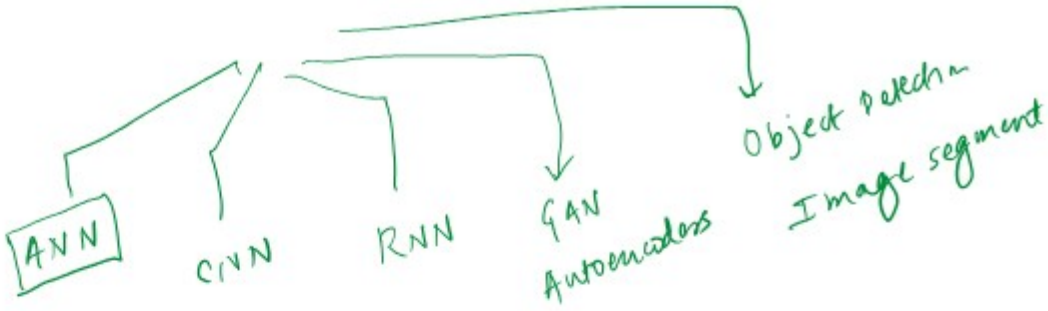
Curriculum

Tuesday, February 15, 2022 3:30 PM



Features

Tuesday, February 15, 2022 3:31 PM

1) Well Researched
2) Easy to consume
3) Well structured → 100 days
4) Tensorflow + Keras
5) Projects

Prerequisites

Tuesday, February 15, 2022 3:31 PM

1) Python →
2) Basics of ML
3) Linear Algebra → 3 Blue 1 Brown / Linear Alg → (5)

Extra Content

Tuesday, February 15, 2022 3:32 PM

1) Deep learning Roadmap
2) Deep learning Project ideas
3) Projects
4) ±nterview questions

↳ Next video
↳ This

→ 10th App & → ANN → CNN
↳ 3-4 months

What is Deep Learning?

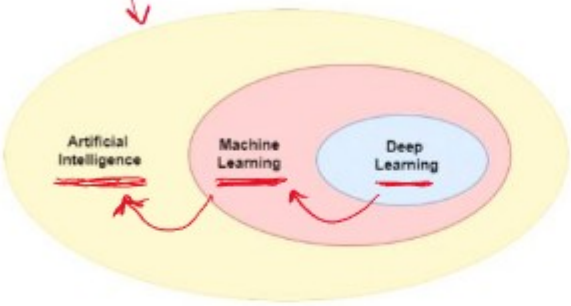
16 February 2022 06:32

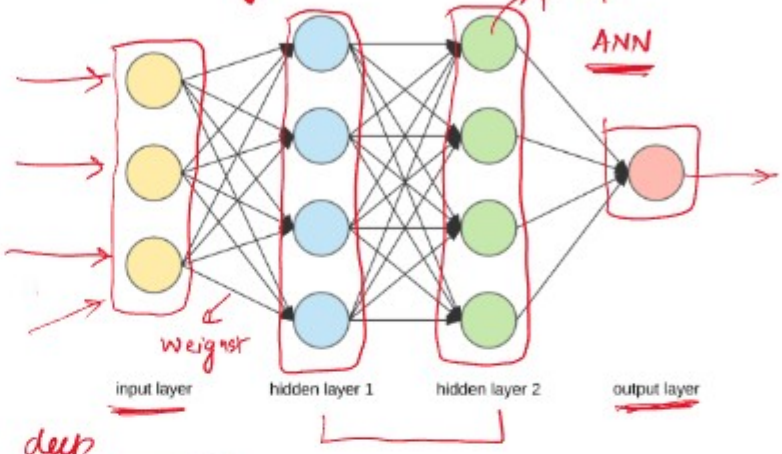
Deep Learning is a subfield of Artificial Intelligence and Machine Learning that is inspired by the structure of a human brain.
Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure called Neural Network.

learn from data

input | output

statistical





deep

ANN →
CNN → convolutional NN ✓ image
RNN → Recurrent NN → speech / text
GAN → generate (text) images

why (dl) getting so famous?

Go → AlphaGo → AI
5 → (4)

1) Applicability →
2) Performance → state-of-the-art → human experts

Applications → self driving, drugs → chemistry

1) Representation

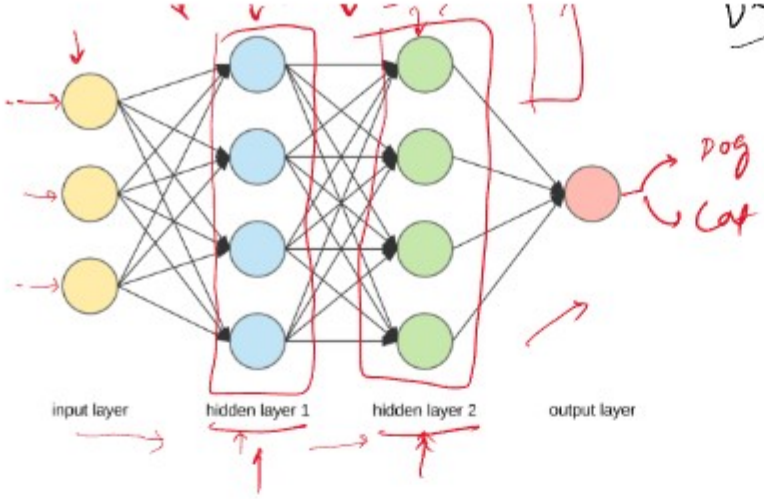
Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning.

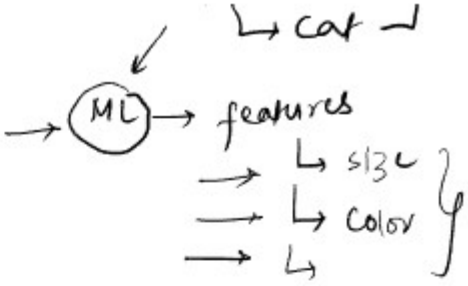
Deep Learning Algorithms uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

feature extraction engineering

features

Deep learning → Dog / cat classifier → Dog / cat


input layer
hidden layer 1
hidden layer 2
output layer
Dog
Cat


Cat
ML
features
size
Color

Deep Learning VS Machine Learning

16 February 2022 06:55

1. Data Dependency
2. Hardware Dependency
3. Training Time
4. Feature Selection
5. Interpretability

costly DL

CPU → run slow

Data ↑ hungry

Data ↑ performance ↑

matrix multiplication → GPU → powerful

Representation learning → features extract

complex

training time - high → weeks

ML → low minutes - hrs

less data

ML wins

Prediction

DL → fast

ML

KNN

DL → Algorithm → feature

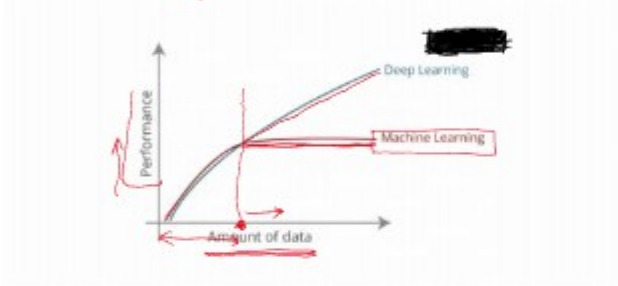
Resume based

→ predict → Y/N

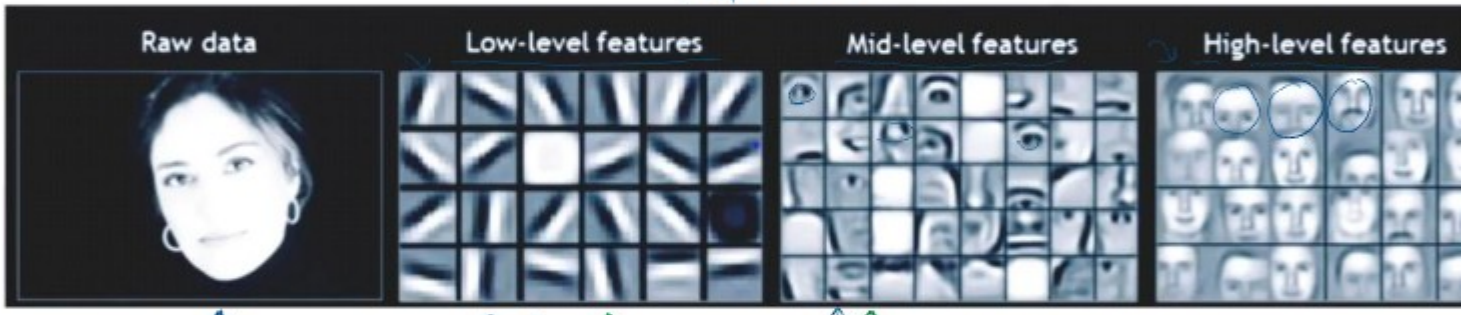
→ features

domain experts → 12th marks, 10th mars

teachers → No. of ach, No. of course, quality of college



Deep learning



feature → interpre

Deep learning

comment → ban

user

Dog

cat

Black box

logistic

cgpa | iq | placement

w_1 w_2

DT

flow chart

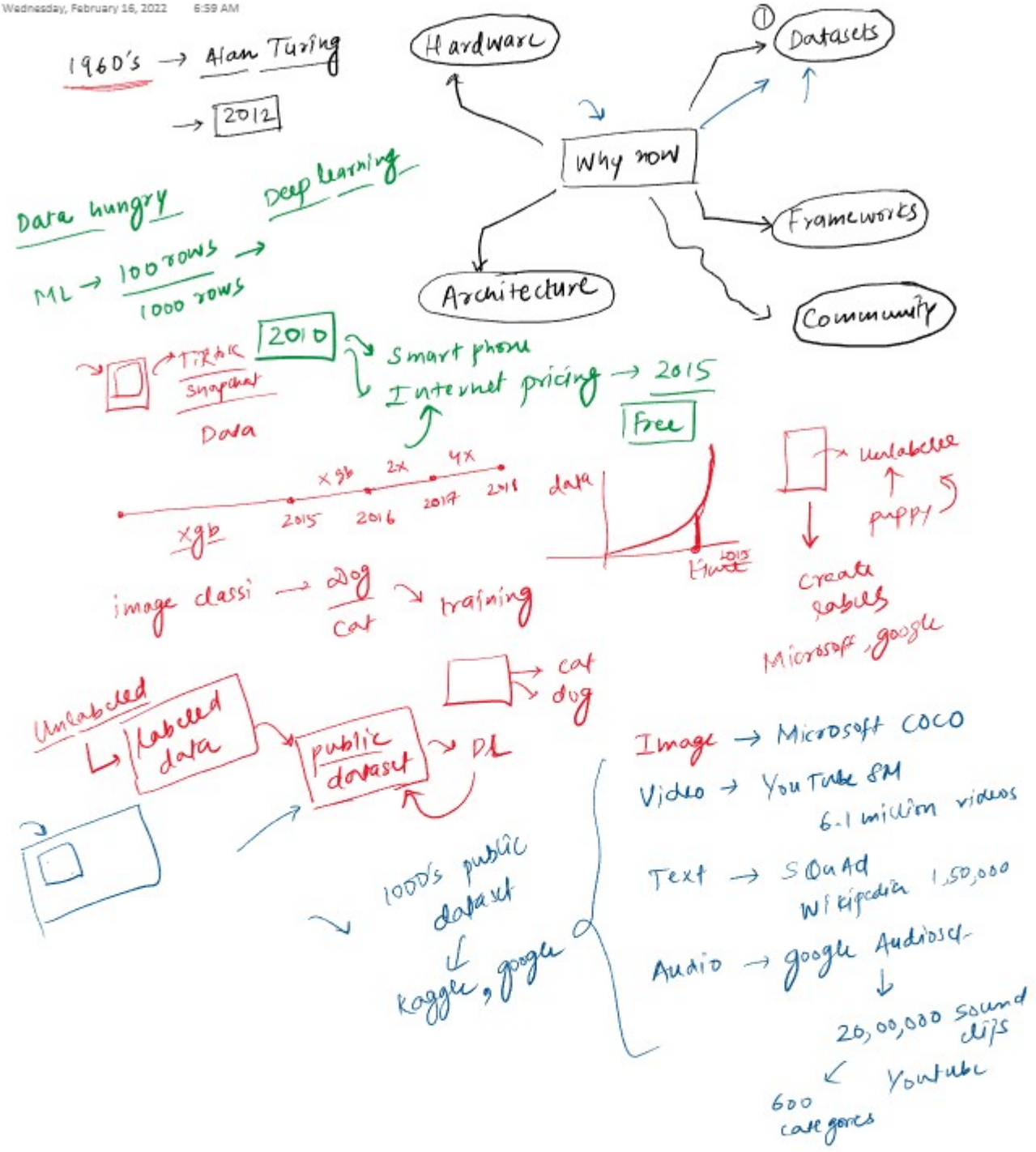
explainability

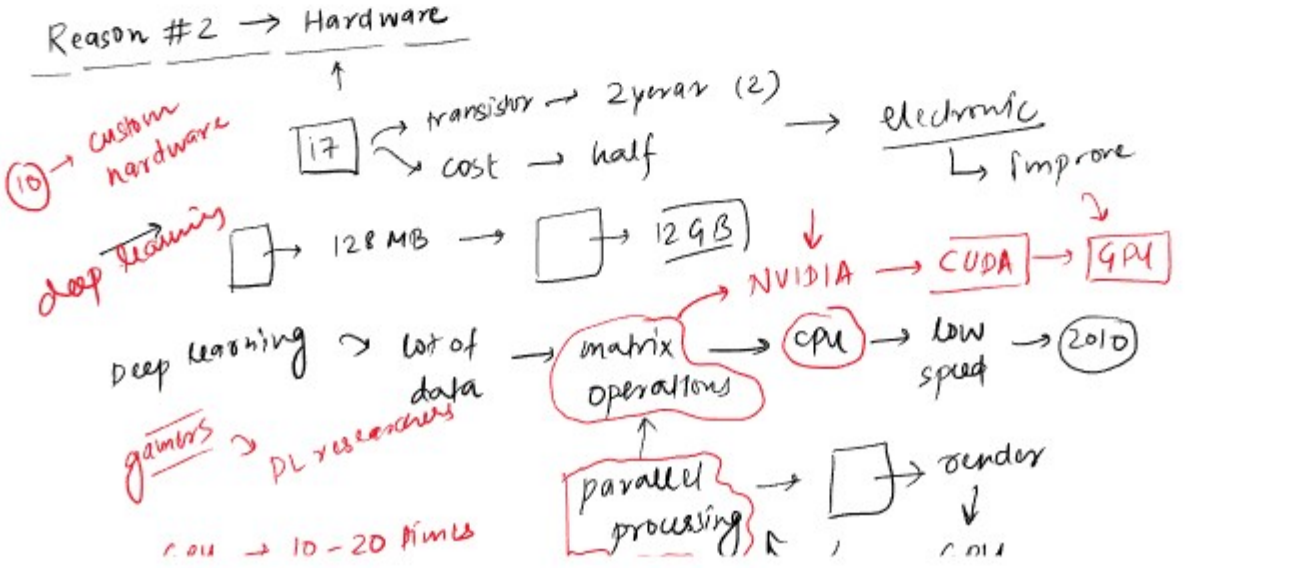
High ↑

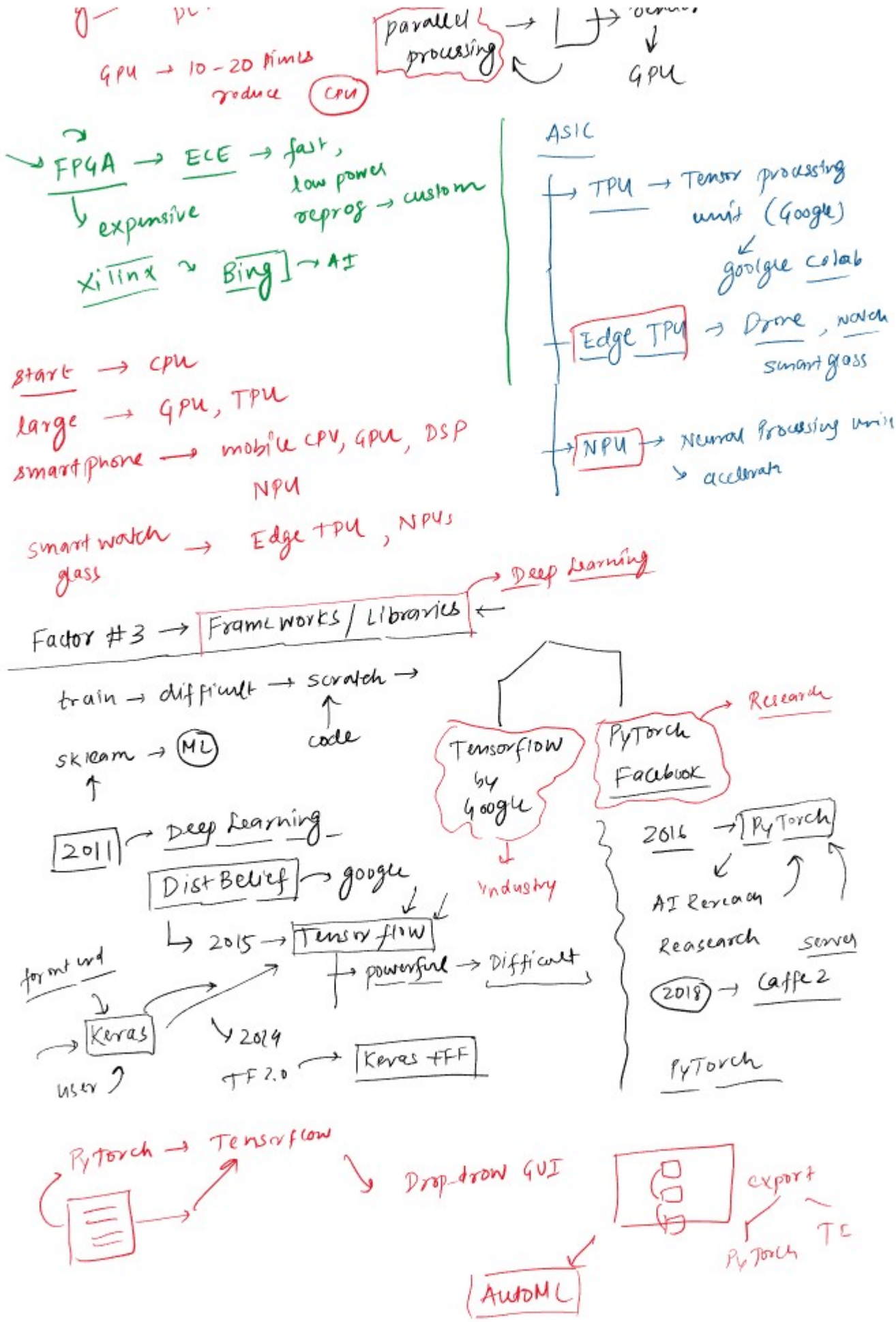
Machine learning →

Why now?

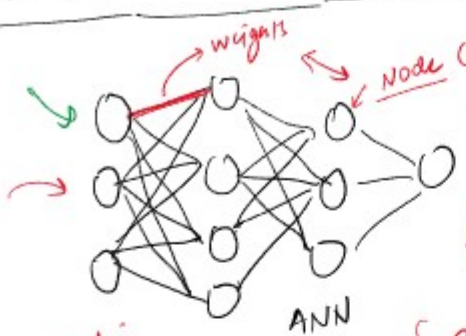
Wednesday, February 16, 2022 6:59 AM






parallel processing
GPU
GPU → 10 - 20 times reduce CPU
FPGA → ECE → fast, low power reprog → custom
expensive
Xilinx & Bing → AI
ASIC
TPU → Tensor processing unit (Google)
google colab
Edge TPU → Drone, watch smart glass
NPU → Neural Processing Unit → accelerate
start → CPU
large → GPU, TPU
smartphone → mobile CPU, GPU, DSP NPU
smart watch glass → Edge TPU, NPUs
Factor #3 → Frameworks / Libraries
Deep Learning
train → difficult → scratch → code
sklearn → ML
2011 → Deep Learning
DistBelief → google
2015 → Tensorflow
powerful → Difficult
front end
Keras
user
2019
TF 2.0 → Keras +FF
Tensorflow by Google
Industry
PyTorch Facebook
Research
2016 → PyTorch
AI Research
Research server
2018 → Caffe2
PyTorch
PyTorch → Tensorflow
Drop-down GUI
export
PyTorch
TF
AutoML
Factor #4 → Deep Learning Architectures

Factor #4 → Deep Learning Architectures



CNN RNN

DNN archi

Experiment
- time
- effort
- money

Transfer learning

State-of-the-art best accuracy

ANN → existing archite

NN → Archi
- fires down
- Dataset → train → ready model

image classification → ResNET
Text classifi → BERT
Image segment → UNet
image tran → PIX2PIX

