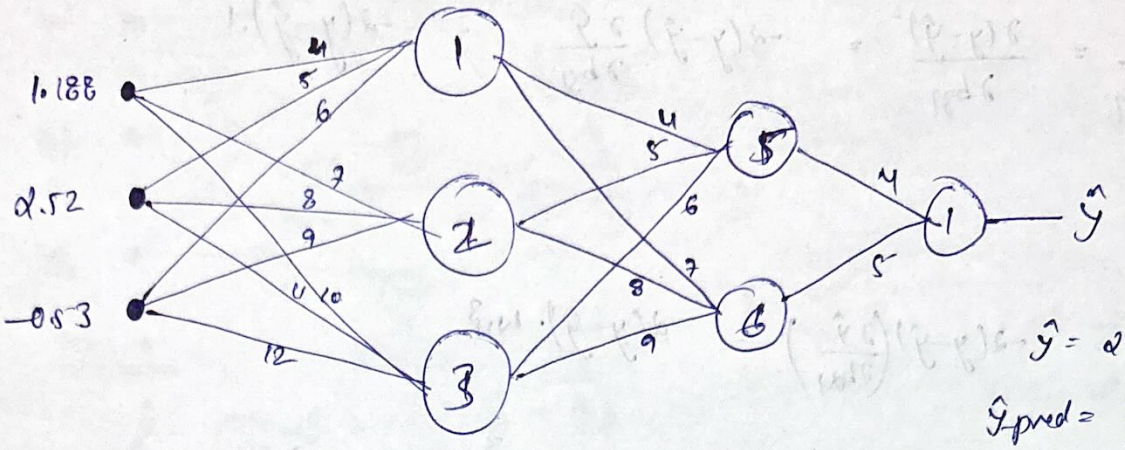
$$\frac{\partial L}{\partial b_{22}} = -2(y-\hat{y})\frac{\partial \hat{y}}{\partial b_{22}} = -2(y-\hat{y})\,w_{21}^3$$

layer 1

$$\frac{\partial L}{\partial b_{11}} = -2(y-\hat{y})\cdot\frac{\partial \hat{y}}{\partial b_{11}} = -2(y-\hat{y})\left\{(w_{11}^2 \cdot w_{11}^3) + (w_{12}^2 \cdot w_{21}^3)\right\}$$

$$\frac{\partial L}{\partial b_{12}} = -2(y-\hat{y})\left\{(w_{21}^2 \cdot w_{11}^3) + (w_{22}^2 \cdot w_{21}^3)\right\}$$

$$\frac{\partial L}{\partial b_{13}} = -2(y-\hat{y})\left\{(w_{31}^2 \cdot w_{11}^3) + (w_{32}^2\, w_{21}^3)\right\}$$