Object dect → YOLO
Speech gen → WaveNET

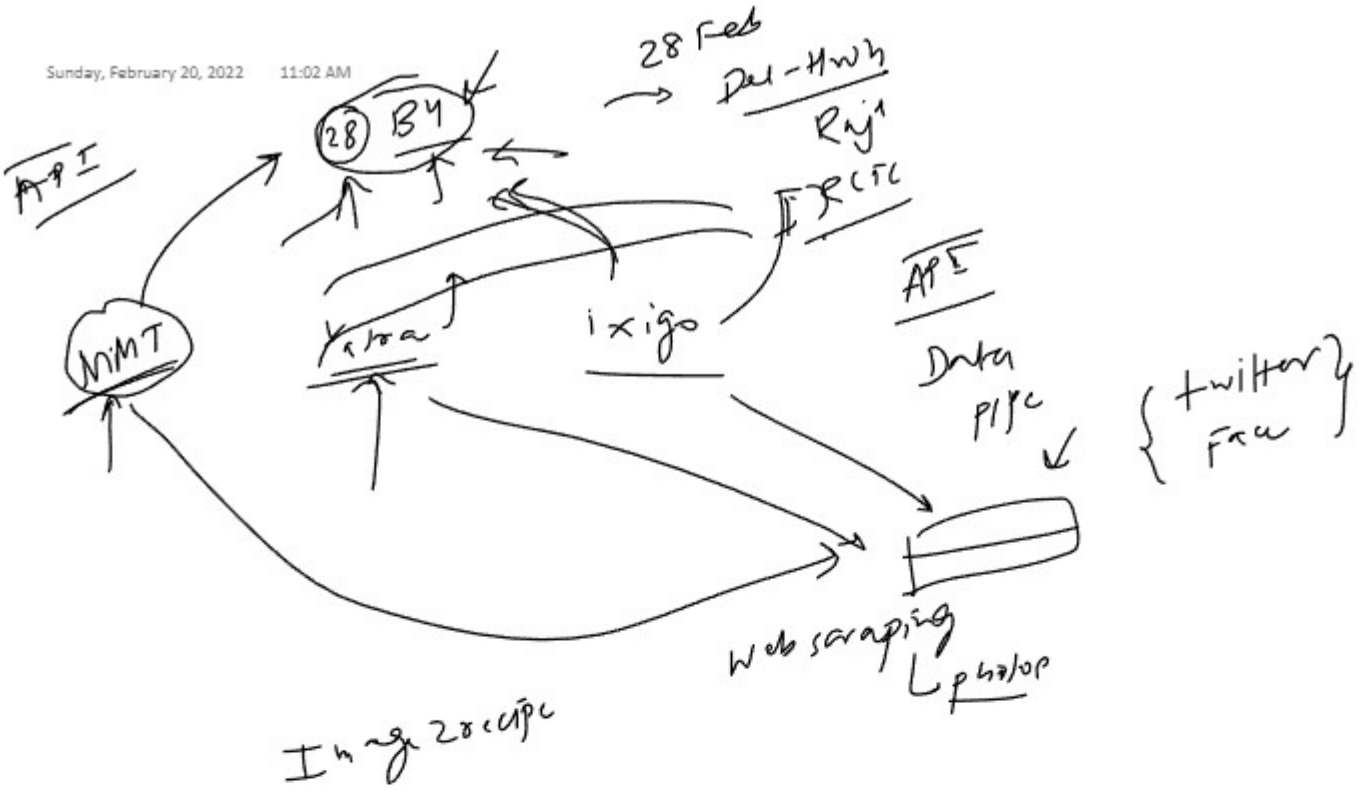
#5 → People / community

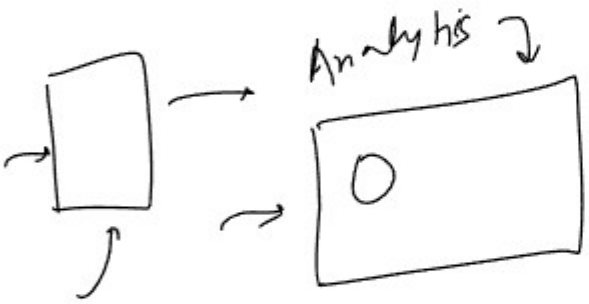
↳ Deep learning → 1960 → 2011 → PL →

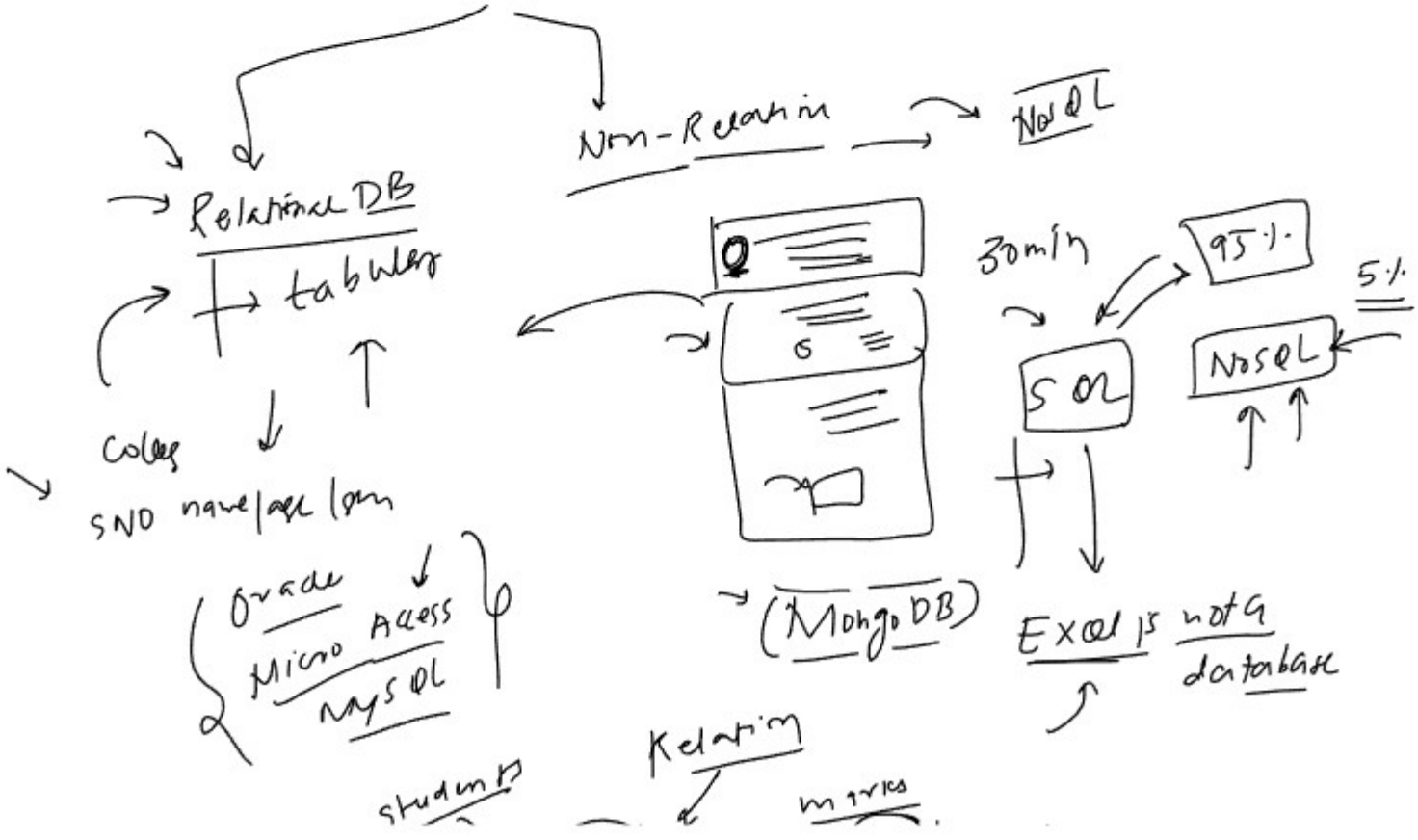
Research
↳ DL@Stu
↳ Teach → Student → Kaggle

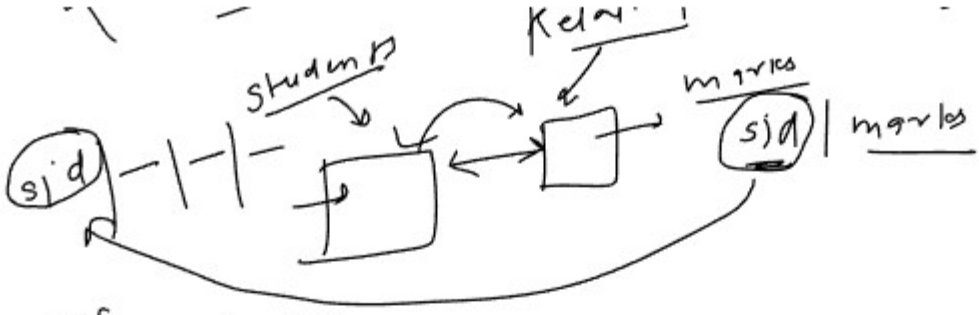
Code Example

Saturday, February 19, 2022 7:10 AM

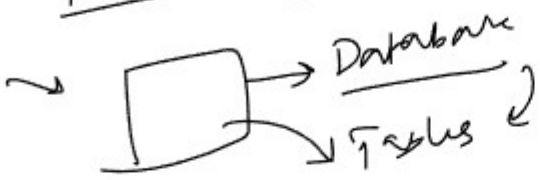

28 Feb
Del-Hwh
Raj
API
28
BY
IRCTC
API
MMT
ixigo
Data
Pipe
{ twitter
Fac }
Web scraping
Image recipe

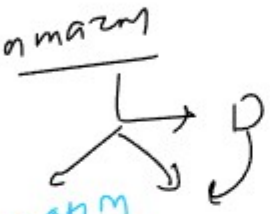

Analytics


Relational DB
tables
Non-Relation
NoSQL
30min
95%
5%
SQL
NoSQL
Colleges
SNO name/address
Oracle
Micro Access
MySQL
(MongoDB)
Excel is not a database
Relation
student
marks



DBMS Oracle





Normalization

student-marks

DB → storage

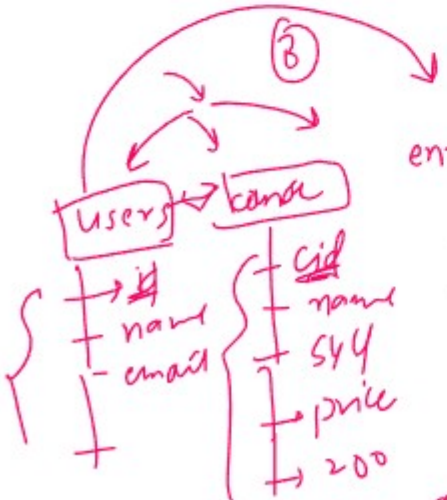
sid	name	gender	age	course	marks
1	satyam	M	21	Eng	24
1	Satyam	M	21	Maths	32

Denormalization

OLTP

OLAP

website

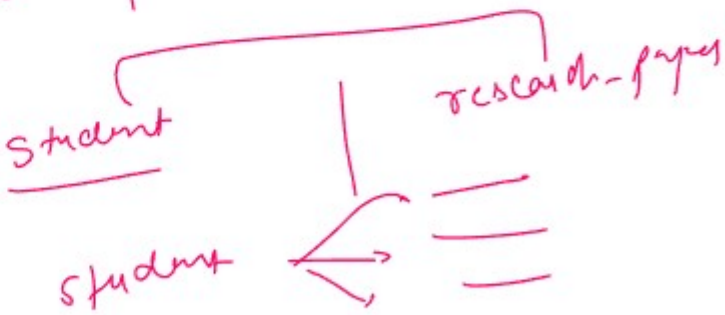


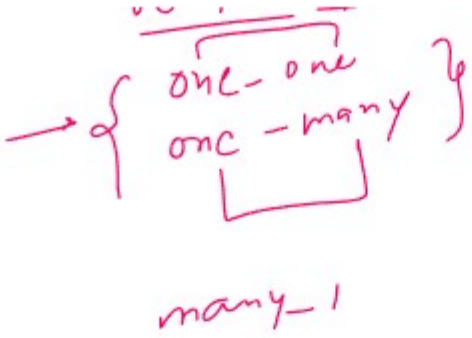
enrollment

id
cid
pay
date

relationship

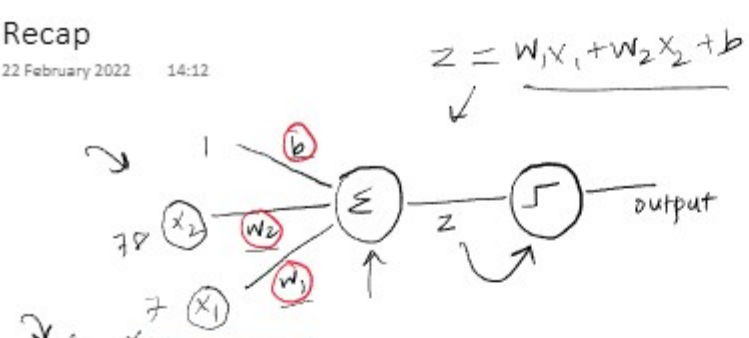
one-one

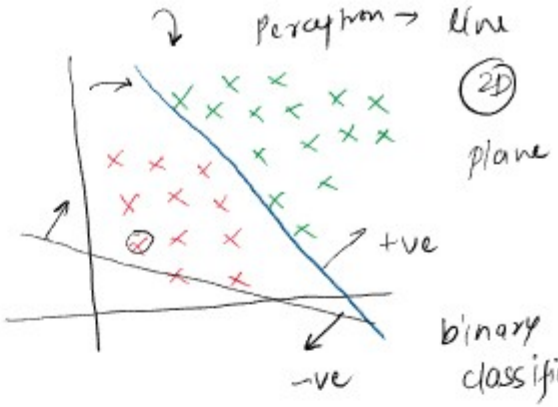



one-one
one-many
many-1

Recap

22 February 2022 14:12





x_1 cgpa	x_2 iq	placed
7	78	1
6	61	0
9	92	1

1) mathematical model → neuron

$z \geq 0 \rightarrow 1$

$z < 0 \rightarrow 0$

Prediction

→ Training

1000 random

Perceptron Rule

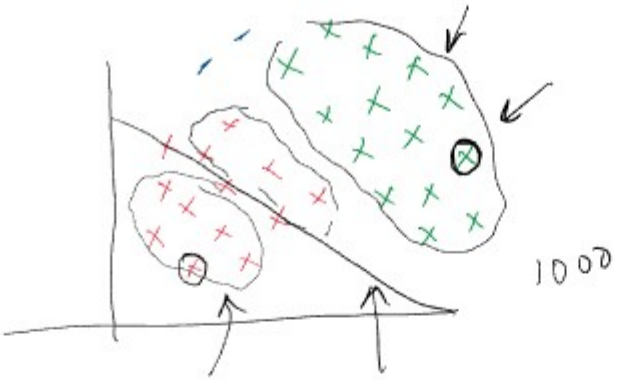
$Ax + By + c = 0$

$A \rightarrow w_1$

$B \rightarrow w_2$ $C \rightarrow b$

Problem with Perceptron Trick

22 February 2022 16:10



1000

line → w_1, w_2, b

randomly per

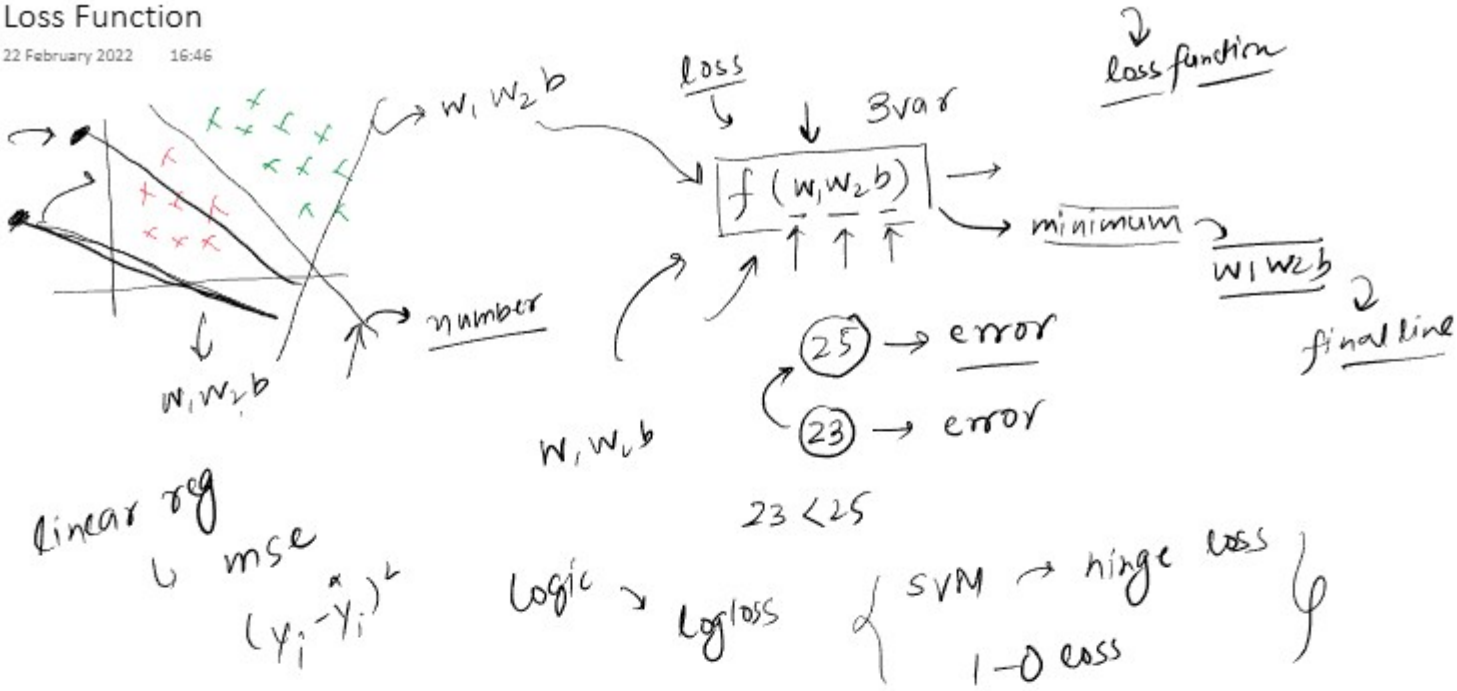
converge

quantify

w_1, w_2, b → loss functions

Loss Function

22 February 2022 16:46

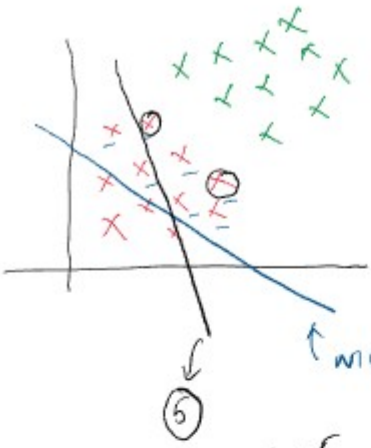


Perceptron Loss Function

22 February 2022 17:47

$f(w_1, w_2, b)$ → number → error

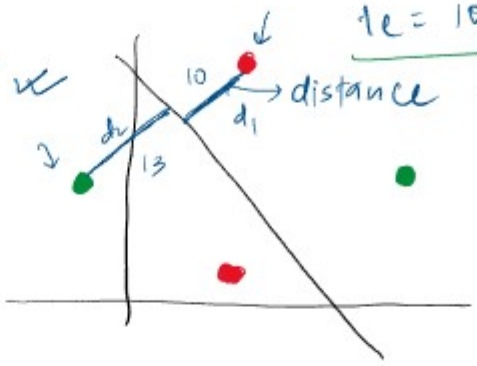
misclassified points



$w, w_2, b \rightarrow$ (7)

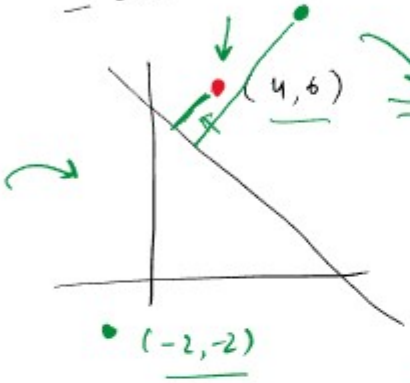
(5)

$|+|+|+|-1| = 5$



$1e = 10 + 13$

complex ↑ distance



$= 2(4) + 3(6) + 4 = 30$

↑ dot product

30 + 6

$2(-2) + 3(-2) + 4 = |-6| = 6$

Sklearn

$$L(w_1, w_2, b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w_1, w_2)$$

Regularization

spatial placement → y_i

$$L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$$

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

$z =$

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

hinge

$L(w_1, w_2, b)$

n: # rows in data

$w_1\ w_2\ b$

$$L = \underset{w_1, w_2, b}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

minimum

gradient descent

$L = \underset{w_1, w_2, b}{\text{argmin}} \sum_{i=1}^{n}$

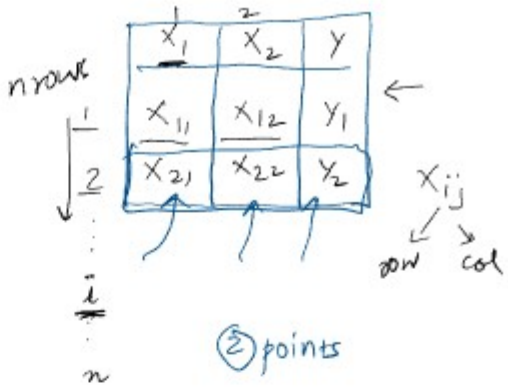
gradient descent

Explanation of Loss Function

23 February 2022 08:07

$$L = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i)) \quad \text{w.r.t } w_1, w_2, b$$

where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$



$\max(0, -y_i f(x_i)) = \max(0, x)$ → $= x$ if $x \geq 0$; $= 0$ if $x < 0$

$-y_i f(x_i) \geq 0$ ↑ $< 0 = 0$

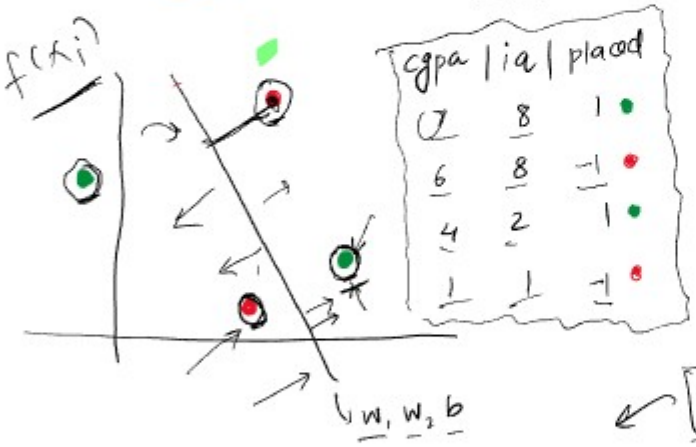
$f(x_2) = w_1 x_{21} + w_2 x_{22} + b$

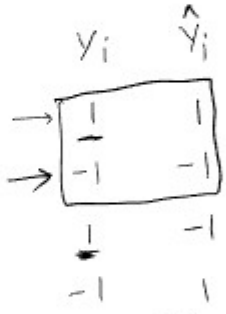
$y_i f(x_i)$ is

$L = \frac{1}{2}\left[\max(0, -y_1 f(x_1)) + \max(0, -y_2 f(x_2))\right]$ = 0

2 points, n points; distance

$\max(0, -y_i f(x_i))$





$w_1 x_1 + w_2 x_2 + b \geq 0$

$\max(0, -ve) = 0$

$\frac{1}{+ve}$ $f(x_i) = +ve$

$+ve \quad +ve = -ve$

$\max(0, -ve) = 0$

y_i $f(x_i) = +ve$

$\frac{-1}{-ve}$ $-ve$ $-ve$

y_i $f(x_i)$

$+ve \quad -ve = -ve = +ve$

$\max(0, +ve) = +ve$

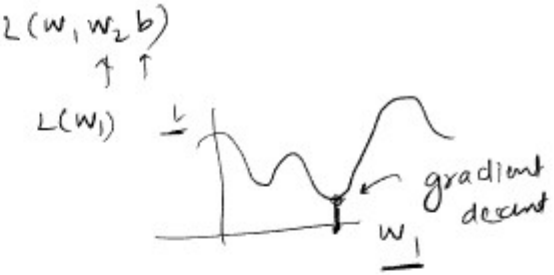
y_i $f(x_i)$

$-ve \quad +ve \rightarrow -ve \; -1 = +ve$

Gradient Descent

23 February 2022 08:50

$$L = \underset{w_1, w_2 b}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i)) \right]$$



where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$

100/1000

$\eta = 0.01$

$(w_1, w_2, b) = 1$

for i in epochs:

$$w_1 = w_1 + \eta \frac{\partial L}{\partial w_1}$$

$$w_2 = w_2 + \eta \frac{\partial L}{\partial w_2}$$

$$b = b + \eta \frac{\partial L}{\partial b}$$

$$\left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial b} \right]$$

Loss Function Differentiation

23 February 2022 12:57

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

where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$

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

cgpa | iq | placed

$\rightarrow x_{i1}$

$$\frac{\partial L}{\partial f(x_i)} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases} \qquad \frac{\partial f(x_i)}{\partial w_1} = x_{i1}$$

$$\frac{\partial L}{\partial w_1} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i x_{i1} & \text{if } y_i f(x_i) < 0 \end{cases}$$