$\hat{y} = 208.32$

$\hat{y}_{pred} = 4626.42$

loss =

$O_{11} = 15.19$

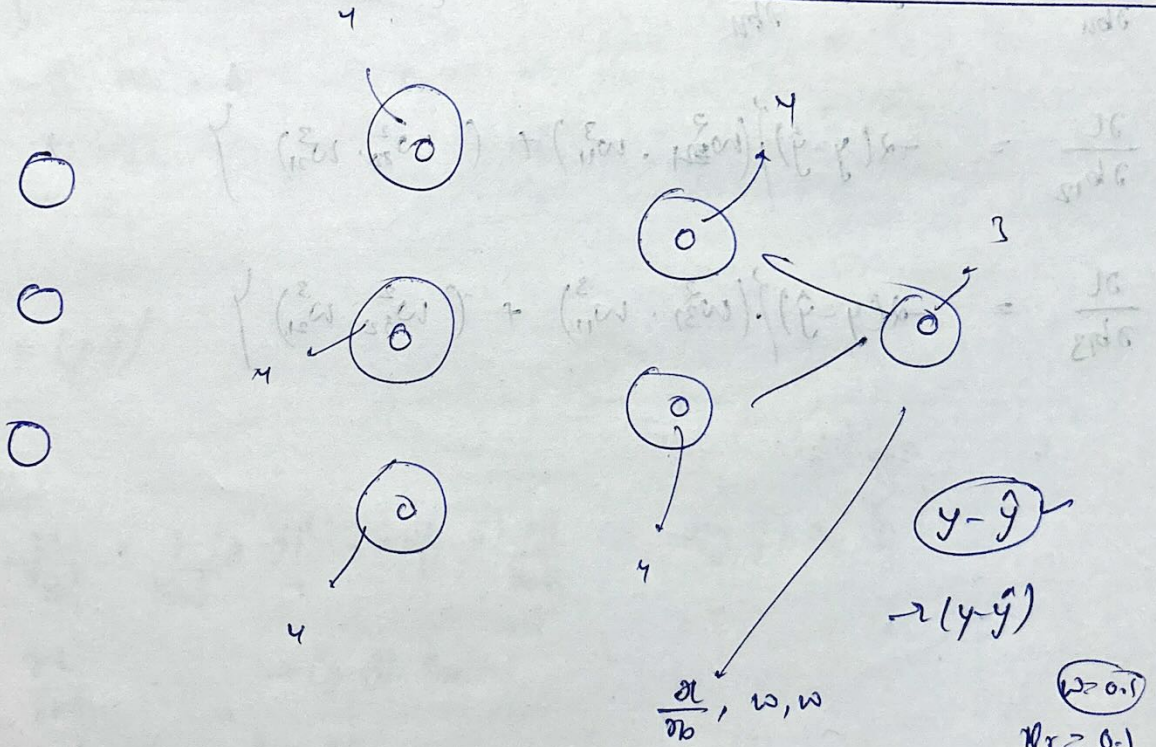
$O_{12} = 24.20$

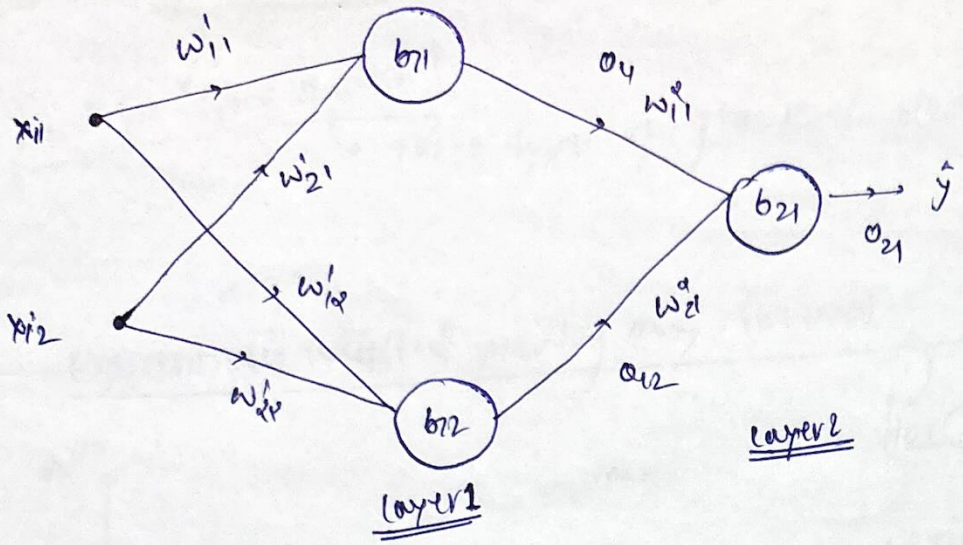
$O_{13} = 39.24$

$O_{21} = 390.62$ $O_{23} = 4628.48$

$O_{22} = 613$

(23)





$o_{11} = x_{i1} \cdot w'_{11} + x_{i2} \cdot w'_{21} + b_{11}$

$o_{21} = o_{11} \cdot w^2_{11} + o_{12} \cdot w^2_{21} + b_{21}$

$o_{12} = x_{i1} \cdot w'_{12} + x_{i2} \cdot w'_{22} + b_{12}$

$$\begin{bmatrix} 0.1 & 0.2 \\ w'_{11} & w'_{21} \\ w'_{12} & w'_{22} \\ 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} x_{i1}=1 \\ x_{i2}=2 \end{bmatrix} + \begin{bmatrix} 0.25 \\ b_{11} \\ b_{12}=0.35 \end{bmatrix} = \begin{bmatrix} o_{11} \\ o_{12} \end{bmatrix} \rightarrow \sigma\left(\begin{bmatrix} o_{11} \\ o_{12} \end{bmatrix}\right) = \begin{bmatrix} 0.82 \\ 1.18 \end{bmatrix}$$

$$\begin{bmatrix} w^2_{11} & w^2_{21} \\ 0.5 & 0.6 \end{bmatrix} \begin{bmatrix} o_{11} \\ o_{12} \end{bmatrix} + \begin{bmatrix} b_{21} \\ 0.09 \end{bmatrix} = [o_{21}] \Rightarrow \sigma([o_{21}]) \quad [1.083]$$