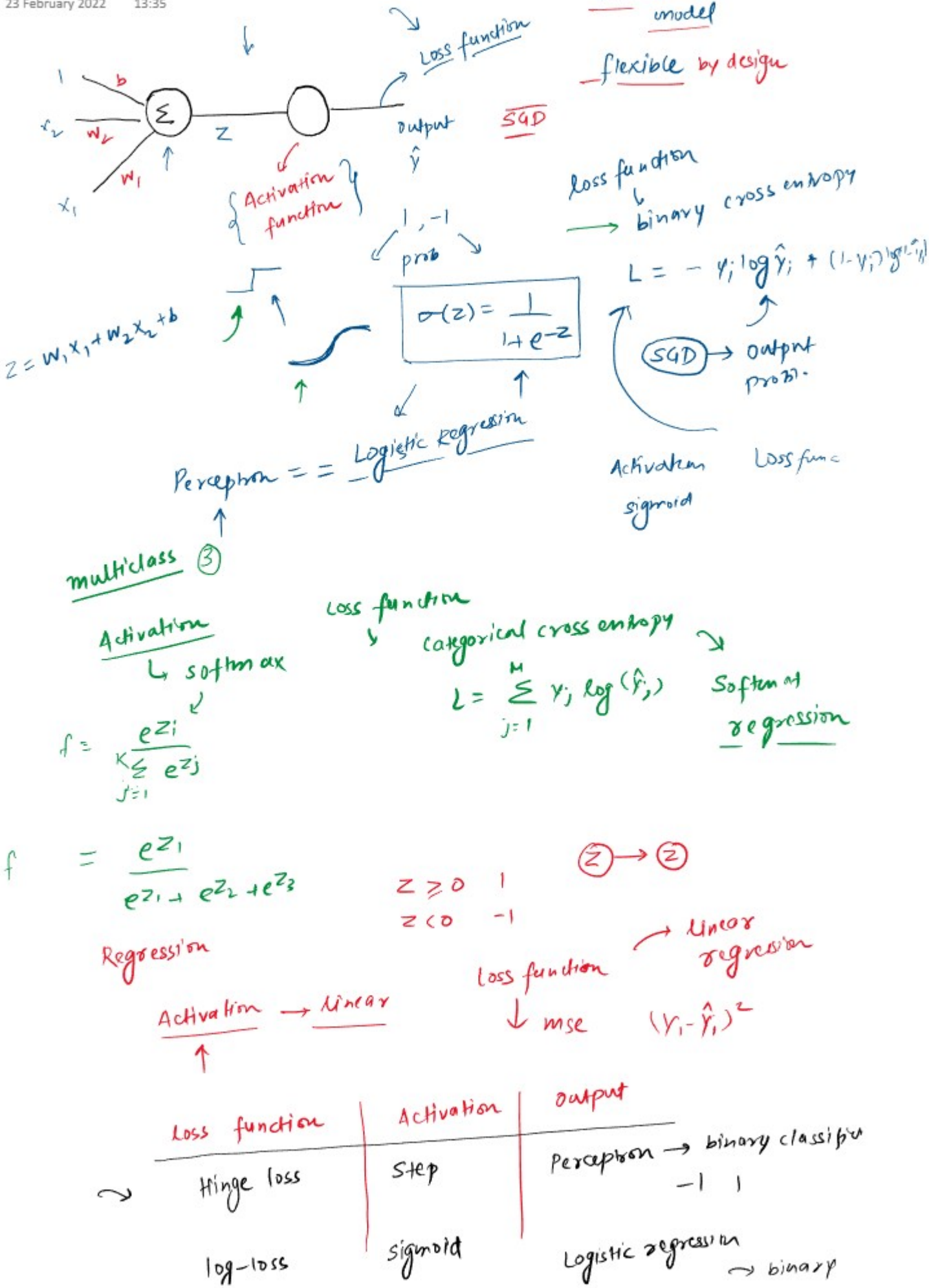
$$\frac{\partial L}{\partial w_2} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i x_{i2} & \text{if } y_i f(x_i) < 0 \end{cases}$$

$\rightarrow$ code using gradient descent

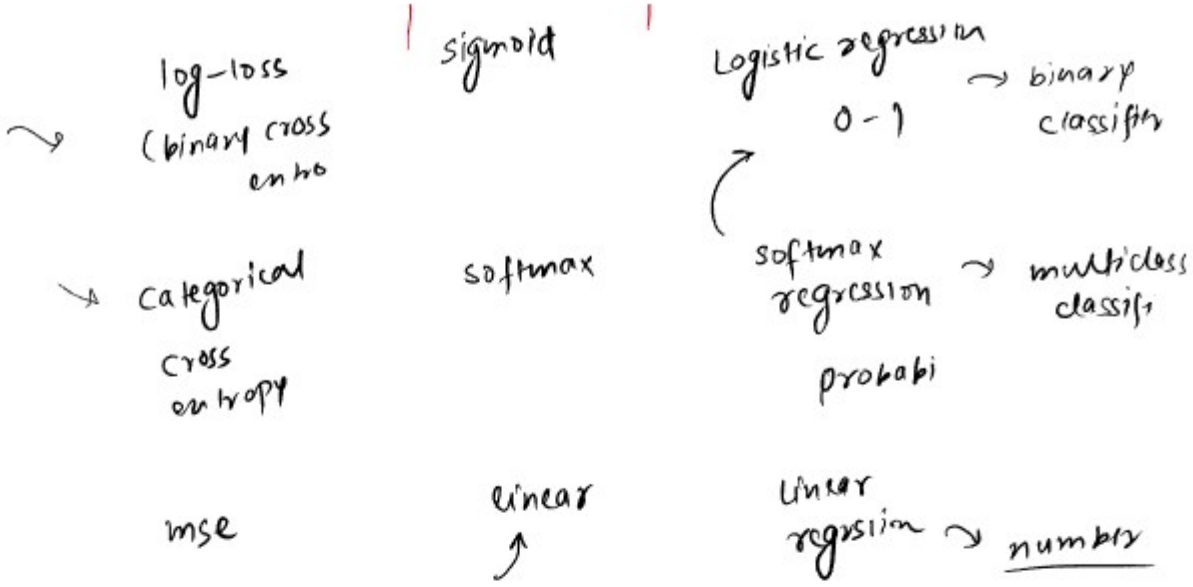
$$\frac{\partial L}{\partial b} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases}$$

More Loss Functions

23 February 2022 13:35

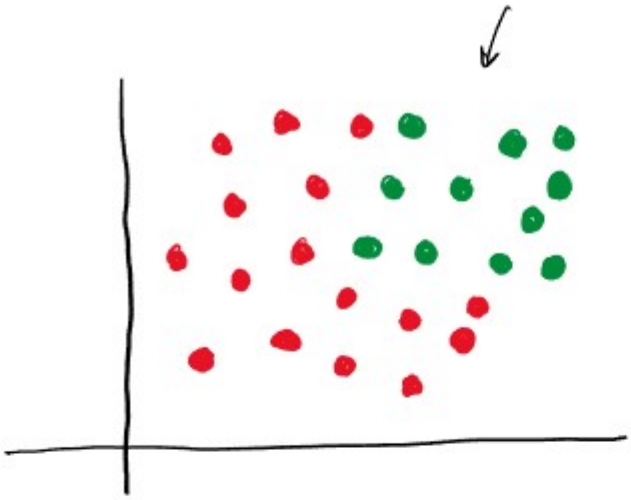


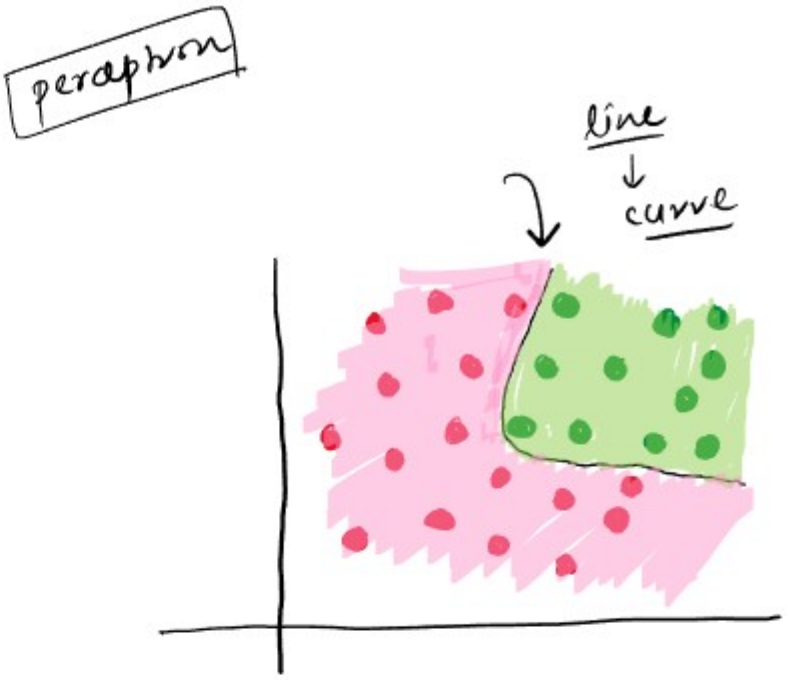
Loss function	Activation	output
Hinge loss	Step	Perceptron → binary classifier (-1, 1)
log-loss	sigmoid	Logistic regression → binary


log-loss
(binary cross entro
categorical
cross
entropy
mse
sigmoid
softmax
linear
Logistic regression
0-1
binary classifi
softmax
regression
multiclass
classifi
probabi
Linear
regrssion
numbr