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

$$(y-\hat{y}) = (10 - 1.083) = 8.917$$

$$-2(y-\hat{y}) = -17.834$$

layer2

$$\frac{\partial L}{\partial w^2_{11}} = -2(y-\hat{y}) \cdot \left(\frac{\partial \hat{y}}{\partial w^2_{11}}\right) = -2(y-\hat{y}) \cdot o_{11} = -10.66$$

$$\frac{\partial L}{\partial w^2_{21}} = -2(y-\hat{y}) \cdot o_{12} = -21.04$$

$$\frac{\partial L}{\partial b_{21}} = -2(y-\hat{y}) \cdot 1 = -17.834$$

layer 1

$$-8.917 = \frac{\partial L}{\partial w'_{11}} = -2(y-\hat{y}) \cdot x_{i1} \cdot (w^2_{11})$$

$$-17.834 = \frac{\partial L}{\partial w'_{21}} = -2(y-\hat{y}) \cdot x_{i2} \cdot w^2_{11}$$

$$-8.917 = \frac{\partial L}{\partial b_{11}} = -2(y-\hat{y}) \cdot w^2_{11}$$

$$\frac{\partial L}{\partial w'_{12}} = -2(y-\hat{y}) \cdot x_{i1} \cdot w^2_{21} = -10.7$$

$$\frac{\partial L}{\partial w'_{22}} = -2(y-\hat{y}) \cdot x_{i2} \cdot w^2_{21} = -21.4$$

$$\frac{\partial L}{\partial b_{12}} = -2(y-\hat{y}) \cdot w^2_{21} = -10.7$$

Both sigmoid

data → hidden layer (1) → output layer

$x_i = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}_{(n \times 1)}$

units 64

units (1)

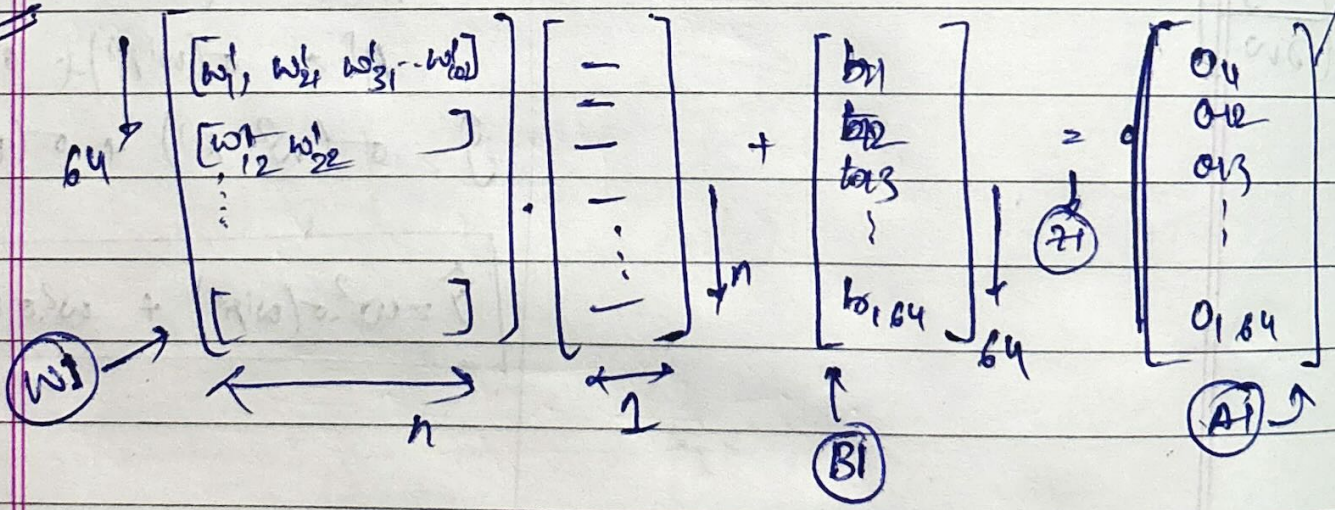
$z = WX + B$

$(64 \times 1) \quad (64 \times n)(n \times 1) \quad (64 \times 1)$

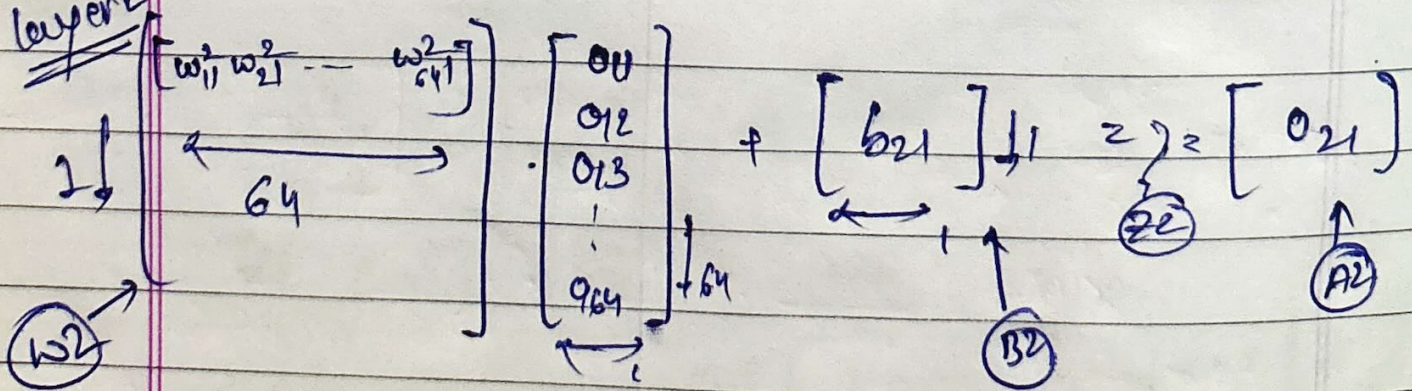
$z = WX + B$

$(1 \times 64) \quad (64 \times 1) \quad (1 \times 1)$

Layer 1



Layer 2



$z_1 = w_1 \cdot x_1 + b_1$

$a_1 = \sigma(z_1)$

$z_2 = w_2 \cdot a_1 + b_2$

$a_2 = \sigma(z_2)$

$\frac{\partial l}{\partial w_1} = \frac{1}{m} \cdot x (\hat{y} - y)^T$ | $\frac{\partial l}{\partial b} = \frac{1}{m} \sum (\hat{y}_i - y_i)$

$loss = -(y \cdot \log \hat{y} + (1-y) \cdot \log(1-\hat{y}))$

$\sigma(x) = \frac{1}{1+e^{-x}}$, $\sigma'(x) = \frac{e^x}{(1+e^x)^2}$

$\frac{\partial l}{\partial w_2} = \frac{\partial l}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2}$

$z_1 = w_1 x + b_1$

$a_1 = \sigma(z_1) =$

$w_2 = w_2 - \eta \frac{\partial l}{\partial w_2}$

$\begin{bmatrix} z_1 = w_1 \cdot x + b_1 \\ a_1 = Relu(z_1) \end{bmatrix}$ $\frac{\partial z_2}{\partial w_2}$

$\hat{y} = \begin{bmatrix} z_2 = w_2 \cdot a_1 + b_2 \\ a_2 = sigmoid(z_2) \end{bmatrix}$

$l = -y \cdot \log \hat{y} - (1-y) \log(1-\hat{y})$

$\frac{\partial l}{\partial \hat{y}} = \frac{-y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})}$

$\frac{\partial \hat{y}}{\partial z_2} = \sigma(z_2)(1-\sigma(z_2))$

$= \hat{y}(1-\hat{y})$

$\frac{\partial l}{\partial z_2} = \frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2}$

$\frac{\partial l}{\partial z_2} = -y(1-\hat{y}) + \hat{y}(1-y)$

$z_1 = w_1 \cdot x + b_1 \Rightarrow A_1 = \text{relu}(z_1)$

$z_2 = w_2 \cdot A_1 + b_2 \rightarrow A_2 = \text{sigmoid}(z_2) = \hat{y}$

$$\frac{\partial L}{\partial w_1} = \left[\frac{-y}{\hat{y}} + \frac{(1-y)}{1-\hat{y}}\right] \cdot \frac{\partial \hat{y}}{\partial w_1}$$

$$\frac{\partial z_2}{\partial w_1} = w_2 \cdot \frac{\partial A_1}{\partial w_1}$$

$$\frac{\partial(\sigma(z_2))}{\partial w_1} = \sigma(z_2)(1-\sigma(z_2)) \cdot \frac{\partial z_2}{\partial w_1}$$

$$w_2 \cdot \frac{\partial A_1}{\partial w_1} \rightarrow \begin{cases} 1, & z_1 > 0 \\ 0, & z_1 \leq 0 \end{cases}$$