The Problem

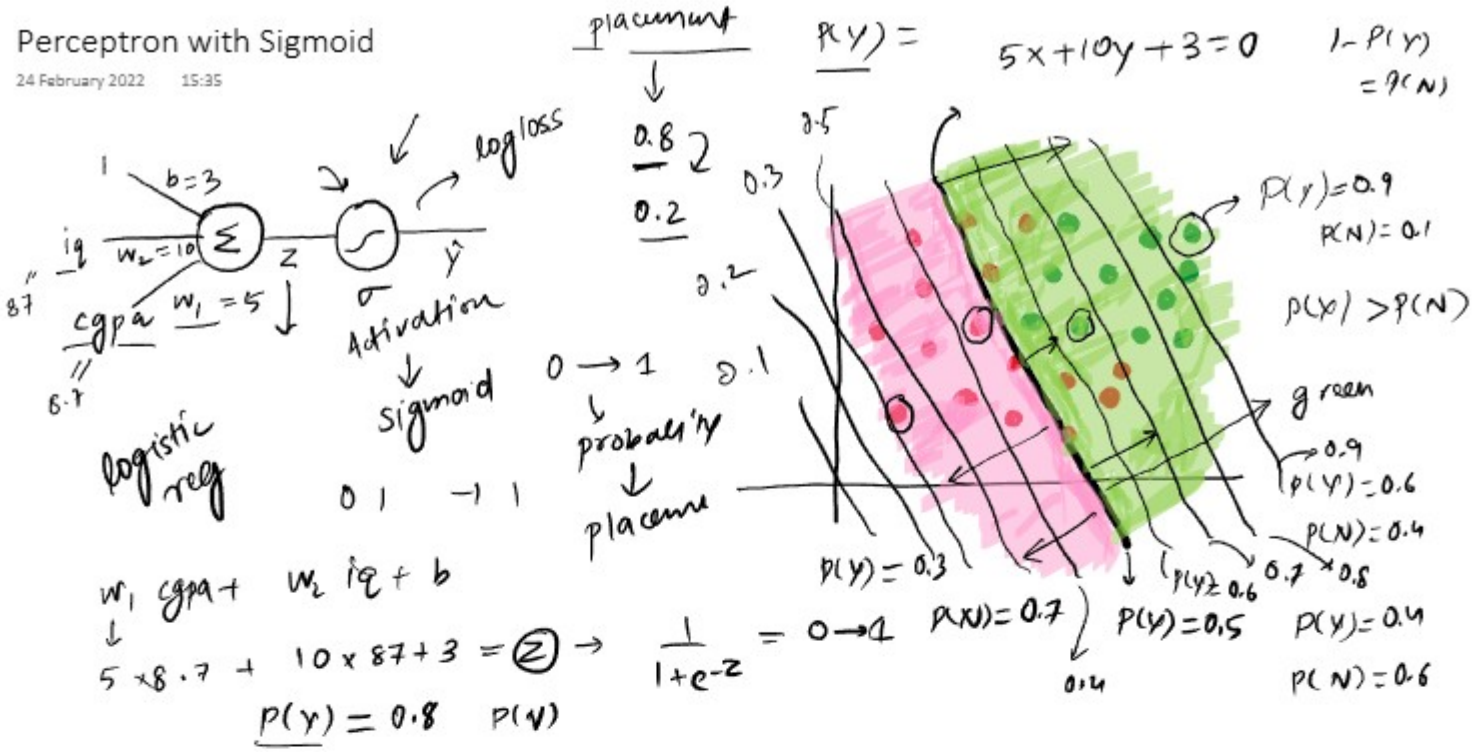
24 February 2022 13:41



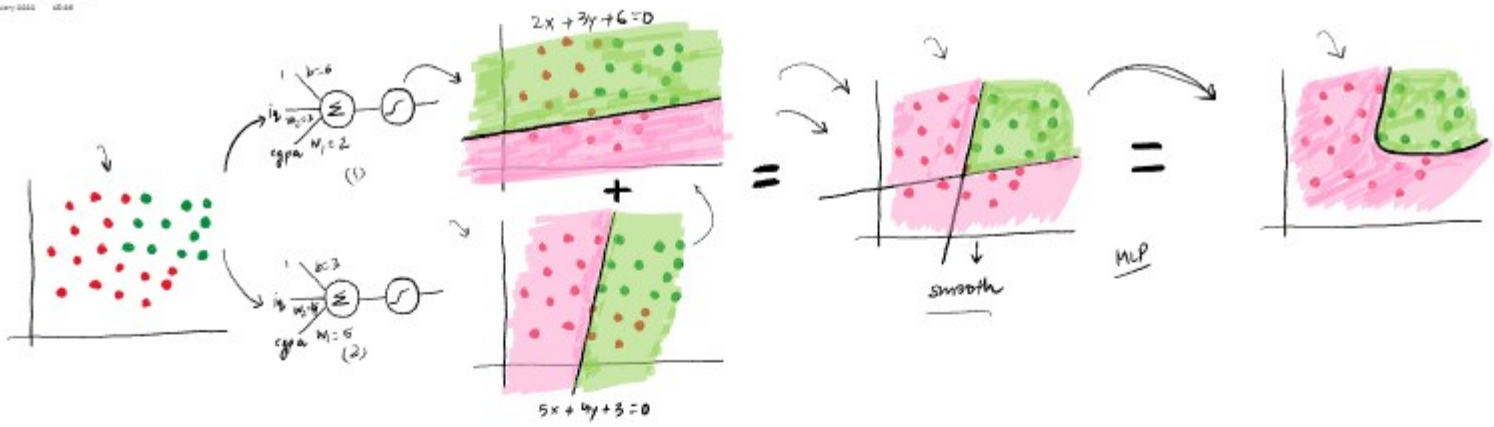


Perceptron with Sigmoid

24 February 2022 15:35

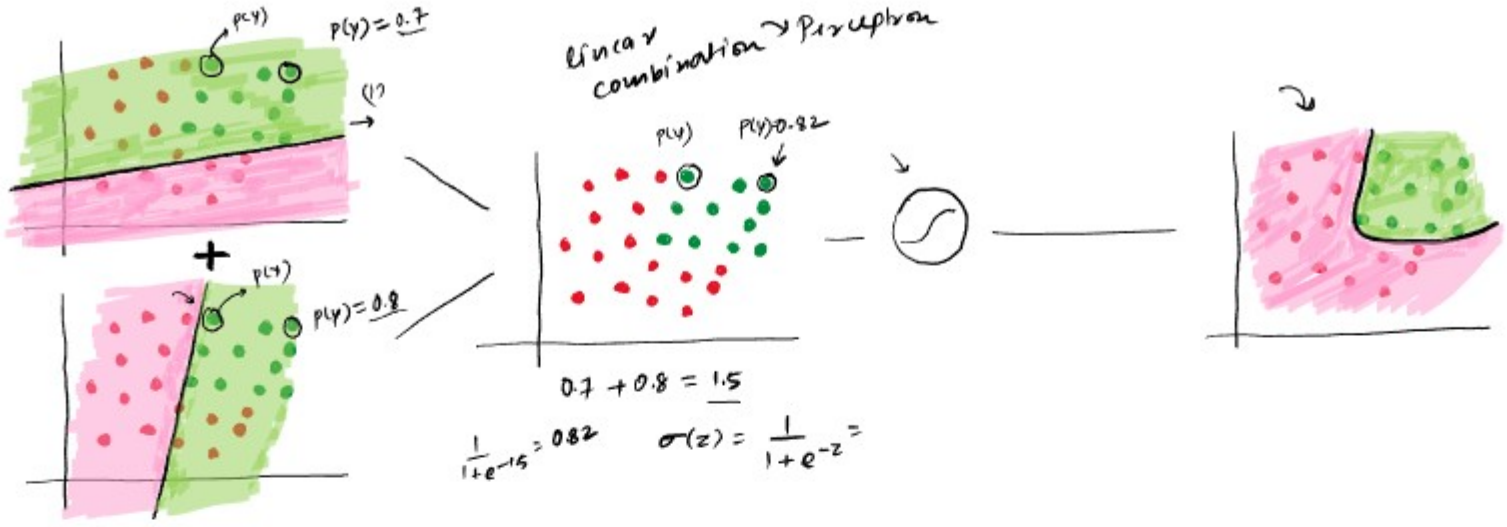


Abstract Solution



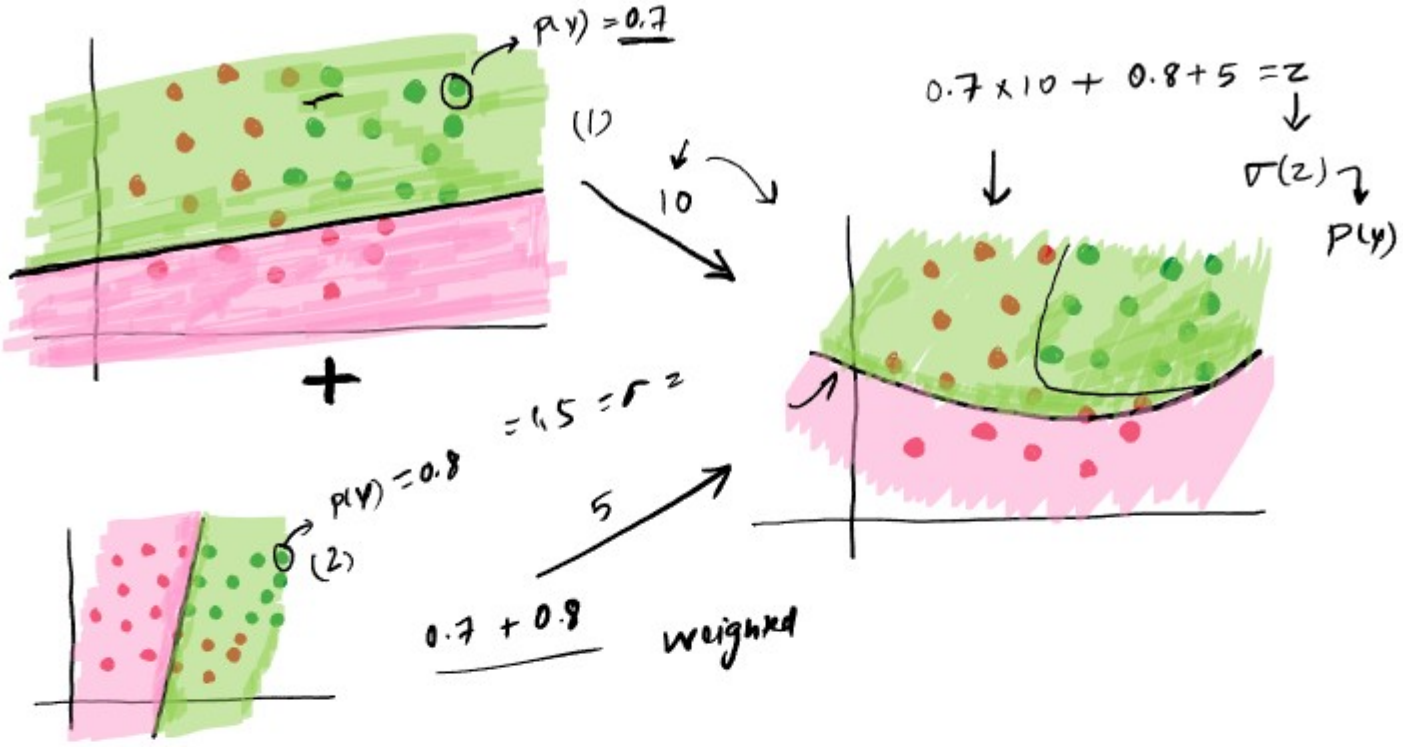
Mathematical Reasoning

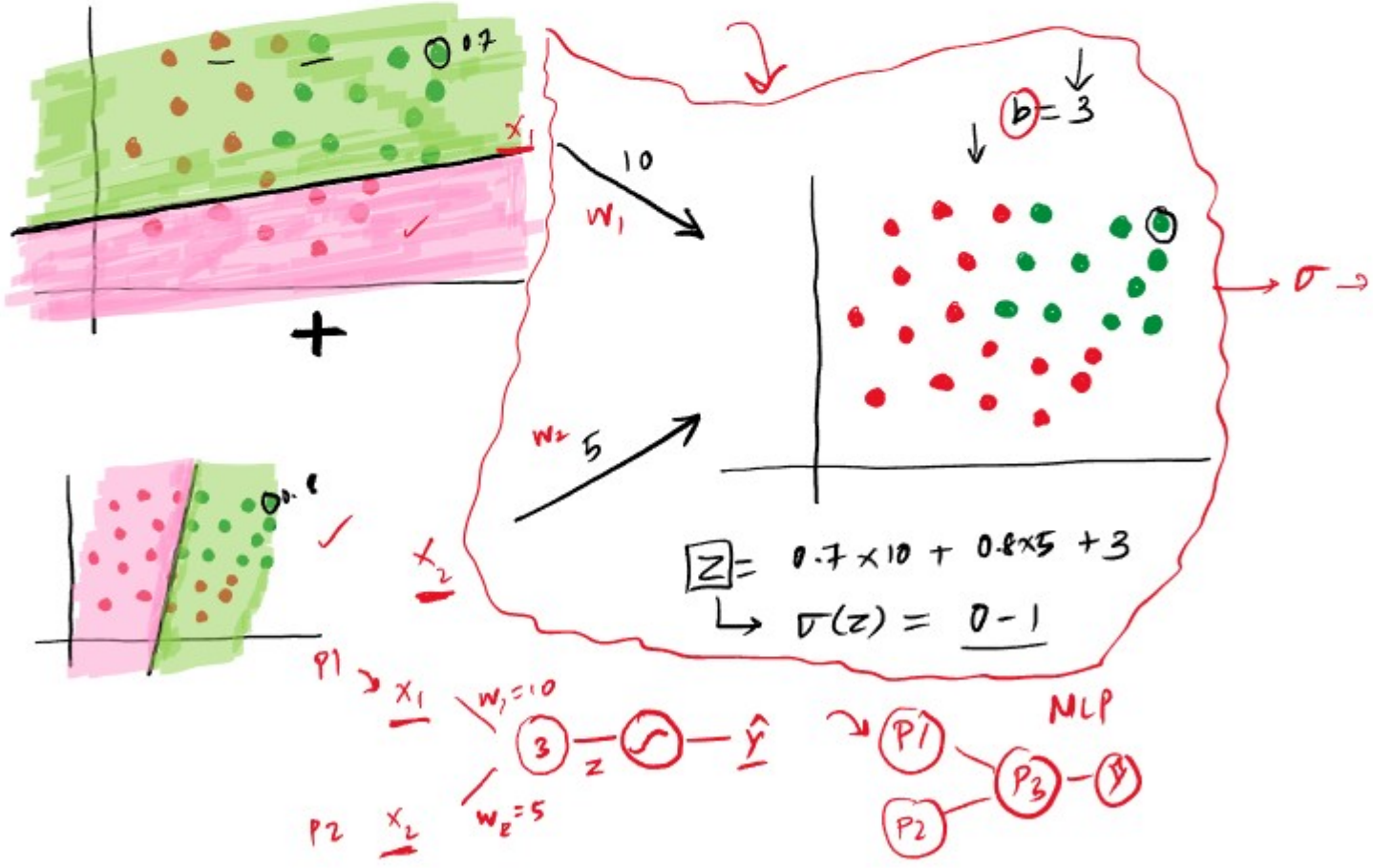
16 February 2022 15:36



Adding Weights

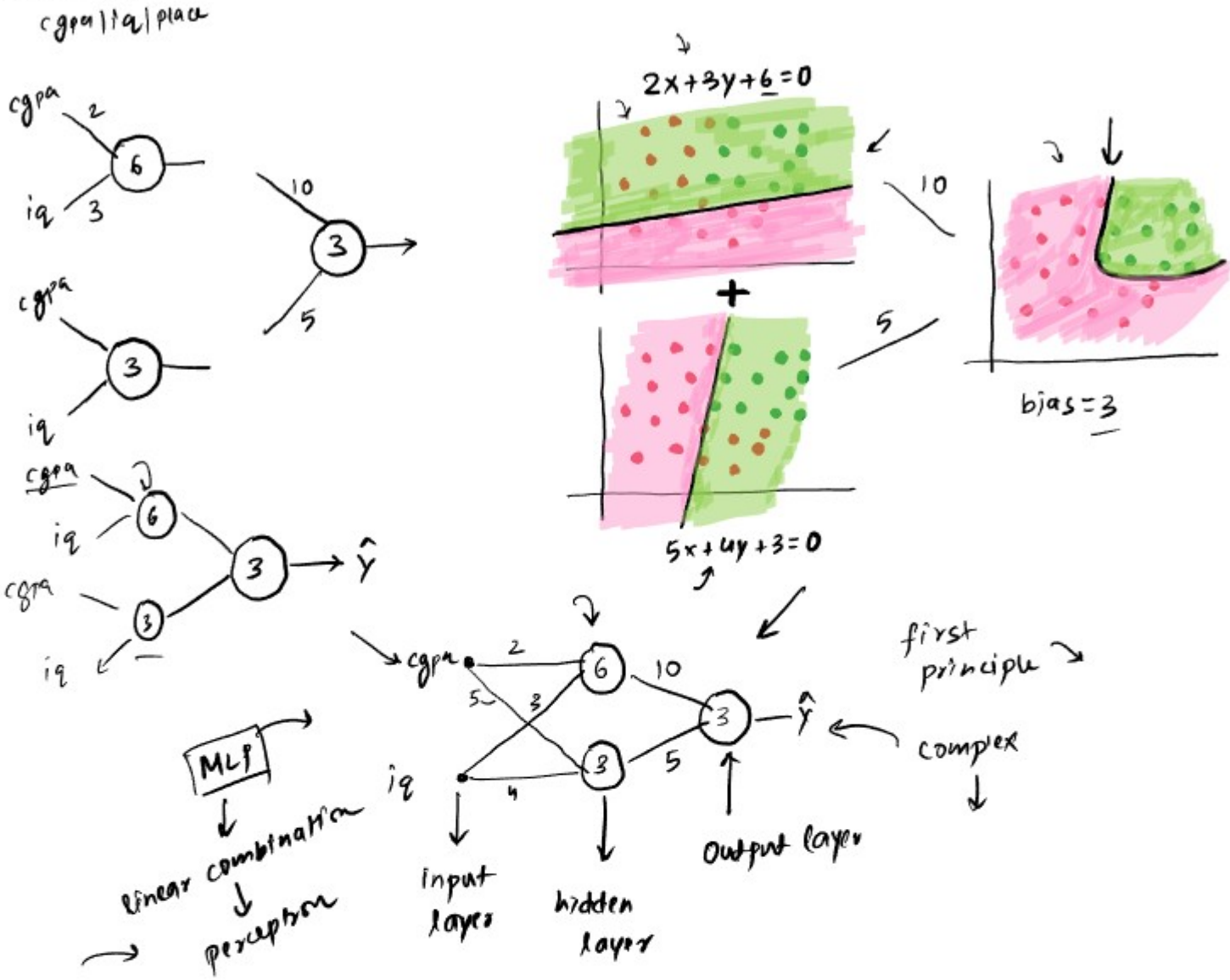
24 February 2022 15:37





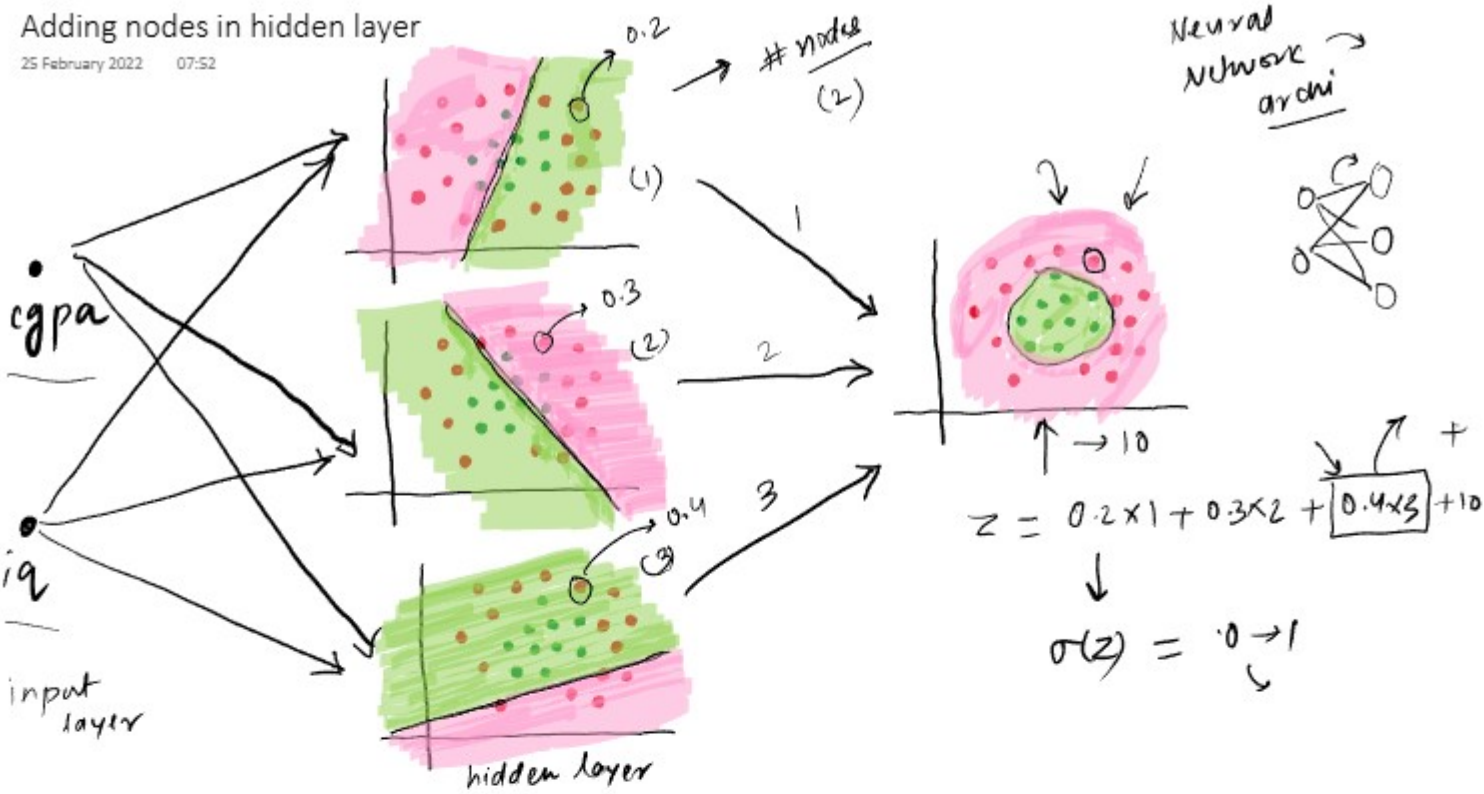
The Idea of MLP

24 February 2022 15:37



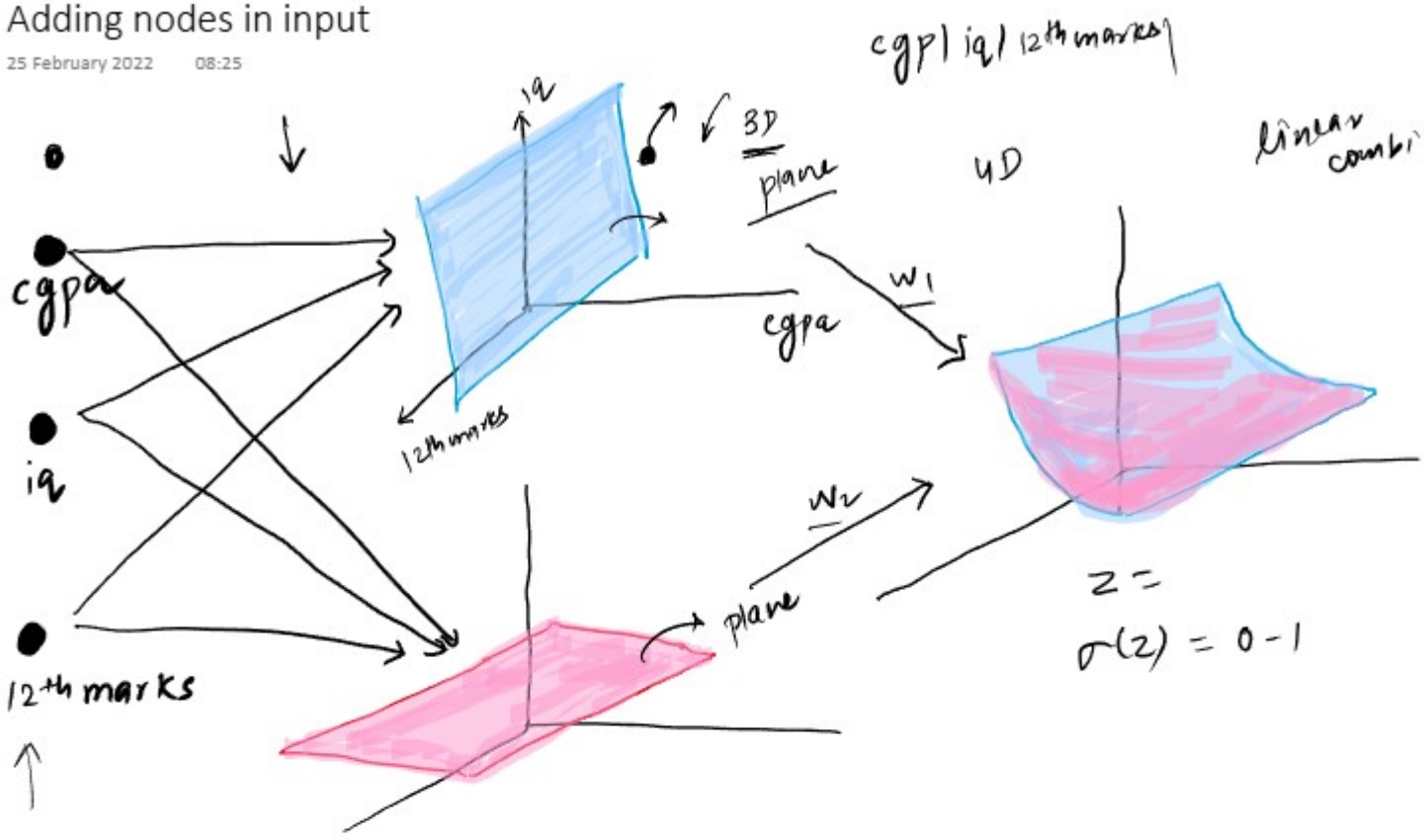
Adding nodes in hidden layer

25 February 2022 07:52



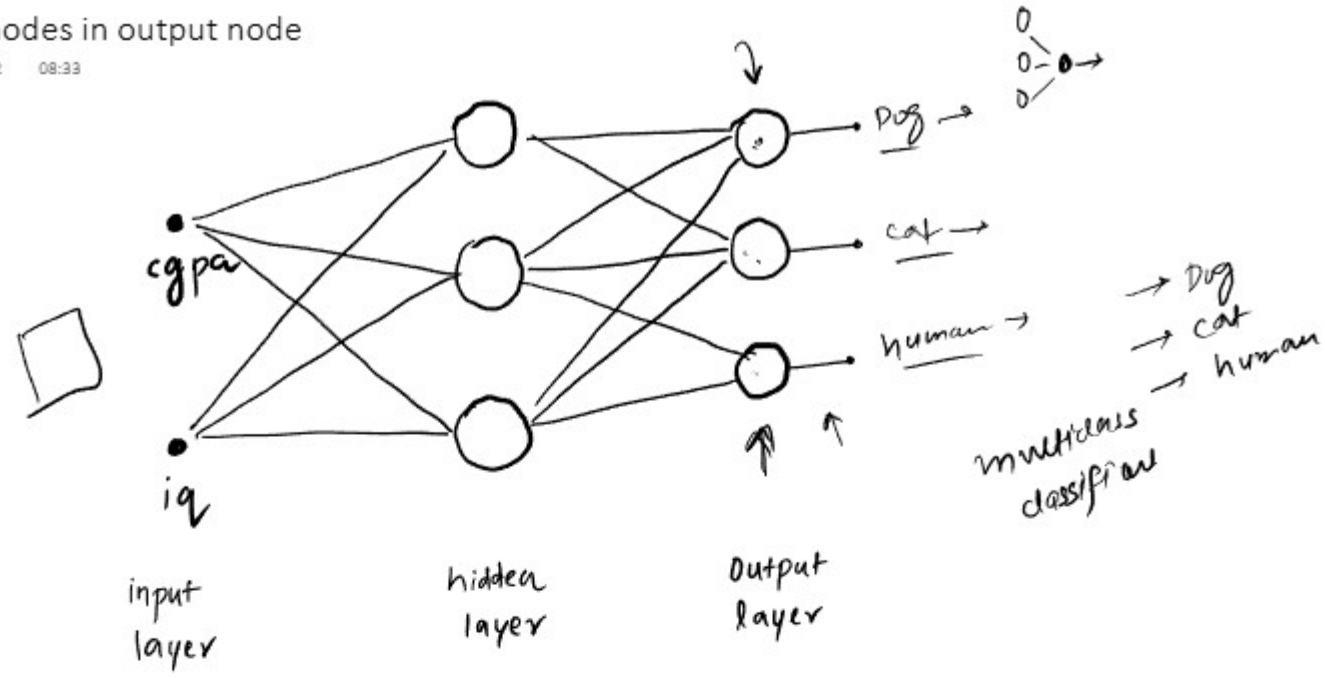
Adding nodes in input

25 February 2022 08:25



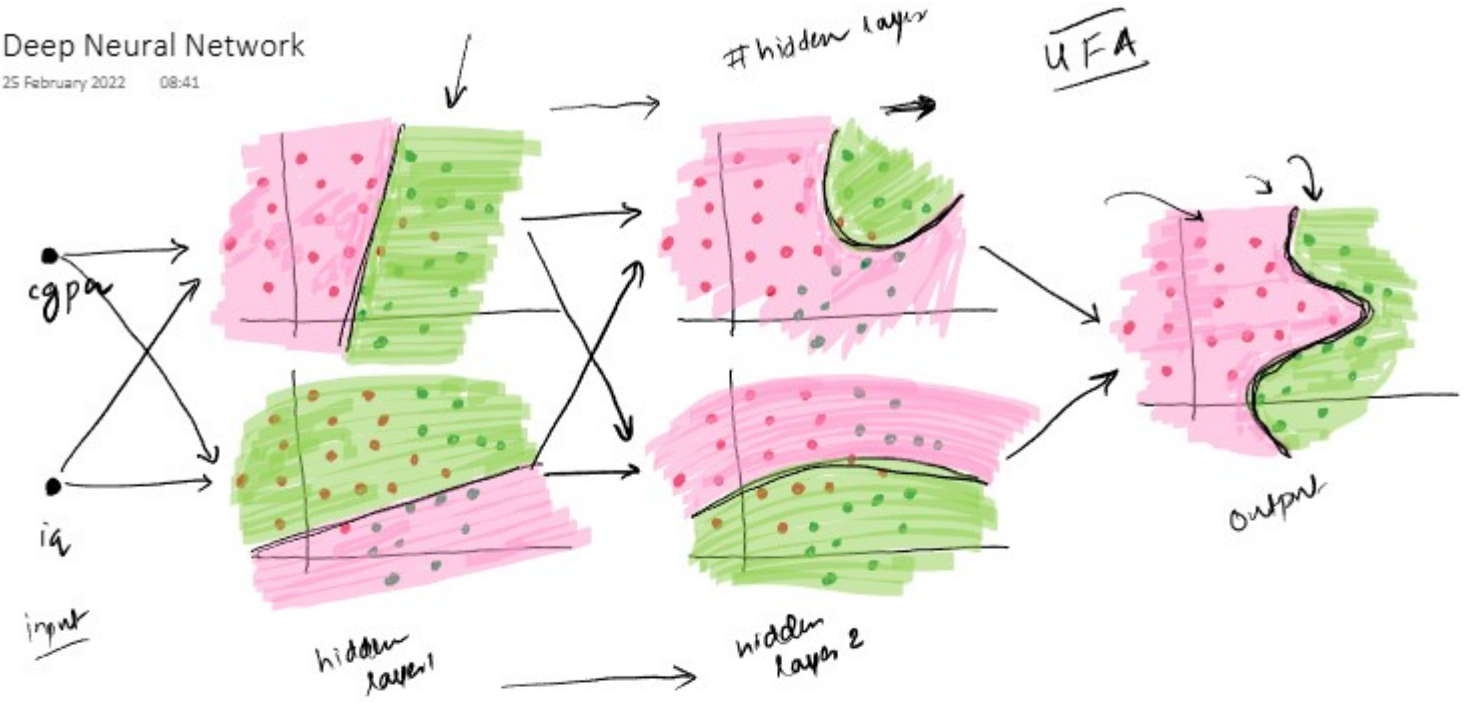
Adding nodes in output node

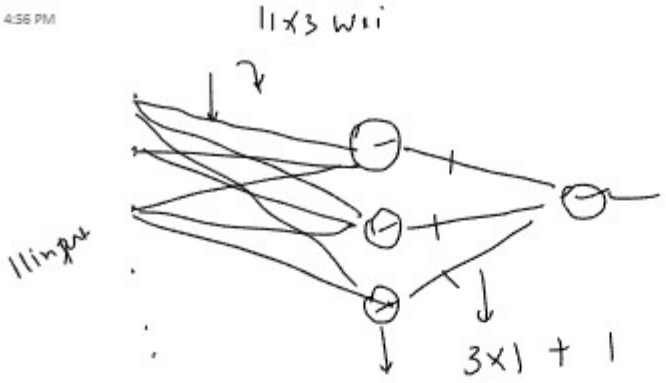
25 February 2022 08:33

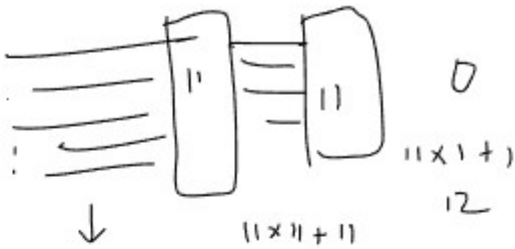


Deep Neural Network

25 February 2022 08:41

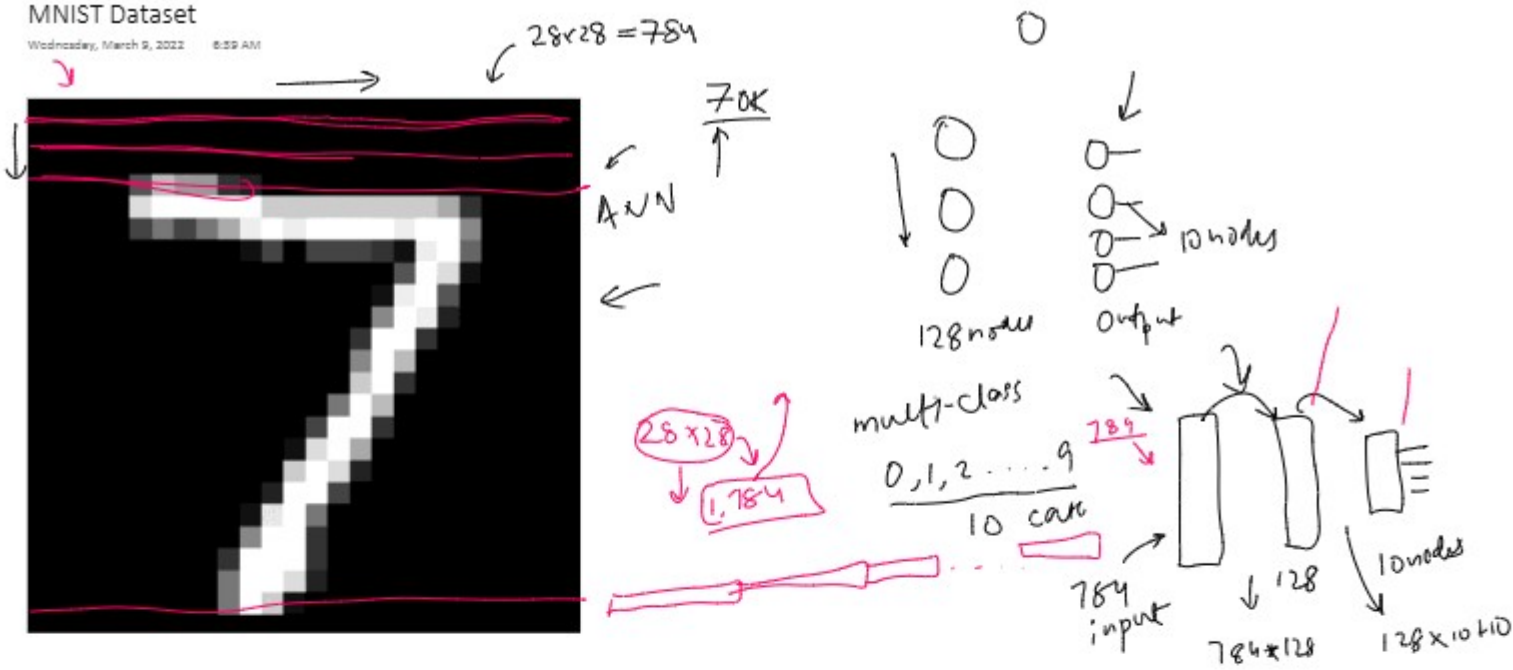


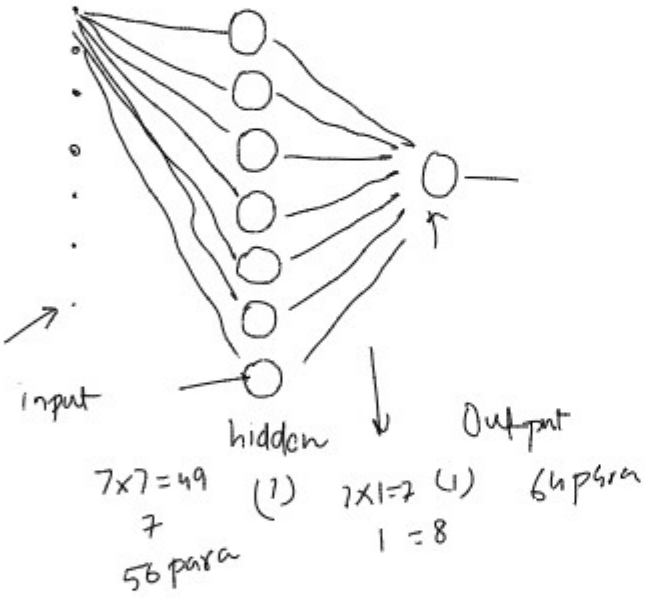

11x3 wii
11inputs
3x1 + 1
11x3 +3
36
4 = 40


11
11
0
11x1 + 1
12
11x11 + 11
133
11x11 + 11
133
266 = 278
12

MNIST Dataset

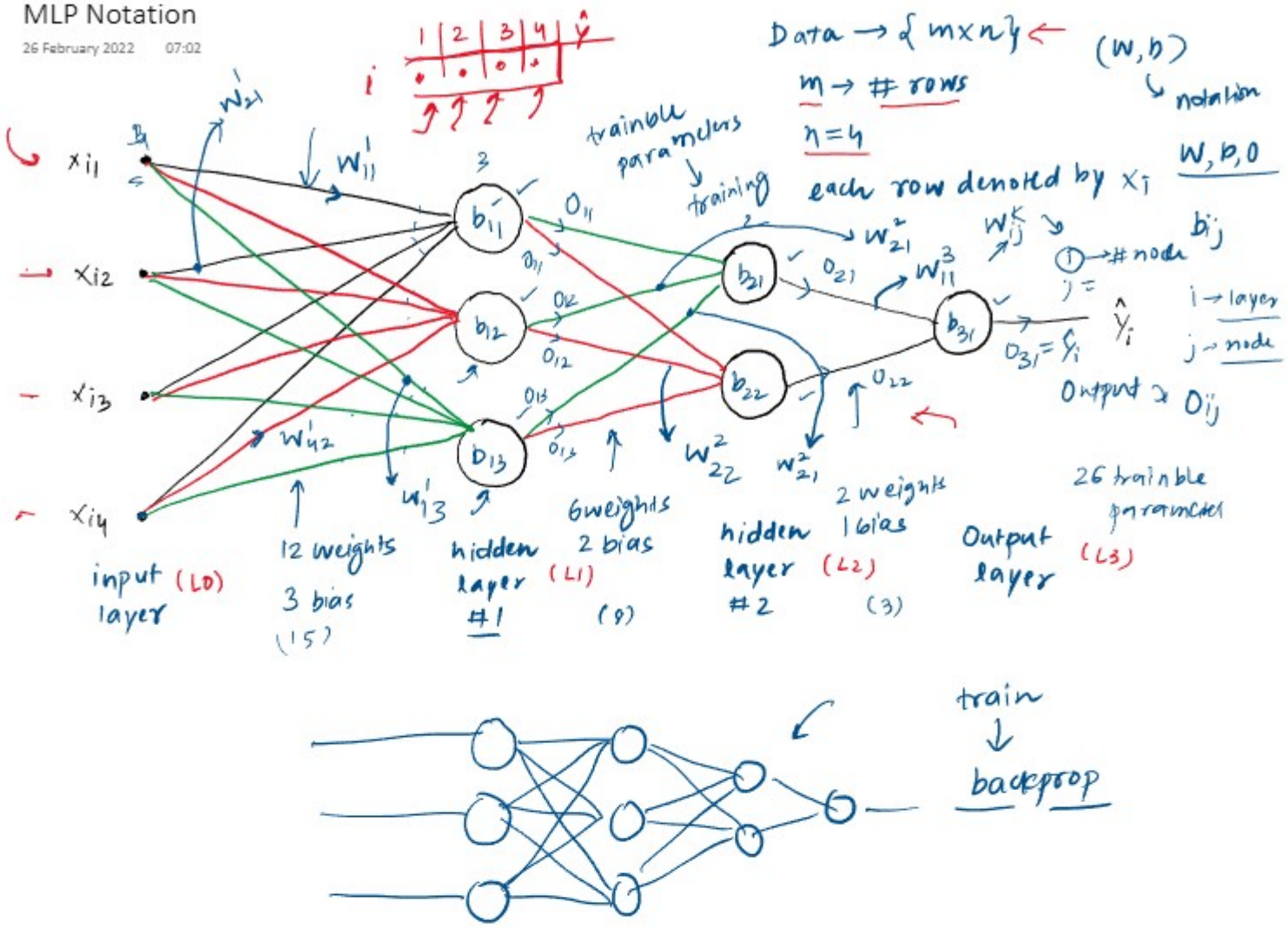
Wednesday, March 9, 2022 8:59 AM





MLP Notation

26 February 2022 07:02

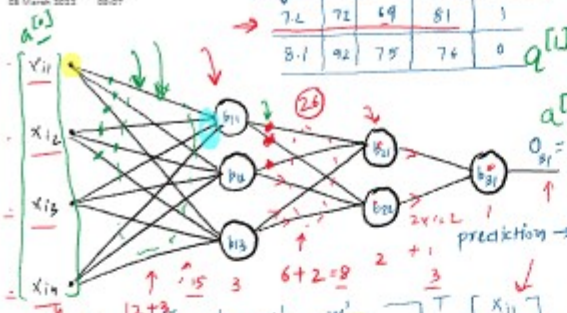


train → backprop

Forward Propogation

cgpa	iq	10th m	12th m	placed
7.2	72	69	81	1
8.1	92	75	76	0

$\hat{y}$ → # of trainable parameters



$a^{[1]} = \sigma(a^{[0]} w^{[1]} + b^{[1]})$

$a^{[2]} = \sigma(a^{[1]} w^{[2]} + b^{[2]})$

$a^{[3]} = \sigma(a^{[2]} w^{[3]} + b^{[3]})$

prediction → $\sigma(w^T x + b)$ →

$$\sigma\left(\sigma\left(\left(\sigma(a^{[0]} w^{[1]} + b^{[1]})\right) w^{[2]} + b^{[2]}\right) w^{[3]} + b^{[3]}\right)$$

$a^{[1]}$; $a^{[2]}$; $a^{[3]} = \hat{y}$

Layer #1

$$\begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \\ w^1_{41} & w^1_{42} & w^1_{43} \end{bmatrix}^T \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$3\times 4 \xrightarrow{T} 4\times 3$; 4×1 ; 3×1 → 3×1

$$= \begin{bmatrix} w^1_{11} x_{i1} + w^1_{21} x_{i2} + w^1_{31} x_{i3} + w^1_{41} x_{i4} \\ w^1_{12} x_{i1} + w^1_{22} x_{i2} + w^1_{32} x_{i3} + w^1_{42} x_{i4} \\ w^1_{13} x_{i1} + w^1_{23} x_{i2} + w^1_{33} x_{i3} + w^1_{43} x_{i4} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} \leftarrow$$

3×1

$$= \sigma\left(\begin{bmatrix} w^1_{11} x_{i1} + w^1_{21} x_{i2} + w^1_{31} x_{i3} + w^1_{41} x_{i4} + b_{11} \\ w^1_{12} x_{i1} + w^1_{22} x_{i2} + w^1_{32} x_{i3} + w^1_{42} x_{i4} + b_{12} \\ w^1_{13} x_{i1} + w^1_{23} x_{i2} + w^1_{33} x_{i3} + w^1_{43} x_{i4} + b_{13} \end{bmatrix}\right)$$

$$= \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix} \rightarrow a^{[1]}$$

Layer #2

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

$3\times 2 \xrightarrow{T} 2\times 3$; 3×1 ; 2×1

$$= \sigma\left(\begin{bmatrix} w^2_{11} O_{11} + w^2_{21} O_{12} + w^2_{31} O_{13} + b_{21} \\ w^2_{12} O_{11} + w^2_{22} O_{12} + w^2_{32} O_{13} + b_{22} \end{bmatrix}\right) = \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} \leftarrow a^{[2]}$$

Layer #3

$$\rightarrow \begin{bmatrix} w^3_{11} \\ w^3_{21} \end{bmatrix}^T \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix}$$

$2\times 1 \xrightarrow{T} 1\times 2$; 2×1 ; 1×1

$$= \sigma\left(\begin{bmatrix} w^3_{11} O_{21} + w^3_{21} O_{22} + b_{31} \end{bmatrix}\right) = \hat{y}_i = \begin{bmatrix} O_{31} \end{bmatrix} \quad a^{[3]}$$

What is Loss Function?

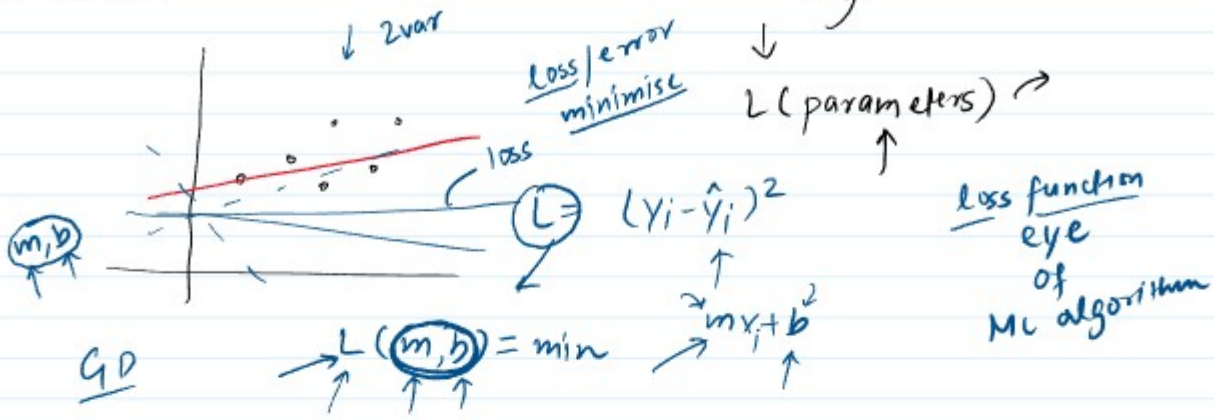
23 March 2022 07:15

Loss function is a method of evaluating how well your algorithm is modelling your dataset.

high → poor

small → great

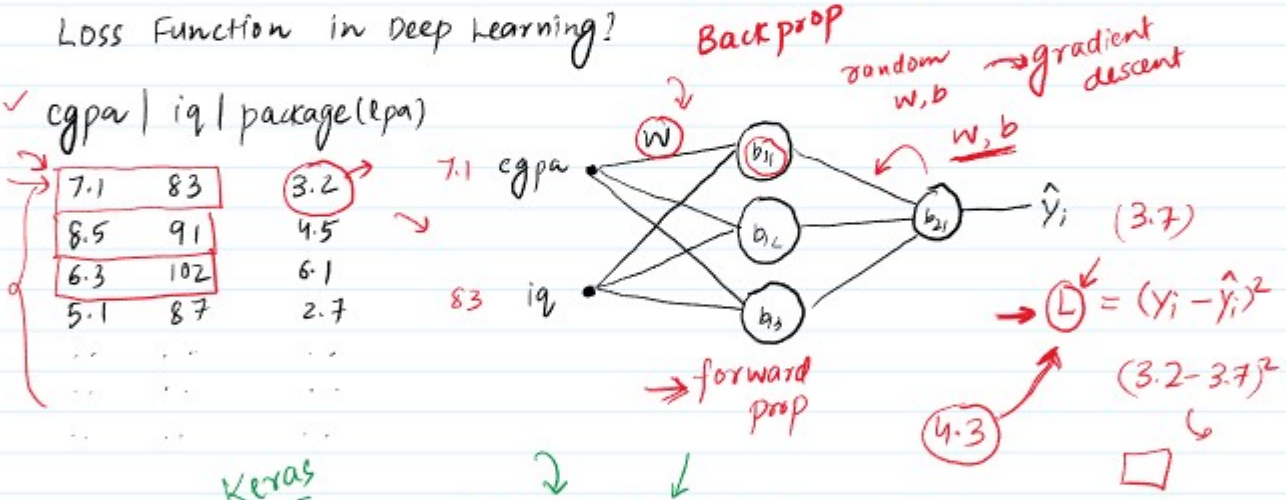
$f(x) = x^2 + 2$



Why is Loss function important?

[You can't improve what you can't measure.] — Peter Drucker

Loss Function in Deep Learning?

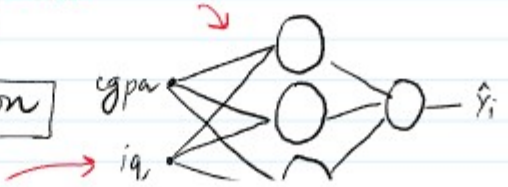


Keras

Loss functions in DL

- Object detection: focal loss
- Regression: mse, mae, huber loss
- Classification: binary cross entropy, categorical cross entro, hinge loss
- Autoencoders: KL divergence
- GAN: discriminator loss, minmax gan loss
- Embedding: Triplet loss

Loss Function Vs Cost Function



Loss Function Vs Cost Function

Error function

Loss function → single training eg.

$y_i = 6.3 \quad \hat{y}_i = 6.1$

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

$(6.3 - 6.1)^2 =$

batch Cost function

$\frac{1}{4}[(6.1-6.3)^2 + (4.1-4)^2 + (3.5-3.7)^2 + (7.2-7)^2] = CF$

$\frac{1}{n}\sum (y_i - \hat{y}_i)^2$

(5) lpa $\sqrt{5}$ lpa $(lpa)^2$

cgpa	iq	y_i) package	$\hat{y}_i$ Prediction	$(y_i - \hat{y}_i)^2$ error
6.3	100	6.3	6.1	0.2
7.1	91	4.1	4	0.1
8.5	83	3.5	3.7	−0.2
9.2	102	7.2	7	0.2
8.1	100	50	8	Overall error

1. Mean Squared Error (MSE)

Squared loss L2 Loss

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

$(\text{true} - \text{predict})^2$

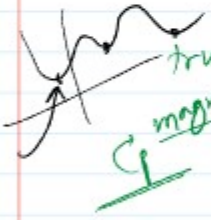
$(6.3 - 6.1)^2 = -$

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

Advan / DisAdv

mae $(y_i - \hat{y}_i)^2$

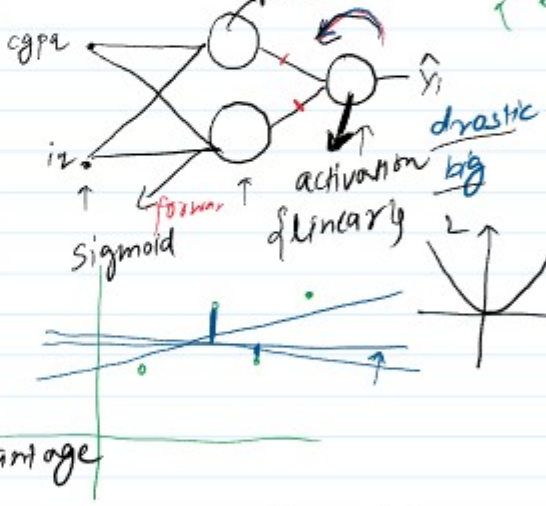
quadratic ^{2} punish



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

1 unit → 1 unit
2 unit → 4 unit
4 unit → 16 unit

(w, b) → (L) min



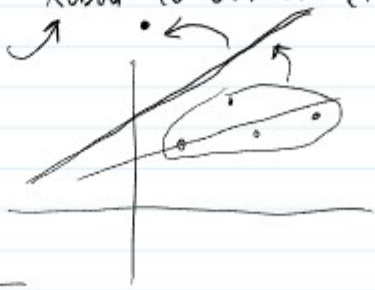
Advantages

1) Easy to interpret
2) Differentiable (GD)
3) 1 local minima

Disadvantage

1) Error unit (squared) → diff
2) Robust to Outliers (Not)

$CF = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$



2. Mean Absolute Error (MAE) → L1 loss

2 lpa

cgpi iq | pack lpa $|(y_i - \hat{y}_i)|$ → lpa

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

2 | abs

$C = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^2$ | true - predict |

punish

$C = \frac{1}{n}\sum_{i=1}^{n} |y_i - y_i|$ | true - predict |

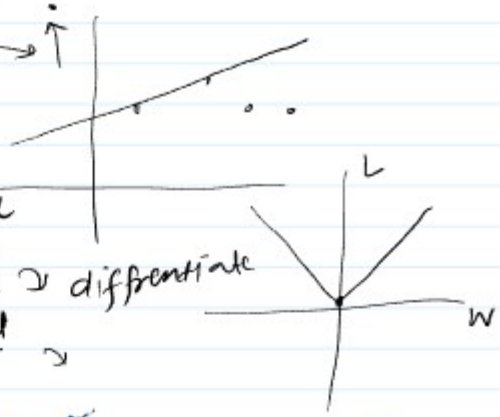
punish

Advantages

1) Intuitive and easy
2) Unit → same - y
3) Robust to outliers

Disadvantage

1) Not differentiable
   - GD → differentiate
   - Subgradient



mae huber mse

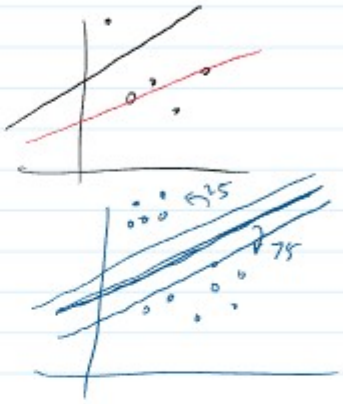
3. Huber Loss ✓

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

mse — mae

hyper

mse — outliers ✓
mae — normal point ✓



4. Binary Cross Entropy — log loss

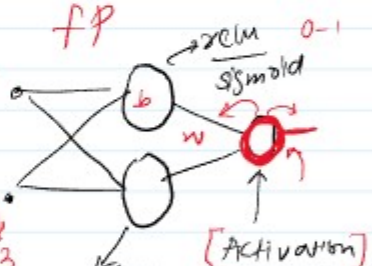
→ classification ✓
→ Two classes ✓

100 days

cgpa	iq	placement	
8	80	1	0.73
7	70	0	0.25
6	60	0	0.13

0.12

GD fp