Layer 2

$$\frac{\partial L}{\partial w_2}, \quad \frac{\partial L}{\partial b_2} \qquad \hat{y}(1-\hat{y})$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2}$$

$$\left(\frac{-y}{\hat{y}}\right) + \left(\frac{1-y}{1-\hat{y}}\right)$$

$\frac{\partial z_2}{\partial w_2} \rightarrow A_1$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} \rightarrow 1.$$

Layer 1

$$\frac{\partial L}{\partial w_1}, \quad \frac{\partial L}{\partial b_1}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_1} \cdot \frac{\partial A_1}{\partial z_1} \cdot (x)$$

$$\frac{\partial A_1}{\partial z_1} \rightarrow \begin{cases} 1, & z_1 > 0 \\ 0, & z_1 \leq 0 \end{cases}$$

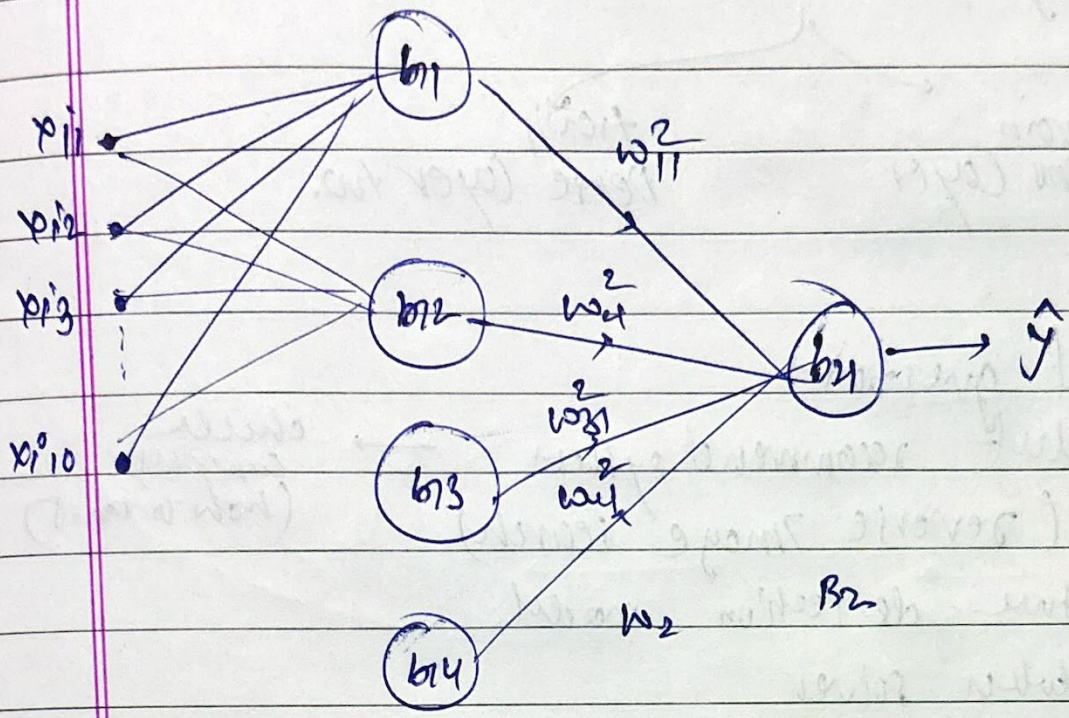
$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_1} \rightarrow \begin{cases} w_2, & z_1 > 0 \\ 0, & z_1 \leq 0 \end{cases}$$