Loss function = $-y \log(\hat{y}) - (1-y)\log(1-\hat{y})$

Keras

y → actual value / target
$\hat{y}$ → NN prediction

$-(1-0)\log(1-0.25)$
$-1 \log(0.75)$
-0.12

$-1 \log(0.73)$
$-1 \times -0.13 = 0.13$

Cost function $= -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$

maximum likli

Logistic Reg

Advantage

* Differentiable

Disadvan

→ multi local minim
→ Intuitive

5. Categorical Cross Entropy [used in Softmax Regression]

→ [illegible] classification ✓

OHE

cgpa	iq	placed?	Yes	No	Maybe

5. Categorical Cross Entropy [used in softmax Regression]

→ [Multi-class] classification

1 point

$$L = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where K is # classes in the data

cgpa	iq	placed?
8	80	Yes 1
6	60	No 2
7	70	Maybe 3

Yes	No	Maybe	
1	0	0	Yes
0	1	0	No
0	0	1	Maybe

1 point

$$L = -(y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3))$$

[0.2 0.3 0.5]

[1 0 0]

$$L = -1 \times \log(0.2)$$

$$\frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$$f(z) = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$$ softmax

GD, Wb

8, 6, 60, 80

forward prop

Activation

neurons

Class ← categories

Yes 0.3, No 0.6, Maybe 0.1

[0.3 0.6 0.1]

[0 1 0]

$-1 \times \log(0.6) =$ —

Sparse Categorical Cross Entropy

	cgpa	iq	placed	OHE
1/2/3	7	70	Yes	1
	8	80	No	2
	6	60	Maybe	3

[0.1 0.4 0.5]

L

$-1 \times \log(0.1)$

[0.1 0.4 0.6]

$-1 \times \log(0.4)$

[0.6 0.2 0.2]

$-1 \times \log(0.2)$

{SCE, CCE} Loss function

SCE → fast

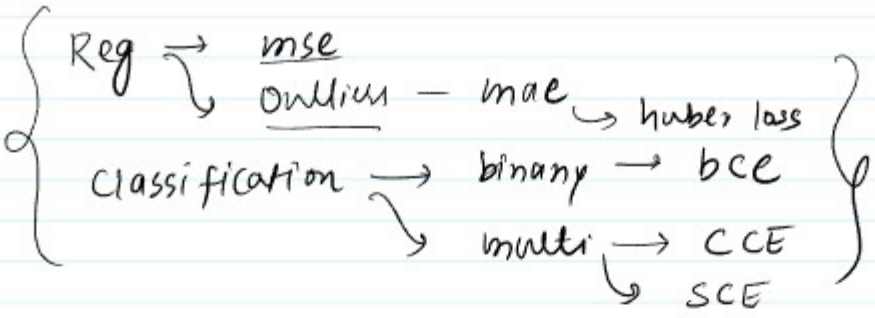
fast

OHE — fast

SCE

$$L = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$ Cost function

$$C = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \log(\hat{y}_{ij})$$


Reg
mse
outliers
mae
huber loss
classification
binary
bce
multi
CCE
SCE

23 March 2022 07:31

Backpropogation

Pre requisite

→ Gradient descent (v) →
→ Forward Propogation (v) →

Backprop → Algo → train nn (How)

training

cgpa	iq	lpa
8	80	8
7	70	7

reg

predict

Weights, bias → correct value

cgpa, iq → $\hat{y}$ (W, b)

$\sigma(z)$

$O_{21} = W^2_{11} O_{11} + W^2_{21} O_{12} + b_{21}$

6 + 3 → 9 trainable

Backprop

4 students

iq	cgpa	lpa	error 80 iq
80	8	3	225
60	9	5	
70	5	8	
120	7	11	

4 activation = linear, Linear

L min → $\hat{y} \to O_{21}$

$\hat{y} = O_{21}$ (W, b) Random val

$W \to 1, b \to 0$

18 lpa

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

$(3-18)^2 = 225$ error

Steps →
0) Init W, b
1) You select a point (row) → student
2) Predict (lpa) → forward prop [Dot product]
3) Choose a loss function (mse)
4) Weights and bias update
↳ Gradient Descent

Hierarchy: Backprop

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

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

$$W^2_{11\,new} = W^2_{11\,old} - \eta \frac{\partial L}{\partial W^2_{11}}$$

$$W^2_{21\,new} = W^2_{21\,old} - \eta \frac{\partial L}{\partial W^2_{21}}$$

$$b_{21\,new} = b_{21\,old} - \eta \frac{\partial L}{\partial b_{21}}$$

derivative of loss wrt weight

$\frac{\partial L}{\partial W^2_{11}}$ ← partial derivative

derivative

$\frac{\partial L}{\partial W^2_{11}}, \frac{\partial L}{\partial W^2_{21}}, \frac{\partial L}{\partial b_{21}}$ | $\frac{\partial L}{\partial W^1_{11}}, \frac{\partial L}{\partial W^1_{21}}, \frac{\partial L}{\partial b_{11}}$ | $\frac{\partial L}{\partial W^1_{12}}, \frac{\partial L}{\partial W^1_{22}}, \frac{\partial L}{\partial b_{12}}$

iq, cgpa → O_{21} → $\hat{y} \to L$

$$\frac{\partial L}{\partial W^2_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W^2_{11}}$$ → chain rule of diffrn

$\frac{dy}{dx}$

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

$$\frac{\partial \hat{y}}{\partial W^2_{11}} = \frac{\partial}{\partial W^2_{11}} [O_{11} W^2_{11} + O_{12} W^2_{21} + b_{21}]$$

$$= O_{11}$$

$$\frac{\partial L}{\partial W^2_{11}} = -2(y - \hat{y}) O_{11}$$

$$\frac{\partial L}{\partial W^2_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W^2_{21}}$$

$$\frac{\partial \hat{y}}{\partial W^2_{21}} = \frac{\partial}{\partial W^2_{21}} [O_{11} W^2_{11} + O_{12} W^2_{21} + b_{21}]$$

$$= O_{12}$$

$$\frac{\partial L}{\partial W^2_{21}} = -2(y - \hat{y}) O_{12}$$

$\hat{y} \to b$

$$\frac{\partial \hat{y}}{\partial b_{21}} = 1$$

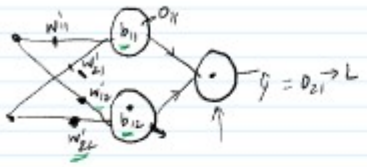
$$\frac{\partial L}{\partial w_{21}^2} = -2(y-\hat{y})o_{12}$$ 2

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

$\hat{y} \to b$, L

$$\frac{\partial \hat{y}}{\partial b_{21}} = 1$$

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial b_{21}}$$



$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{11}} \frac{\partial o_{11}}{\partial w_{11}^1} \to -2(y-\hat{y}) w_{11}^2 x_{i1}$$ 4

$$\frac{\partial L}{\partial w_{21}^1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{11}} \frac{\partial o_{11}}{\partial w_{21}^1} \to -2(y-\hat{y}) w_{11}^2 x_{i2}$$ 5

$$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{11}} \frac{\partial o_{11}}{\partial b_{11}} \to -2(y-\hat{y}) w_{11}^2$$ 6

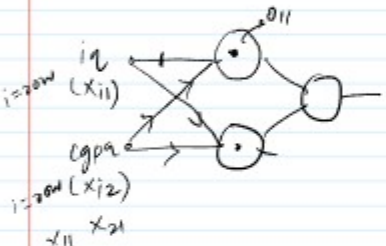
$$\frac{\partial L}{\partial w_{12}^1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{12}} \frac{\partial o_{12}}{\partial w_{12}^1} \to -2(y-\hat{y}) w_{21}^2 x_{i1}$$ 7

$$\frac{\partial L}{\partial w_{22}^1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{12}} \frac{\partial o_{12}}{\partial w_{22}^1} \to -2(y-\hat{y}) w_{21}^2 x_{i2}$$ 8

$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{12}} \frac{\partial o_{12}}{\partial b_{12}} \to -2(y-\hat{y}) w_{21}^2$$ 9

w_{11}^2

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



$$\frac{\partial o_{11}}{\partial w_{11}^1} = \frac{\partial}{\partial w_{11}^1}[iq\, w_{11}^1 + cgpa\, w_{21}^1 + b_{11}] = (iq)$$

$$\frac{\partial o_{11}}{\partial w_{21}^1} = x_{i2}$$

$$\frac{\partial o_{11}}{\partial b_{11}} = 1$$

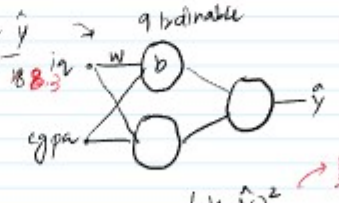
$$\frac{\partial o_{12}}{\partial w_{12}^1} = \frac{\partial}{\partial w_{12}^1}[iq\, w_{12}^1 + cgpa\, w_{22}^1 + b_{12}] = iq\,(x_{i1})$$

$$\frac{\partial o_{12}}{\partial w_{22}^1} = x_{i2}$$

$$\frac{\partial o_{12}}{\partial b_{12}} = 1$$

multiple changes

cgpa	iq	lpa $\hat{y}$
8	80	8
6	60	6
7	70	7
9	90	9



epochs

Steps (once again)

0) weights/bias → init (random $w=1$, $b=0$)

loop - 100/100

converge

1) for i in range(9):

1a) 1 student → forward prop → lpa) → 12 min

1b) Loss calculate (mse) →

1c) Adjust all weights and bias ↗.

The How

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$ → 9 times

Back propagation Algorithm

epochs = 5

for i in range(epochs):

    for j in range(X.shape[0]):

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

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

Ready

```
for i in range(epochs):
    for j in range(X.shape[0]):
        → select 1 row (random)
        → Predict (using forward prop)
        → Calculate loss (using Loss function → mse)
        ↑
        → Update weights and bias using GD
                W_n = W_o - η ∂L/∂W
    → Calculate avg loss for the epoch
      Loss
```

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

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

$$\frac{\partial L}{\partial w^1_{11}} = -2(y-\hat{y})w^2_{11}x_{i1}$$

$$\frac{\partial L}{\partial w^1_{21}} = -2(y-\hat{y})w^2_{11}x_{i2}$$

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

$$\frac{\partial L}{\partial w^1_{12}} = -2(y-\hat{y})w^2_{21}x_{i1}$$

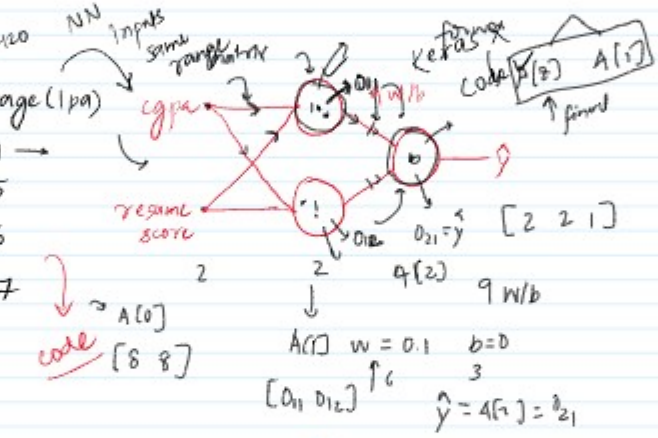
$$\frac{\partial L}{\partial w^1_{22}} = -2(y-\hat{y})w^2_{21}x_{i2}$$

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

Regression Problem — 80-120 — NN inputs same range

cgpa (X_{i1})	resume score (X_{i2})	package (lpa)
8	8	4
7	9	5
6	10	6
5	12	7

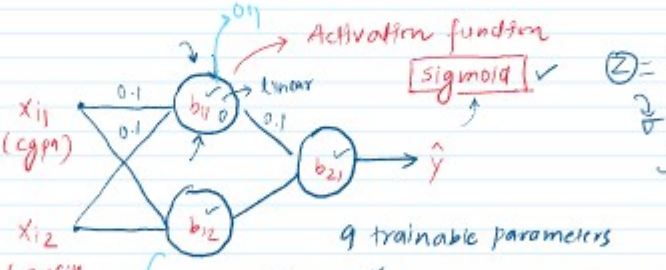
4 students



code → [8 8]

Classification Example

cgpa	profile score	placement
8	8	1
7	9	1
6	10	0
5	5	0



$z = 0.1 \times 8 + 0.1 \times 8 + 0$

$\sigma(z) = O_{11}$

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

Backpropagation Algorithm

```
epochs = 5
for i in range(epochs):  (rows)
    for j in range(X.shape[0]):
        → select 1 row (random) → 1by1
        → Predict (using forward prop) → predict
        → Calculate loss (using Loss function → bce) → code convert
        ↑
        → Update weights and bias using GD
```

Backpropagation Algorithm

Cross entropy

epochs = 5 ✓

for i in range(epochs): (rows)

  for j in range(X.shape[0]):

    → select 1 row (random) → 1by1

    → Predict (using Forward prop) → predict

    → Calculate loss (using loss function → ~~mse~~ bce) → { code convert }

    → Update weights and bias using GD

$$W_n = W_0 - \eta \left(\frac{\partial L}{\partial W}\right)$$ ← q w, b update

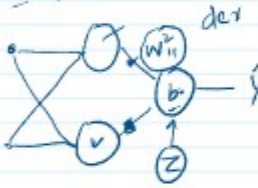
  → Calculate avg loss for the epoch

Loss

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

$\hat{y} = \sigma(z)$

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



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

$\hat{y} \to z_{final} \to w_{11}^2 \times w_{21}^2 \to L$

der wrt L

$\hat{y} \to L$

$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_{11}^2} = -(y-\hat{y})\, o_{11} \quad (3)$$

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

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial b_{21}} = -(y-\hat{y})$$

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

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

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

$\hat{y} = \sigma(z)$

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

$\sigma(z) = \hat{y}$

$$\frac{\partial \hat{y}}{\partial z} = \frac{\partial}{\partial z}(\sigma(z)) = \sigma(z)[1-\sigma(z)] = \hat{y}(1-\hat{y})$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})} \qquad \frac{\partial \hat{y}}{\partial z} = \hat{y}(1-\hat{y}) \Rightarrow \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})} \times \hat{y}(1-\hat{y})$$

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

$$\frac{\partial z_p}{\partial w_{11}^1} = w_{11}^1 x_{11} + w_{21}^1 x_{12} + b_{11}$$

$$\frac{\partial L}{\partial w_{11}^1}, \frac{\partial L}{\partial w_{21}^1}, \frac{\partial L}{\partial b_{11}}$$

$\to -(y-\hat{y})$

$z_f = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$ $o_{11}(1-o_{11})$

$o_{11} = \sigma(z_{prev})$!

$\frac{\partial L}{\partial w^1_{11}}$, $\frac{\partial L}{\partial w^1_{21}}$, $\frac{\partial L}{\partial b_{11}}$

$z_f = w^2_{11} o_{11} + w^2_{21} o_{12} + b_{21}$ $o_{11}(1-o_{11})$

$o_{11} = \sigma(z_{prev})$

$= \sigma(z_{prev})[1-\sigma(z_{prev})]$

$L \to \hat{y} \to z_f \to o_{11} \to z_{prev} \to w^1_{11}$

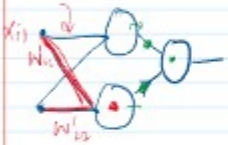
$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{prev}} \times \frac{\partial z_{prev}}{\partial w^1_{11}}$$

$\frac{\partial L}{\partial \hat{y}} = -(y-\hat{y})$; $\frac{\partial \hat{y}}{\partial z_f} = 1$; $\frac{\partial z_f}{\partial o_{11}} = w^2_{11}$; $\frac{\partial o_{11}}{\partial z_{prev}} = o_{11}(1-o_{11})$; $\frac{\partial z_{prev}}{\partial w^1_{11}} = x_{i1}$

$$\frac{\partial L}{\partial w^1_{11}} = -(y-\hat{y}) w^2_{11} o_{11}(1-o_{11}) x_{i1} \quad (3)$$

$$\frac{\partial L}{\partial w^1_{21}} = -(y-\hat{y}) w^2_{11} o_{11}(1-o_{11}) x_{i2}$$

$$\frac{\partial L}{\partial b_{11}} = -(y-\hat{y}) w^2_{11} o_{11}(1-o_{11})$$



$\frac{\partial L}{\partial w^1_{12}}$, $\frac{\partial L}{\partial w^1_{22}}$, $\frac{\partial L}{\partial b_{12}}$

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

$L \to \hat{y} \to z_f \to o_{12} \to w^1_{12} \leftarrow z_p$

$$\frac{\partial L}{\partial w^1_{12}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial z_p} \times \frac{\partial z_p}{\partial w^1_{12}}$$

$\frac{\partial L}{\partial \hat{y}} = -(y-\hat{y})$; $\frac{\partial \hat{y}}{\partial z_f} = 1$; $\frac{\partial z_f}{\partial o_{12}} = w^2_{21}$; $\frac{\partial o_{12}}{\partial z_p} = o_{12}(1-o_{12})$; $\frac{\partial z_p}{\partial w^1_{12}} = x_{i1}$

$$\frac{\partial L}{\partial w^1_{12}} = -(y-\hat{y}) w^2_{21} o_{12}(1-o_{12}) x_{i1} \quad (3)$$

$$\frac{\partial L}{\partial w^1_{22}} = -(y-\hat{y}) w^2_{21} o_{12}(1-o_{12}) x_{i2}$$

$$\frac{\partial L}{\partial b_{12}} = -(y-\hat{y}) w^2_{21} o_{12}(1-o_{2})$$

Backpropagation → The why?

The intuition behind the algorithm ✓

Backpropagation Algorithm

epochs = 5

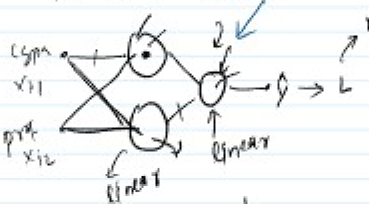
```
for i in range(epochs):                      # epochs x rows
    for j in range(X.shape[0]):
        → select 1 row (random)
        → Predict (using Forward prop)        ŷ  y
        → Calculate loss (using Loss function → mse)   → bce
        → Update weights and bias using GD
            └→ W_n = W_0 - η ∂L/∂W   rule
                 (magic)
    → Calculate avg loss for the epoch
      L_avg
```

→ Loss function is a function of all trainable parameters

data → target

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

cgpa | profile | lpa

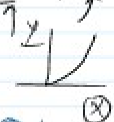


$mse = (y - \hat{y})^2$ variable

$L = (y - \hat{y})^2$ → mathematical function (constant)

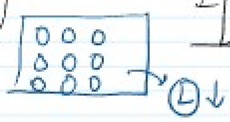
$L(?)$ → $L(\hat{y}) =$

$y = f(x)$ 1 thing



$L(w, b) \downarrow$

9 things → w, b



$\hat{y} = W_{11}^2 O_{11} + W_{21}^2 O_{12} + b_{21}$

$\hat{y} = W_{11}^2 [W_{11}^1 x_{i1} + W_{21}^1 x_{i2} + b_{11}] + W_{21}^2 [W_{12}^1 x_{i1} + W_{22}^1 x_{i2} + b_{12}] + b_{12}$

→ Concept of Gradient → Gradient Descent

└→ fancy word for derivative

$y = f(x) = x^2 + x$

$\frac{dy}{dx} = \frac{d}{dx}(fx) = \frac{d}{dx}(x^2 + x) = 2x + 1$

derivative

$y \to x \to$ derivative $\frac{d}{dx}$

$z = f(x, y) = x^2 + y^2$

$\frac{\partial z}{\partial x} = 2x$ $\frac{\partial z}{\partial y} = 2y$

gradient $\frac{\partial}{\partial x}$

gradient complex → $L(W_{11}^1, W_{12}^1 \ldots W_{21}^2, b_{21}, b_{11}, b_{12})$

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

9 params

└→ 9D function → 9 diff slopes wrt each dim

3D → $z = x^2 + y^2$ $z = f(x, y)$

$y = f(x) = x^2$

→ Concept of Derivative → Derivative at a point

↑ intuition

$\frac{dy}{dx}$ → (y) func x → (x)

rate of change ↑

$\frac{dy}{dx} = \frac{2}{\text{mag}}$ (+ve) ↳ sign

$\frac{dy}{dx} = -2$ (-ve)

$x \rightarrow$

$y \rightarrow$

derivative

$y = x^2 + 2x$ $\quad x=5$ → derivative

$\frac{dy}{dx} = (2x+1)_{x=5}$

$\left(\frac{dy}{dx}\right)_{x=5} = 11$ → slope

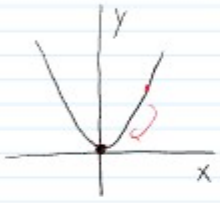
$\frac{\partial L}{\partial w_{11}^1}$

w_{11}^1 — 1 unit

$L \rightarrow ?$ (mag) (sign)

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

→ the concept of minima



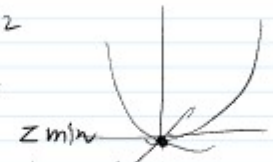
$y = x^2$

$\frac{dy}{dx} = 2x = 0$

$x = 0$

$z = x^2 + y^2$

$(x, y) \rightarrow z$ min



$\frac{\partial z}{\partial x} = 0$ $x \rightarrow$ $\quad 2x = 0 \Rightarrow x = 0$

$\frac{\partial z}{\partial y} = 0$ $y \rightarrow$ $\quad 2y = 0 \Rightarrow y = 0$

L (q param) → (w, b) → min

L ↓↑

$\frac{\partial L}{\partial w_{11}^1}$ ..... $\frac{\partial L}{\partial b_{12}} = 0$

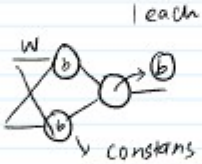
q dim → minima

→ Backprop Intuition

$\eta = 1$

$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$ ← q step

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



each parameters

$L = L(w, b)$ → q → constant

$L(b_{21})$

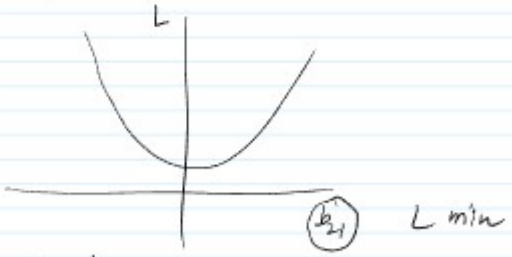
$L = (y - \hat{y})^2$ → $L(b_{21})$; $\hat{y} = (b_{21})$

$L = b_{21}^2$

$b_{21} = b_{21} + \frac{\partial L}{\partial b_{21}}$; $b_{21} = 5$

+ve

-ve



$\frac{\partial L}{\partial b_{21}}$ Smart

L derivative wrt b_{21}

↑ unit → $L \downarrow$ → $b_{21} \downarrow$

$b_{21} \downarrow$

$L \downarrow$ → $\frac{\partial L}{\partial b_{21}} = +ve$ → $b_{21} \uparrow$ $L \uparrow$

$b_{21} \uparrow$

$\frac{\partial L}{\partial b_{21}} = -ve$ → $b_{21} \uparrow$ $L \downarrow$

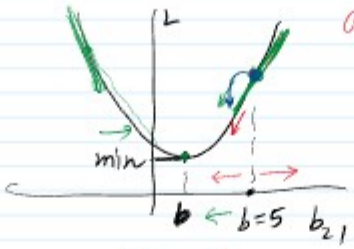
$b_{21} \uparrow$

$b_{21} = b_{21} - \frac{\partial L}{\partial b_{21}}$

game ... -ve of the gradient

game derivative (b_{21}) — -ve of the gradient

$$b_{21} = b_{21} - \frac{\partial L}{\partial b_{21}}$$



algo

$b = -5$

$\frac{\partial L}{\partial b}$ = slope

+ve

$\frac{\partial L}{\partial b}$ = -ve

-ve

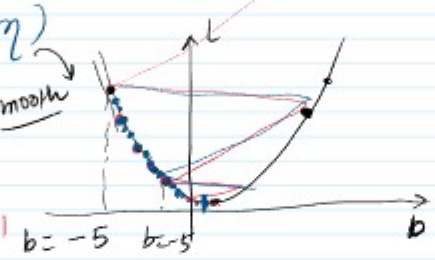
slope (gradient) -ve

Slope

$$w_n = w_0 - \frac{\partial L}{\partial w} \quad \eta \qquad \boxed{w_n = w_0 - \eta \frac{\partial L}{\partial w}}$$

→ Effect of Learning Rate (η)

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$



Smooth

0.1/0.01

train

$\eta = 1$

η = parameter → 0.01

$\eta = 0.00001$

$\frac{\partial L}{\partial b} = -50$

$b = -5 - (-50) = 45$

$b = 45$

$\frac{\partial L}{\partial b} = 50$

$b = 45 - 50$

$b = -5 \quad \frac{\partial L}{\partial b} = -50 \quad \eta = 0.01$

$b = -5 + (0.01 \times 50) = 0.5$

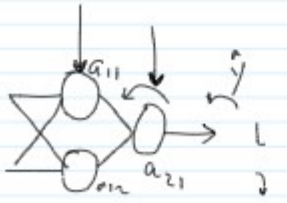
$= -4.5$

→ What is convergence?

$w_n = w_0 - \eta \frac{\partial L}{\partial w} \approx 0$

$w_n = w_0$

while → con → for 1000 epoch → 100



$i = 1, 0$

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

activation = ②

$\hat{y}(1 - \hat{y})\, o_{11}$

$-(y - \hat{y})\, o_{11}$

$-y \log(a_{21}) - (1-y) \log(1 - a_{21})$

$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial a_{21}} \times \frac{\partial a_{21}}{\partial w_{11}^2}$$

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

$\frac{\partial L}{\partial} =$

$$\frac{\partial L}{\partial w^2_{21}} =$$

$$\frac{\partial L}{\partial b_{21}} =$$

curr_activa = $a[1]$

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

$$\begin{cases} -(y-\hat{y})\ a_{21}(1-a_{21})\ a_{11} \\ -(y-\hat{y})\ a_{21}(1-a_{21})\ a_{12} \\ -(y-\hat{y})\ a_{21}(1-a_{21}) \end{cases}$$

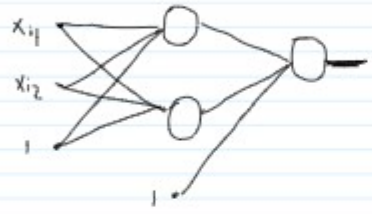
$$\text{delta} = (y-\hat{y})\ a^{[2]}(1-a^{[2]})$$

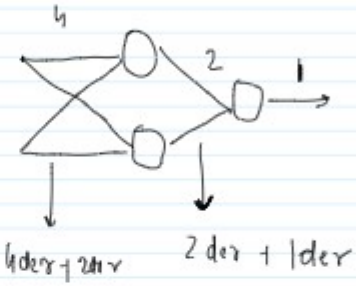
1×1 — 1×2 — 2×1

$w[1]$

1×2

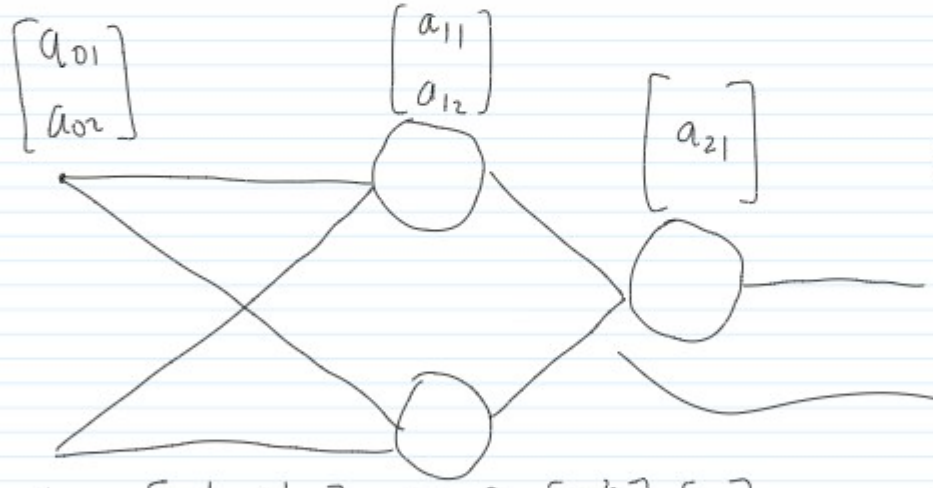
$$-(y-\hat{y})\ a_{21}(1-a_{21}) \times a[1] \begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix}$$





1×1 1×2

1×2 × 2×2



L → L → a_2 → a_1 →

$$\frac{\partial L}{\partial a_2} \times \frac{\partial a_2}{\partial a_1} \frac{\partial a_1}{\partial w}$$

$$\begin{bmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{bmatrix} \quad \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} \quad \begin{bmatrix} w^2_{11} \\ w^2_{12} \end{bmatrix} \quad \begin{bmatrix} b_{21} \end{bmatrix}$$

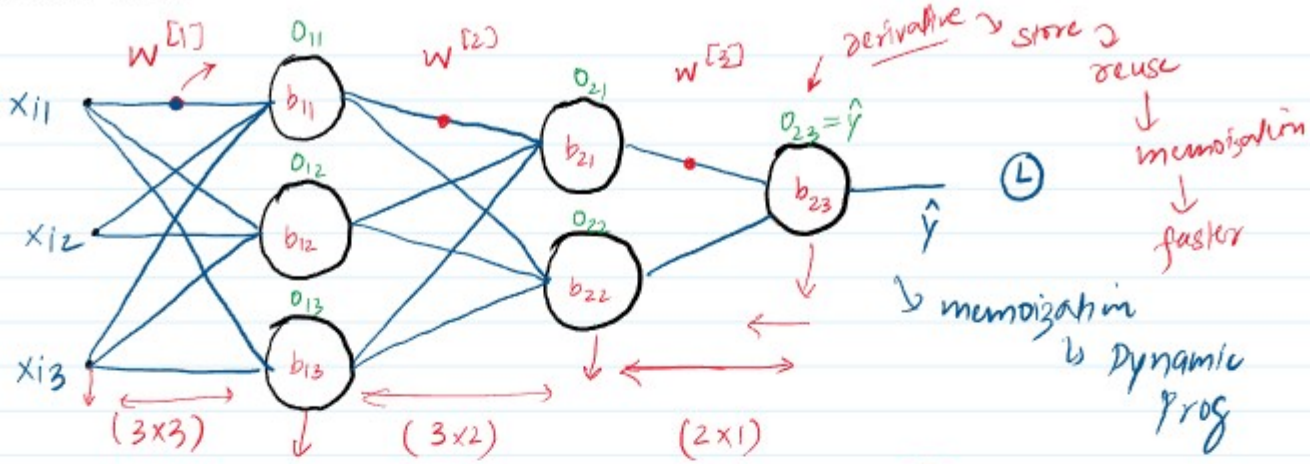
$$\frac{\partial L}{\partial w} = \left[\frac{\partial L}{\partial a_2}\right] \times \left[\frac{\partial a_2}{\partial w}\right]$$

$$\begin{bmatrix} \partial L/\partial w^1_{11} & \partial L/\partial w^1_{12} \\ \partial L/\partial w^1_{21} & \partial L/\partial w^1_{22} \end{bmatrix} \qquad \begin{bmatrix} \partial L/\partial w^1_{11} \\ \partial L/\partial w^1_{12} \end{bmatrix}$$

MLP Memoization

Thursday, April 7, 2022 7:45 AM

backprop update → w / b) deriv



derivative → store → reuse → memoization → faster

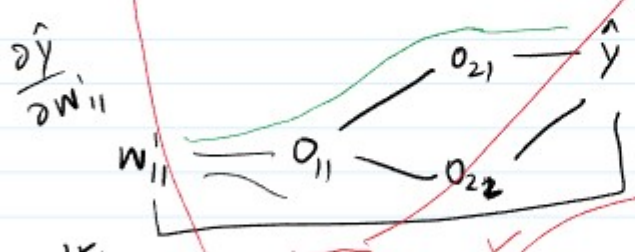
↳ memoization ↳ Dynamic Prog

Chain rule

$L \to \hat{y} \to o_{21} \to o_{11} \to w^1_{11}$

$$\frac{\partial L}{\partial w^3_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w^3_{11}} \qquad \frac{\partial L}{\partial w^2_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial w^2_{11}}$$

$$x \to f(x), g(x) \to h(f(x), g(x)) \quad \frac{\partial h}{\partial x} = \left[\frac{\partial h}{\partial f(x)} \times \frac{\partial f(x)}{\partial x}\right] + \left[\frac{\partial h}{\partial g(x)} \times \frac{\partial g(x)}{\partial x}\right]$$



$$\left[\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}}\left[\frac{\partial \hat{y}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w^1_{11}} + \frac{\partial \hat{y}}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w^1_{11}}\right]\right]$$

Backprop → Chain diff (rule) + Memorization

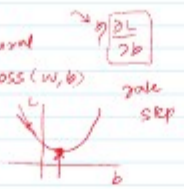
Keras TF — maths — (13)

Gradient Descent

Thursday, April 7, 2022 7:46 AM

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks.