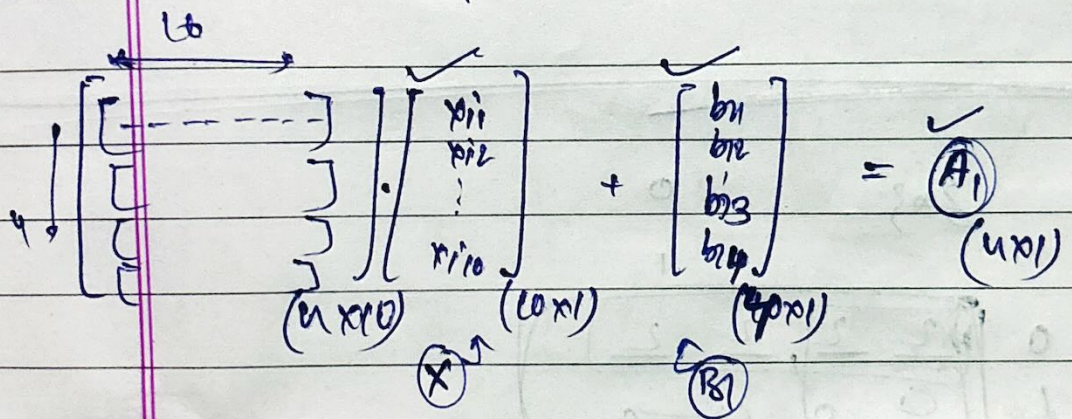
$$\frac{\partial z_2}{\partial w_1} = \left(w_2^T\right)\left(\frac{\partial A_1}{\partial w_1}\right) \rightarrow (4\times 1)$$

$(1\times 4)^T \rightarrow (4\times 1)$

No.

Date : / /





$$[[\text{----}]] \leftarrow W_2 \cdot A_1 + B_2 = A_2$$

$(1\times 4)\ (4\times 1)\quad (1\times 1)\quad (1\times 1)$

$$\frac{\partial z_2}{\partial w_1} = w_2 \frac{\partial A_1}{\partial w_1} \times \frac{\partial z_1}{\partial w_1} \rightarrow x_{i1}$$

$$\begin{cases} 1, & z_1 > 0 \\ 0, & z_1 < 0 \end{cases}$$

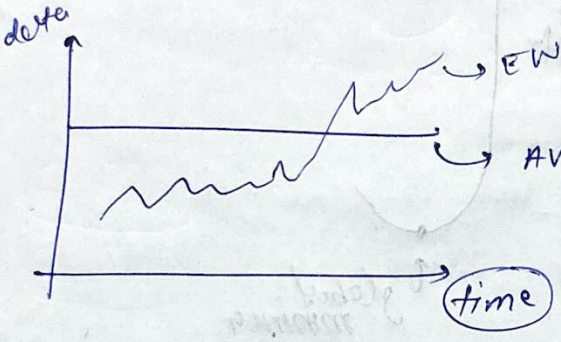
lect 32 # optimizers → methods which are used to find weights and biases.

what we have studied? → three types of gradient descent.

lect 33

Exponentially weighted moving Avg (EWMA)

for time series data



weitage
Data t_2 > Data t_1
where $t_2 > t_1$

$$V_t = \beta . V_{t-1} + (1-\beta)\theta_t$$

V_t → EWMA at time t

V_{t-1} → EWMA at $(t-1)$

θ_t → data at time t

Date	θ (temp)
1	28
2	13
3	17
4	31
5	43

$\beta \in (0,1)$

$\frac{1}{1-\beta}$ → EWMA is Avg of last this much time

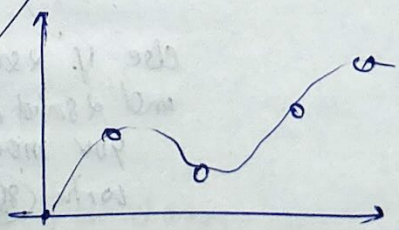
$V_0 = 0$ or $V_0 = \theta_0$ more accurate result

$V_1 = (0.9).V_0 + (0.1).13$

$V_1 = 1.3$

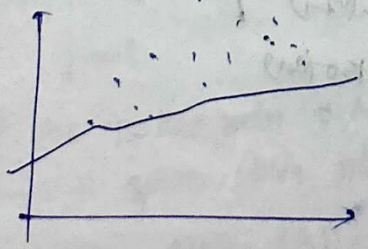
$V_2 = (0.9)(1.3) + (0.1)(17)$

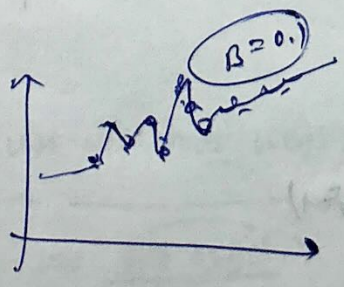
V_3, V_4, V_5



$\beta \uparrow \Rightarrow$ more weitage to prev points

$\beta \downarrow \Rightarrow$ more ——— newer points.