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.



Back propagation Algorithm

epochs = 5 ↙ stochastic

for i in range(epochs):

  for j in range(X.shape[0]): → gradient descent

    → select 1 row (random)

    → Predict (using Forward prop)

    → Calculate loss (using loss function → mse)

    → Update weights and bias using GD

$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

→ Calculate avg loss for the epoch

Loss

50 points
1 epoch
↳ 50 times
w, b update

Batch GD
epoch = 5
↳ 50 points → predict J (current weight)
↑ dot product
y_hat = np.dot(X, w) + b
↳ 50 predict
y = 50 actual obs.
y y_hat J loss
↘ $-\sum_{i=1}^{50}$

Batch / Stochastic / mini batch

1 single (w, b)

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

$\frac{\partial L}{\partial w}$ derivative → 3 types
accuracy → time → tradeoff.

Batch GD (Vanilla GD)

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```

entire dataset J update

epochs = # of updates

Stochastic GD

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

50 × 10 = 500

epoch → 10 (50 rows) 50 rows
for i in range(10):
  → shuffle
  for i in range(X.shape[0]) 10 × 50 = 500 update
    ↳ 1 random point → bias eliminate
    ↳ y_hat → forward
    ↳ loss
    ↳ w, b update → $w_n = w_0 - \eta \frac{\partial L}{\partial w}$
avg loss print → for the epoch

5 epochs
↳ 5 times
↓
w, b update

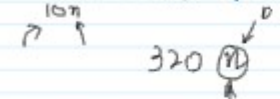
Batch GD → 1 loop

epoch = 10 (epoch) rows
for in range(10): (loop) → total → 10 times w, b update
y_hat = np.dot(X, w) + b
50 values dot → smart replacement → loops
Y = 50 values ↳ vectorization ↘ faster loop

frequency of weight update higher

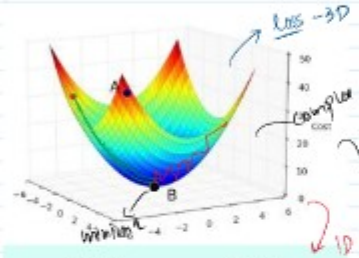
50 values dot → smart replacement → loop
↳ vectorization → faster loop
Y = 50 values Optimized !
Y_hat, Y → loss
↺ w, b update $w_n = w_0 - \eta \frac{\partial L}{\partial w}$
→ loss

Which is faster (given same no. of epochs)

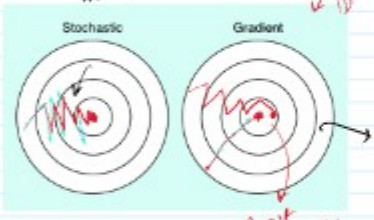


Which is the faster to converge (given some # epochs)
↑
5

Loss function → 2 Algo



Loss → 3D
Stochastic → random → point → updates
Batch → dataset → all point → 1 update



1D
Spiky SGD useful
Yes No
SGD
BGD
SGD → help the algo to move out of local minima

Exact solution
Approximate diff
Vectorization → faster → loops → big dataset
np.dot (X, W) + b
RAM
X → dataset of 10 crore