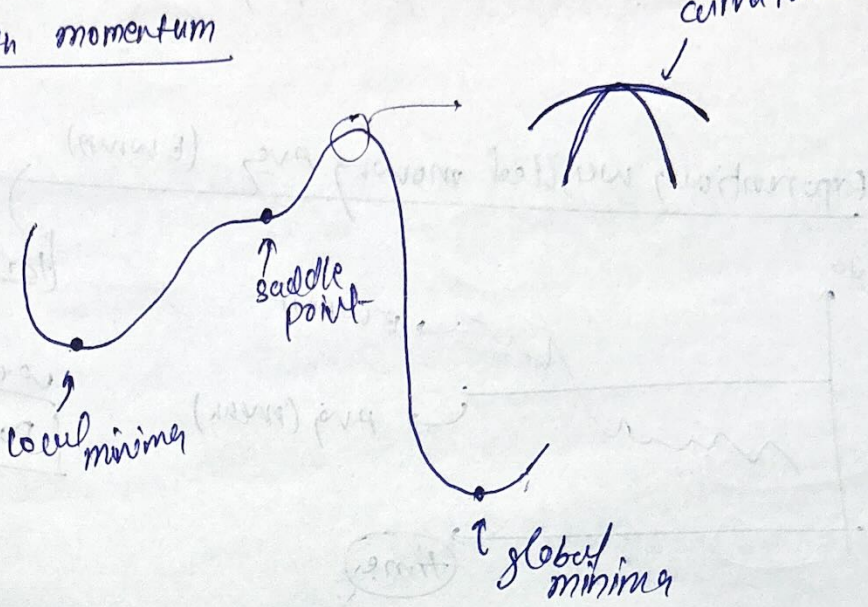
In pandas

df['column_name'].ewm(alpa = 0.1).means()

Part 34]

Sto. gradient descent

SGD with momentum



problems

1. local minima
2. saddle point
3. high curvature

→ makes training slow.

momentum technique is good in all three cases.

Ball

speed

→ — — ⇒ then you increase your speed

If 4 people tell you, you are moving right

else if 2 said right and 2 said left, you move right with slow speed.

$$w_{t+1} = w_t - \eta \cdot w_t$$

v → velocity

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

$$v_t = \beta \cdot (v_{t-1}) + \eta \cdot (\Delta w_t)$$

$\beta \in (0,1)$

In practice (keras)

lect 35] # Nesterov Accelerated Gradient (NAG)

$W_{new} = W_{old} - (\eta \frac{\partial L}{\partial W} + \beta V_t)$

$\eta \frac{\partial L}{\partial W}$ → derivative βV_t → Acceleration

Here $W_{new} = W_{old} - \beta V_t$

βV_t → At start of $\eta \frac{dL}{dW}$

Here your magnitude is less for moving.

→ But NAG can make you settle in local minima. forever.

In keras

tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0, nesterov=0, name="SGD", **kwargs)

SGD	SGD with momentum	Nesterov.
0	momentum = β (0.1)	= β (0.1)
False	nesterov = False	= True

lect 36] # AdaGrad

Adaptive Gradient → learning rate keep on changing.

Apply when most of the features values are zero. (sparse data)

how?

Jis column mai bahut zeros ho (sparse) uss dim mai weight/bias kam update hoga and ——— sparse data nhi hai ——— Bahut update hoga.

thats why NAG, SGD are slow. use AdaGrad

[lect 37] # Root mean square prop (RMS prop).

↓

Improvement over Adagrad.

First you move very quickly towards non-sparse data, then towards sparse data,

Because of this what happens is, if there were some movement left in non-sparse dirn it cannot converge as the lr has become soo low.

to solve this RMS prop.

* Best optimizing technique. (No disadvantage)

[lect 38] # Adam optimizer

→ sri hari

[lect 39] # keras Tuner (Hyperparameter)

```
import keras_tuner as kt
def build_model(hp):
    model = Sequential()
    model.add(Dense(32, relu, ——)                    loss='——'?'
    model.compile(optimizers = hp.Choice('optimizers', ['adam', 'sgd']))
    return model.
tuner = kt.RandomSearch(build_model, objective='val_accuracy',
                        max_trials=5)
tuner.search(xtrain, ytrain, epochs=5, validation_data=(xtest, ytest))
```