Mini Batch Gradient Descent ← use
BGD ↔ SGD Best of both worlds
SGD → BGD

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for batch in get_batches(data, batch_size=50):
        params_grad = evaluate_gradient(loss_function, batch, params)
        params = params - learning_rate * params_grad
```

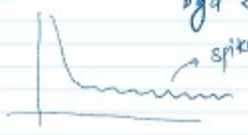
320 rows in every epoch
10 batches
batches → 32
10 update

$\frac{n}{x}$ = # of batches
Keras batch_size = x
updates/epoch

for i in epochs →
  for j in num of batch
    1 batch
    np.dot → y_pred (vector)
    → loss
    ↳ update

bgd > mbgd > sgd
con
bgd < mbgd < sgd

MBGD



spiky SGD → Vectorization → smaller batch

→ Why batch_size is provided in multiple of(2)?

2, 4, 8, 32, 64, 128, 256

10, 15

RAM effective → binary

RAM → optimization → cores

faster

→ What if batch_size doesn't divide # rows properly

e.g # of rows $n = 400$

batch_size = 150

# of batch = $\frac{400}{150} = 2.66$

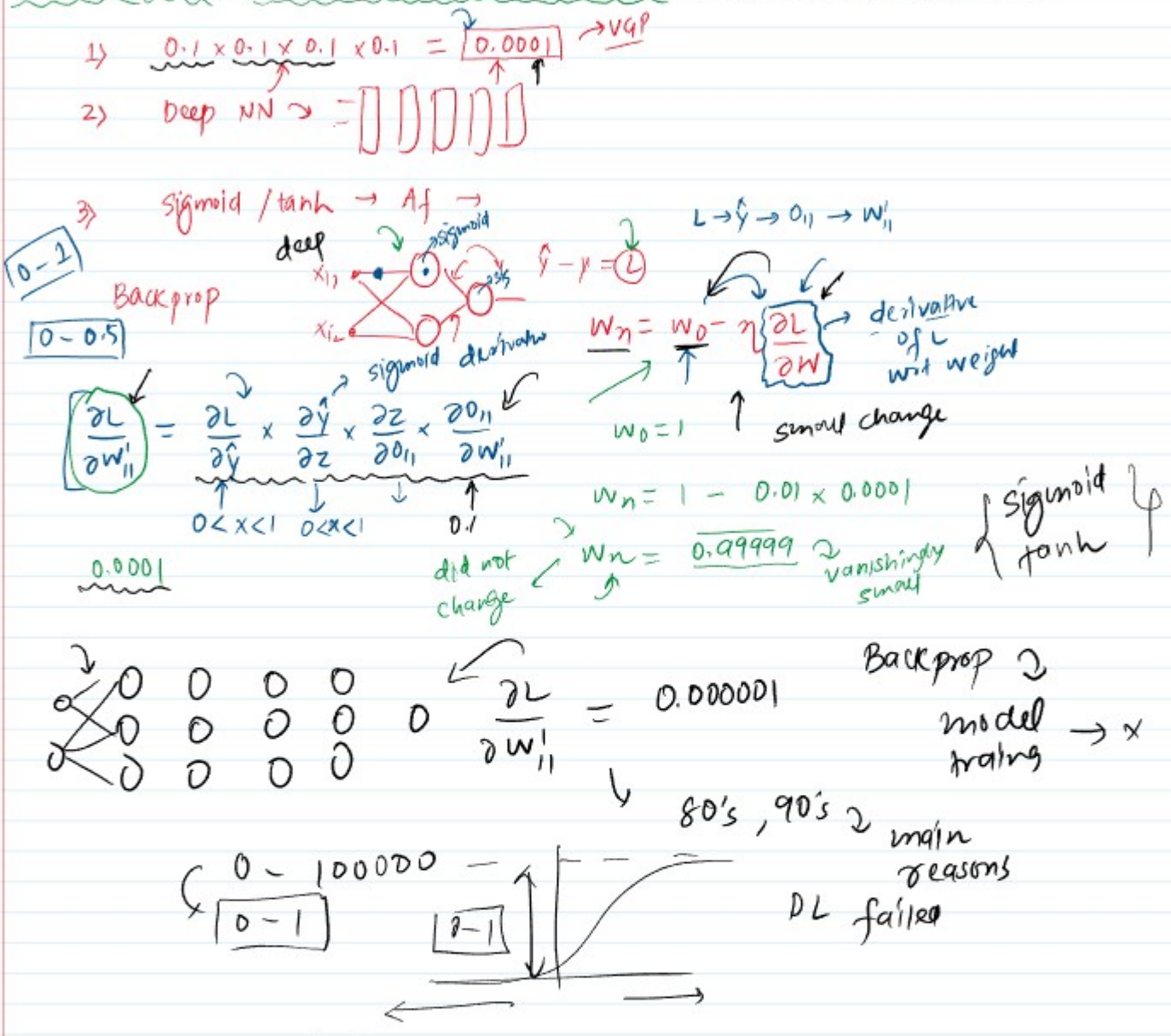
150, 150, left 100

1 batch, 2 batch, 3rd batch

Vanishing Gradient Problem

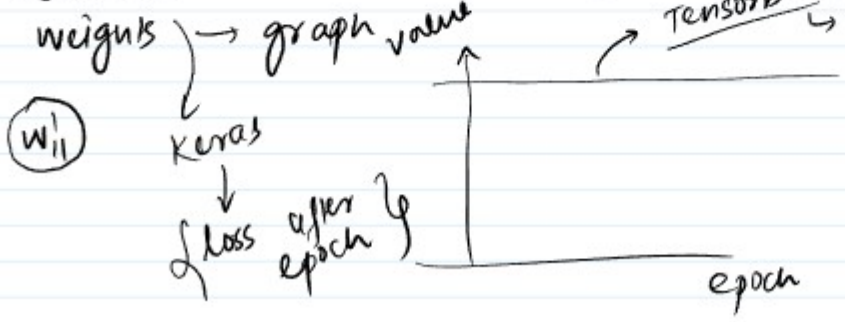
Thursday, April 7, 2022 7:45 AM

In machine learning, the **vanishing gradient problem** is encountered when training artificial neural networks with gradient-based learning methods and backpropagation. In such methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training. ~~As one example of the problem cause, traditional activation~~



How to recognize?

1) Loss focus → epoch → no changes → VGP
2) weights → graph value → Tensorboard → callbacks

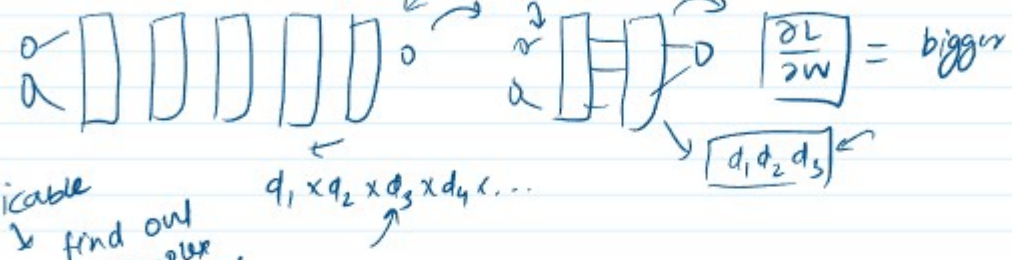


$$w_n = w_0 - \eta \frac{\partial L}{\partial w} \qquad \frac{w_0 - w_n}{\eta} =$$

$$w_n = w_0 - \eta \left[\frac{\partial L}{\partial w}\right] \qquad \left[\frac{w_0 - w_n}{\eta}\right] =$$

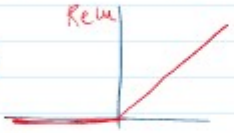
How to handle Vanishing Gradient Problem →

1) Reduce model complexity



Applicable
↳ find out complex patterns

2) Using ReLU Activation functions



max (0, z) → 0, 1

z = 1 × 1 × 1 = 1 → VGP

1000 − 1000 → squish
0 − 1000
0-1

dying relu → derivative $w_n = w_0$

{leaky relu}

3) Proper weight Init → Glorot, xavier

4) Batch norm → layer →

5) Residual Network → CNN → ResNET
↳ building block → ANN

Exploding Gradient Problem → RNN

big number

10, 10, 10, 10 → 10000

$\frac{\partial L}{\partial w_{11}^1} = d_1 \times d_1 \times \phi_1 \times \phi_3$

$w_n = w_0 - \eta \frac{\partial L}{\partial w}$ (0.1, 1000)

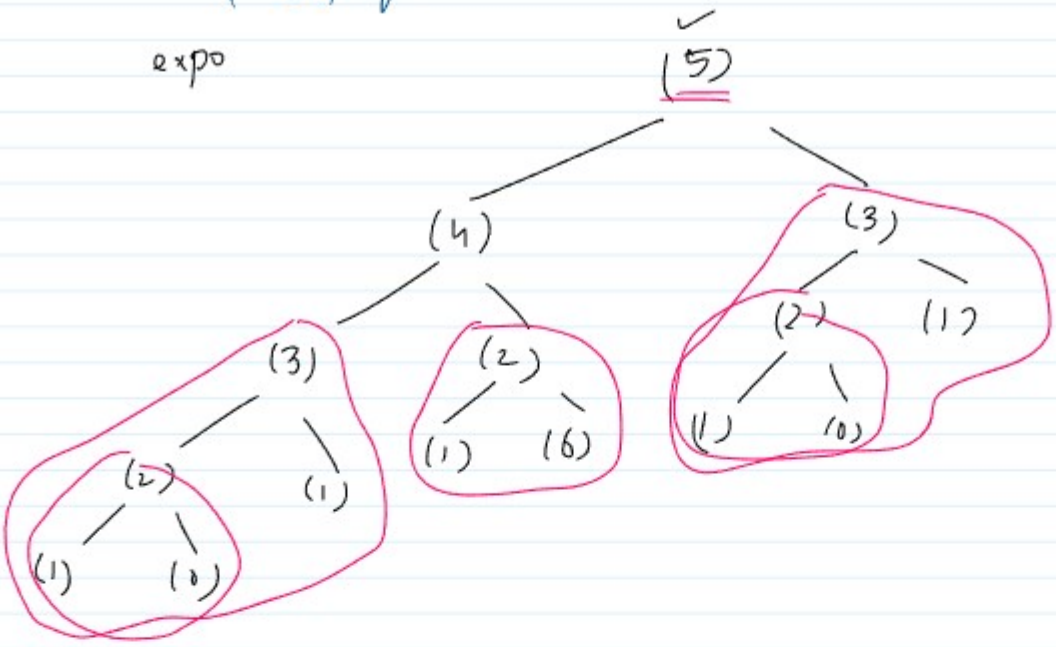
random

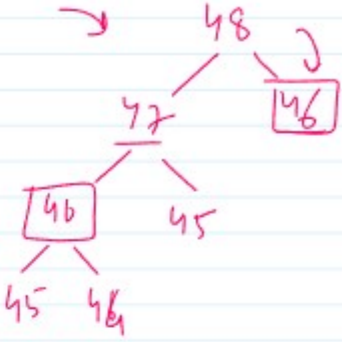
Loss↓ x 1 − 100 → 99 → 1000 →

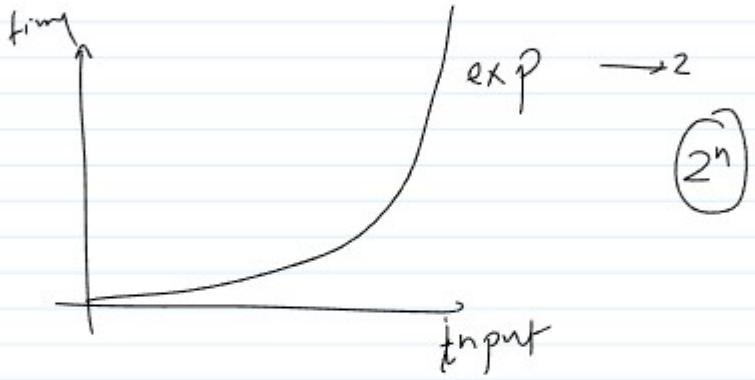
{Gradient clipping}

expo

expo







Saturday, April 9, 2022 10:01 AM

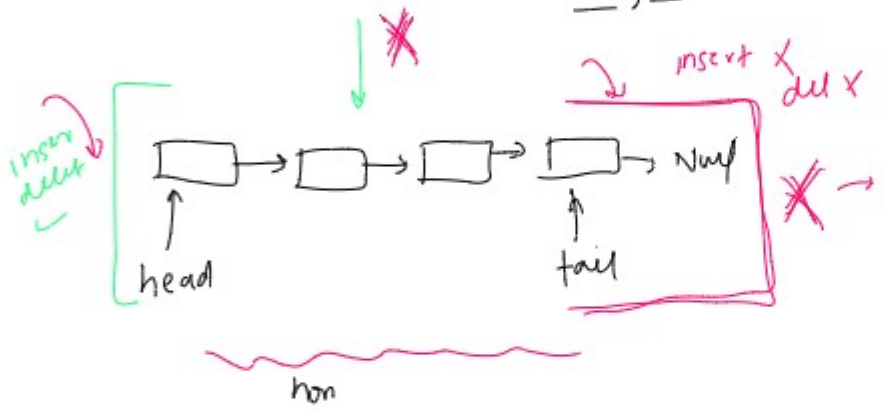
DS Algo → Time complexity
→ Arrays → Dynamic Arrays
→ Linked list

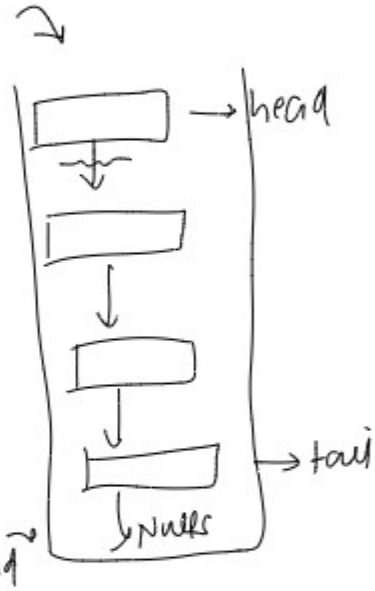
Today → Stacks → LL
→ Arrays

→ 3 → Tuesday
last 3 videos

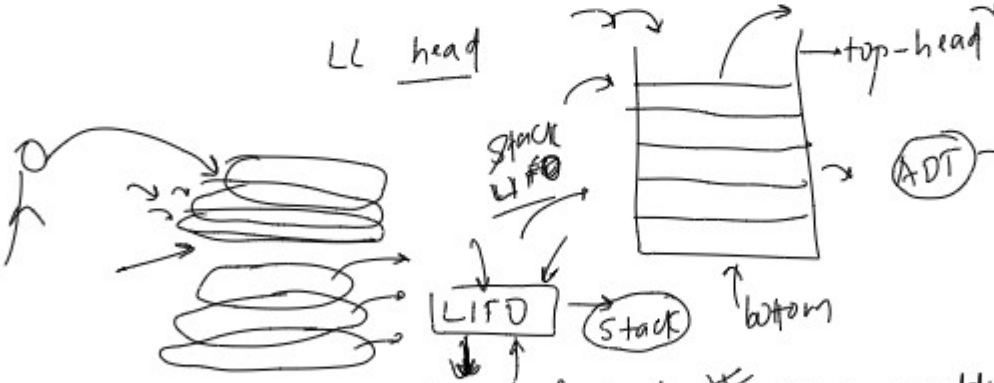
Next → Queues
→ Practice →

→ Stacks → DS
Useful





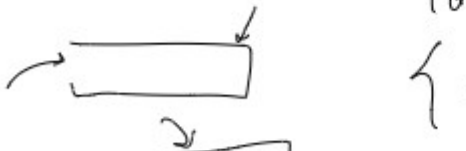
LL head



→ conceptual thing → Normal data type

last in first out

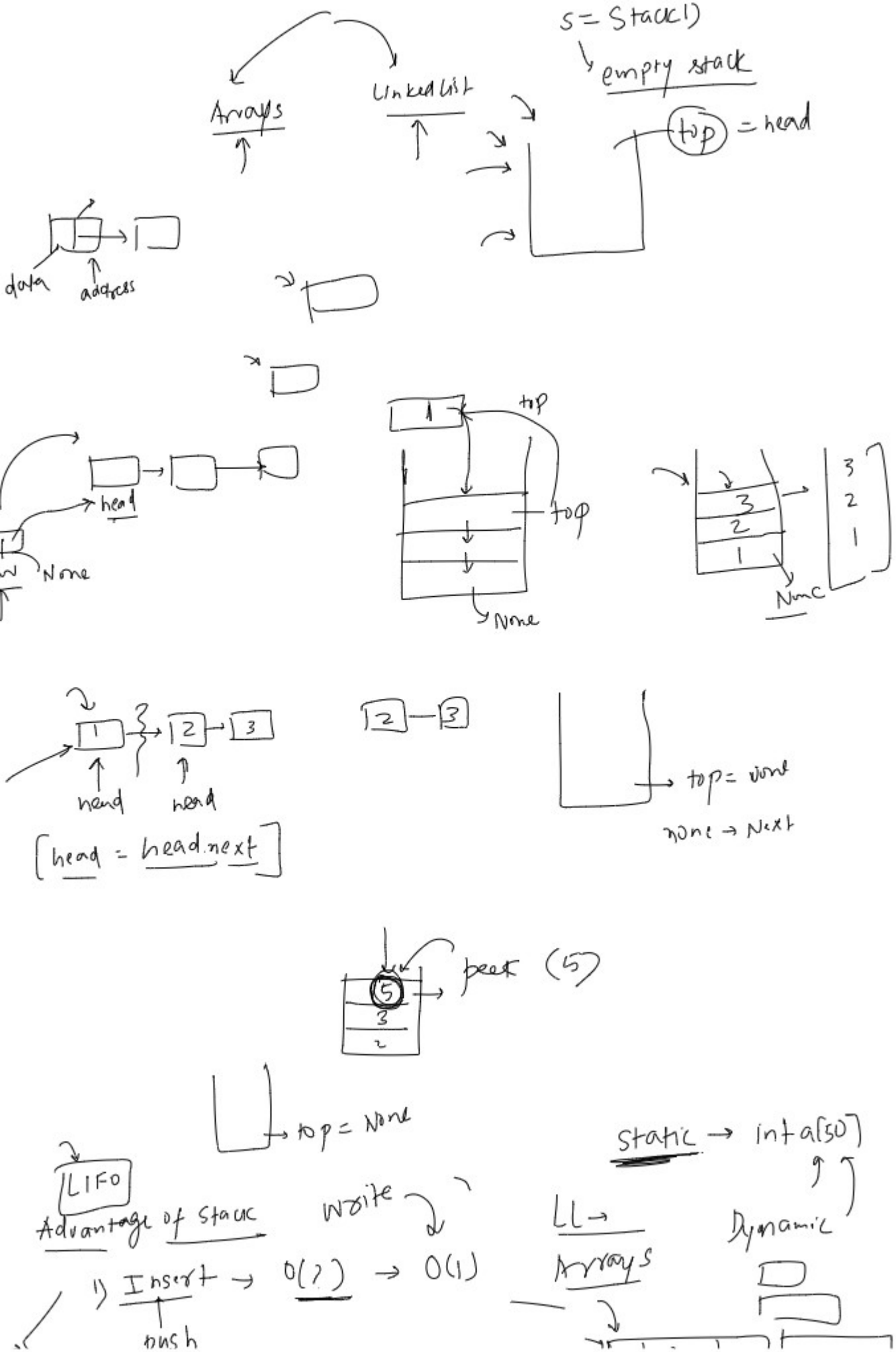
- push → adding new node from top
- is_empty → empty
- pop → removing from top
- peek → top node → value
- size → kitne nodes

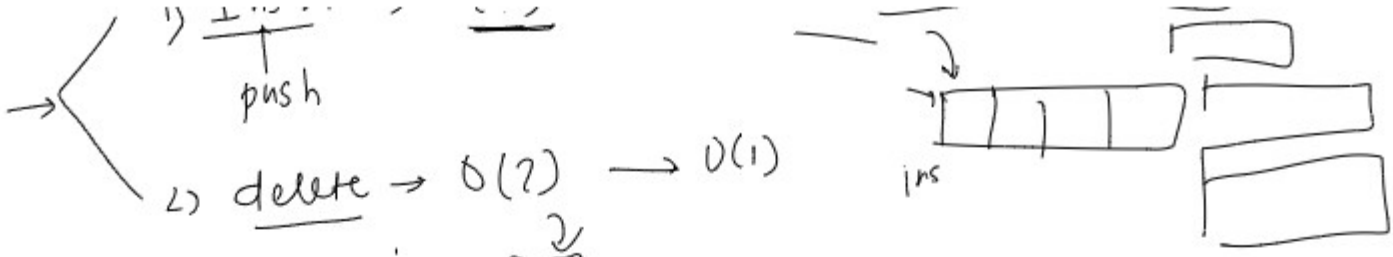


LIFO

- expression matching
- undo redo
- browser back button

S = Stack()


Arrays
Linked list
s = Stack()
empty stack
top = head
data
address
head
new
None
top
None
3
2
1
None
head
head
top = None
None → Next
head = head.next
peek (5)
top = None
LIFO
Advantage of stack
1) Insert → O(?) → O(1)
push
write
LL →
Arrays
static → int a[50]
Dynamic



int a[50] 5

size = 4

top = -1 → 0 → 1 → 2 → 3

1, 2, 3, 4

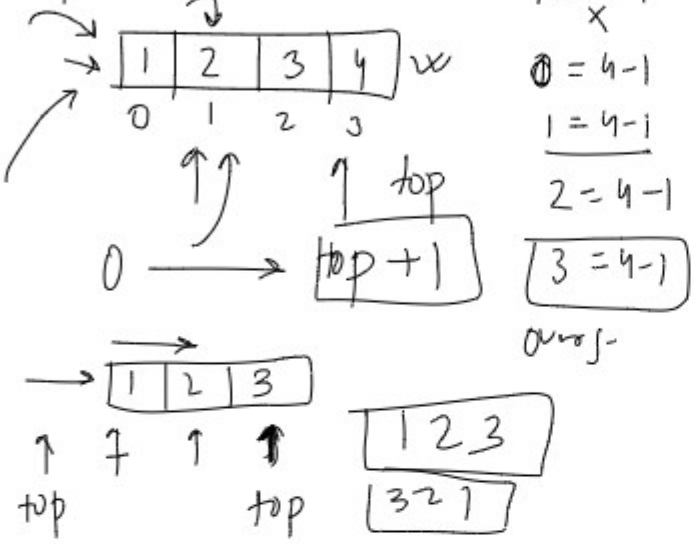
-1 = 4-1 ×

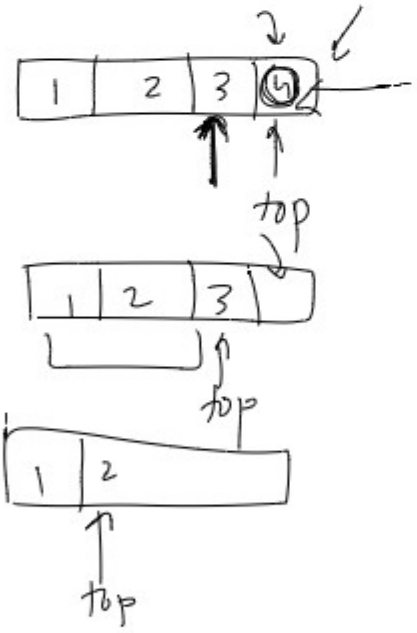
0 = 4-1

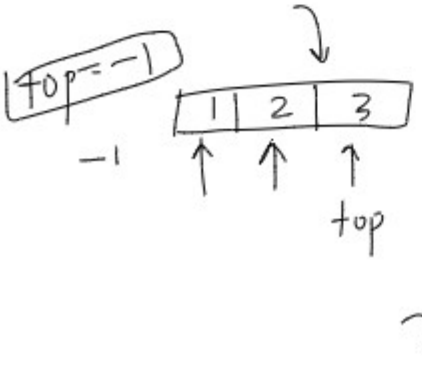
1 = 4-1

2 = 4-1

3 = 4-1 overs-



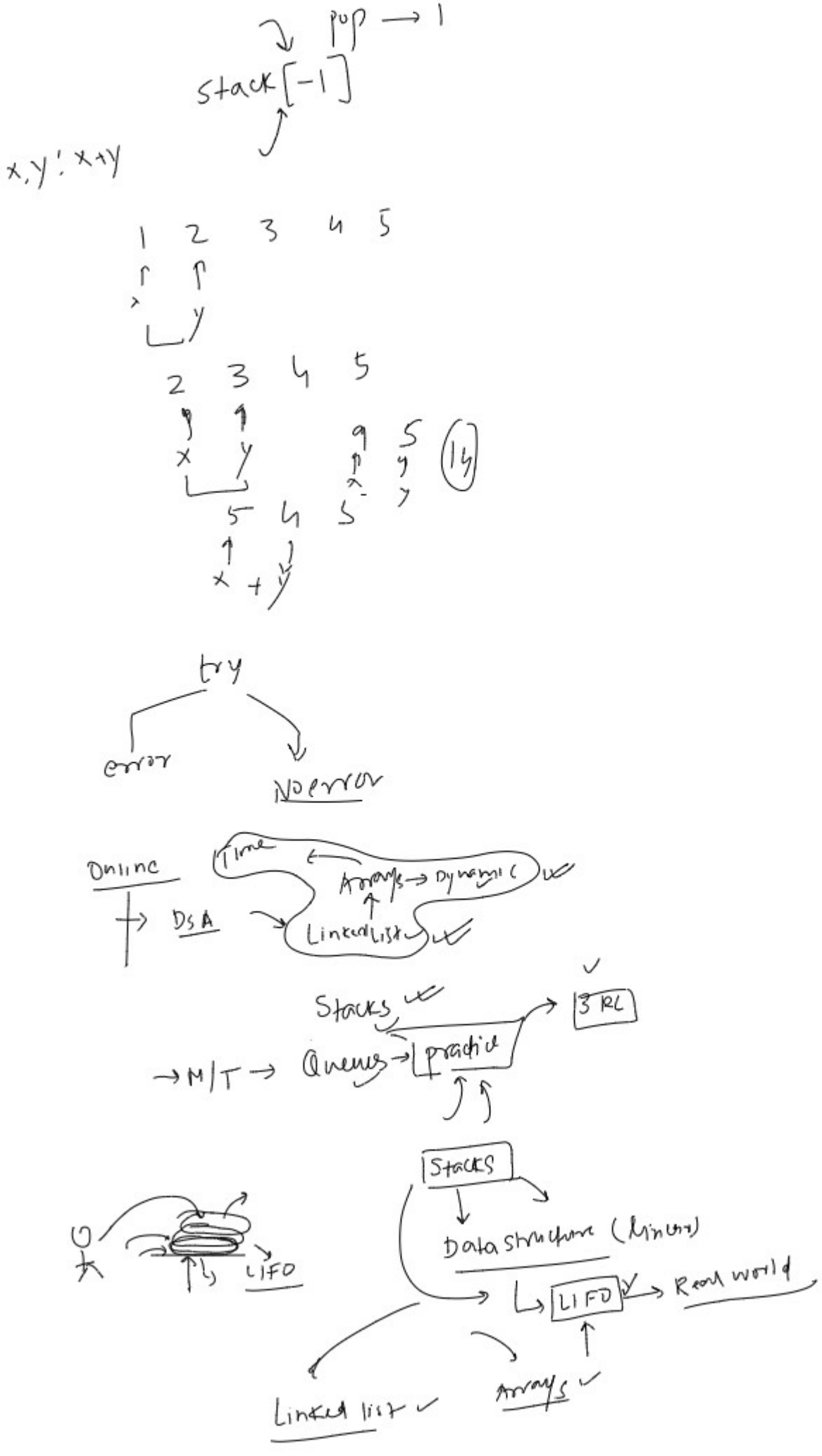


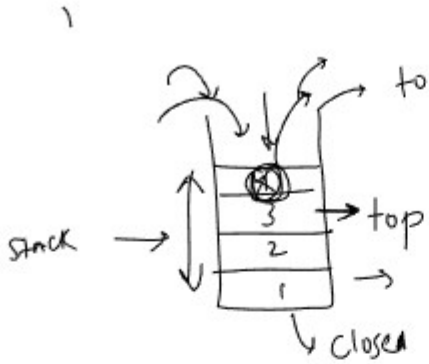


pop → 3

pop → 2

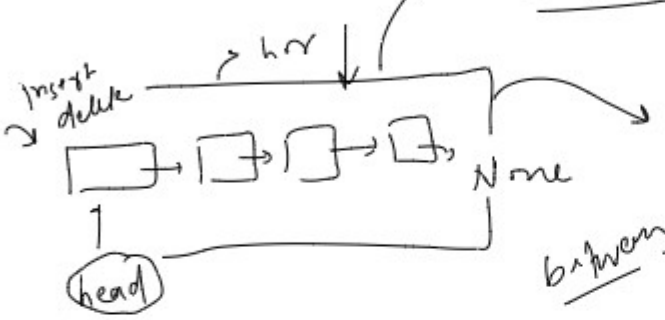
pop → 1

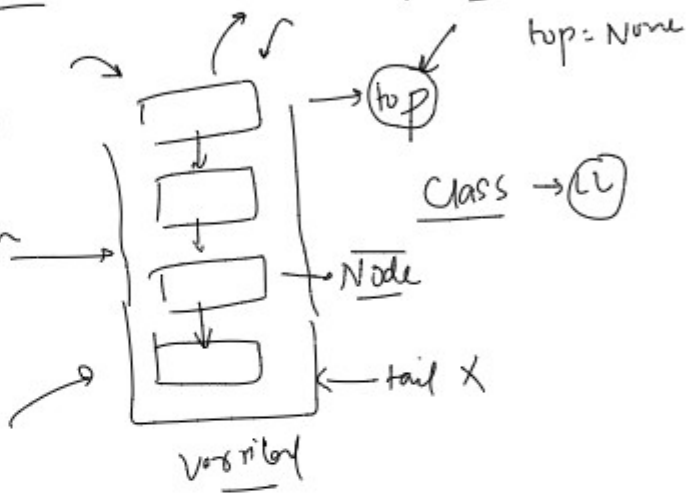

pop → 1
stack[-1]
x, y : x+y
1 2 3 4 5
2 3 4 5
14
try
error
No error
Online
DSA
Time
Arrays → Dynamic
LinkedList
Stacks
3 RL
→ M/T →
Queues
practice
Stacks
Data structure (Linear)
LIFO
LIFO
Real world
Linked list
Arrays



push
pop
peek
is empty
size

Simple



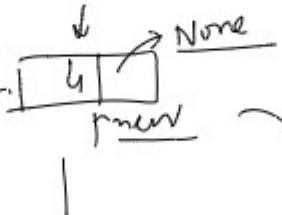


Application

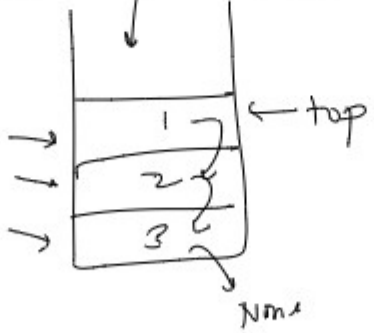
- expression
- undo/redo
- recursion →
- browser → back

LIFO

LL

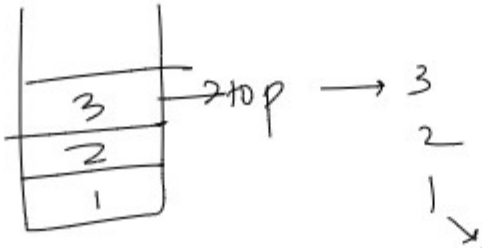


insert → head



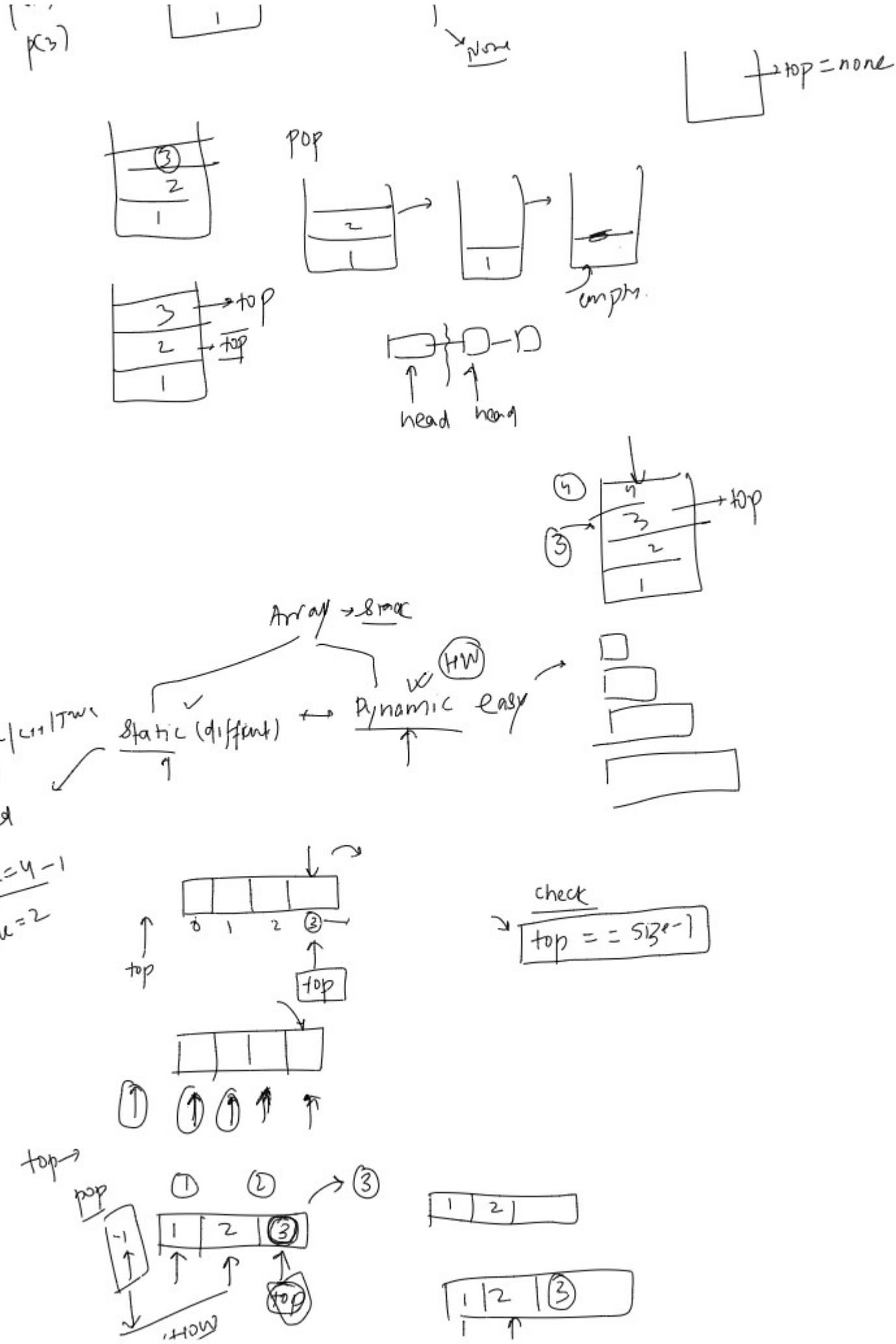
new.next = None
new.next = top
top = new

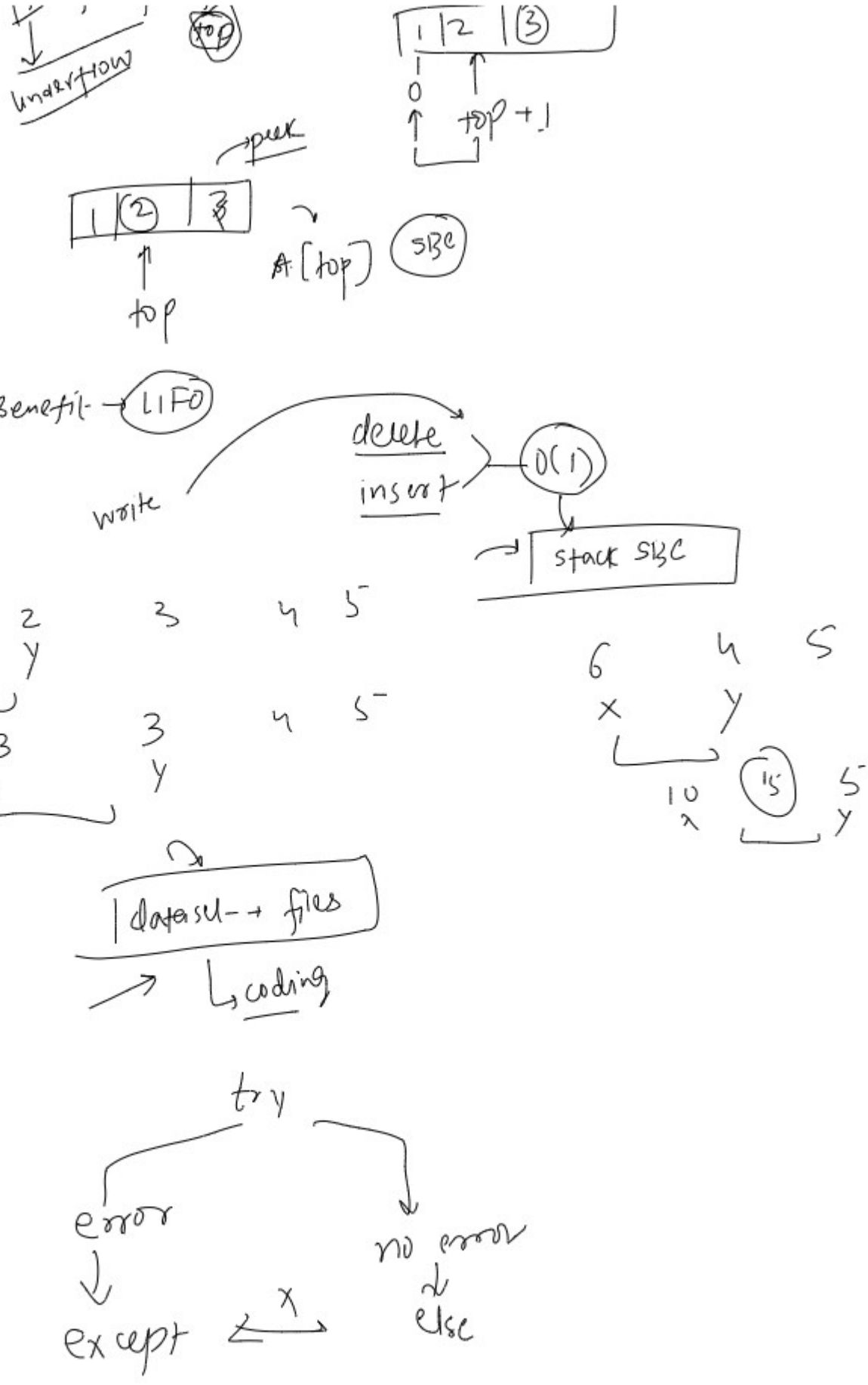
1 2 3



p(1)
p(2)
p(3)

1 1.next = None


p(3)
1
None
top = none
3
2
1
pop
2
1
1
empty
3
2
1
top
top
head
head
4
3
2
1
top
Array → Stack
Static (different)
Dynamic easy
HW
c/c++/Java
fixed
size = 4 - 1
value = 2
0 1 2 3
top
top
check
top == size-1
top →
pop
1 2 3
-1
top
How
1 2
1 2 3


underflow
top
1 2 3
0
top + 1
peek
1 2 3
top
A[top]
SBC
Benefit → LIFO
write
delete
insert
O(1)
stack SBC
1 2 3 4 5
x y
3 3 4 5
x y
6 4 5
x y
10
15
5
y
dataset → files
↳ coding
try
error
except
X
no error
else

How to improve a neural network ✓

29 April 2022 13:51

1. Fine tuning NN hyperparameters ✓
2. By solving problems:
   - → Vanishing / Exploding gradient
   - → Not enough data
   - → Slow training
   - → Overfitting

epochs, # hidden layers, # neurons per layer, learning rate → GD, Optimizer → GD, Batch size, Activation function

128 → 64 → 32, 32 row → update, hidden

Roadmap → 15 videos: ANN → Keras (92.1), GD

DL? perceptron, MLP → forward prop → train MLP / ANN → Backprop → GD

vanishing of weight, weights, gradient, sigmoid, 0.00001, update no meaning

Fine tuning Hyperparameters

- No. of hidden layers ✓
- No. of neurons per layer ✓
- Learning rate
- Optimizer
- Activation function
- Batch size
- Epochs

1 No. of hidden layers

hidden

1 hidden layer → 512 neurons → complex

lot of neuron, 1 h → 512

Representation

fewer neurons, 3 / 50 / 300

3 hidden → 32 neuron

deep neural network

Overfitting

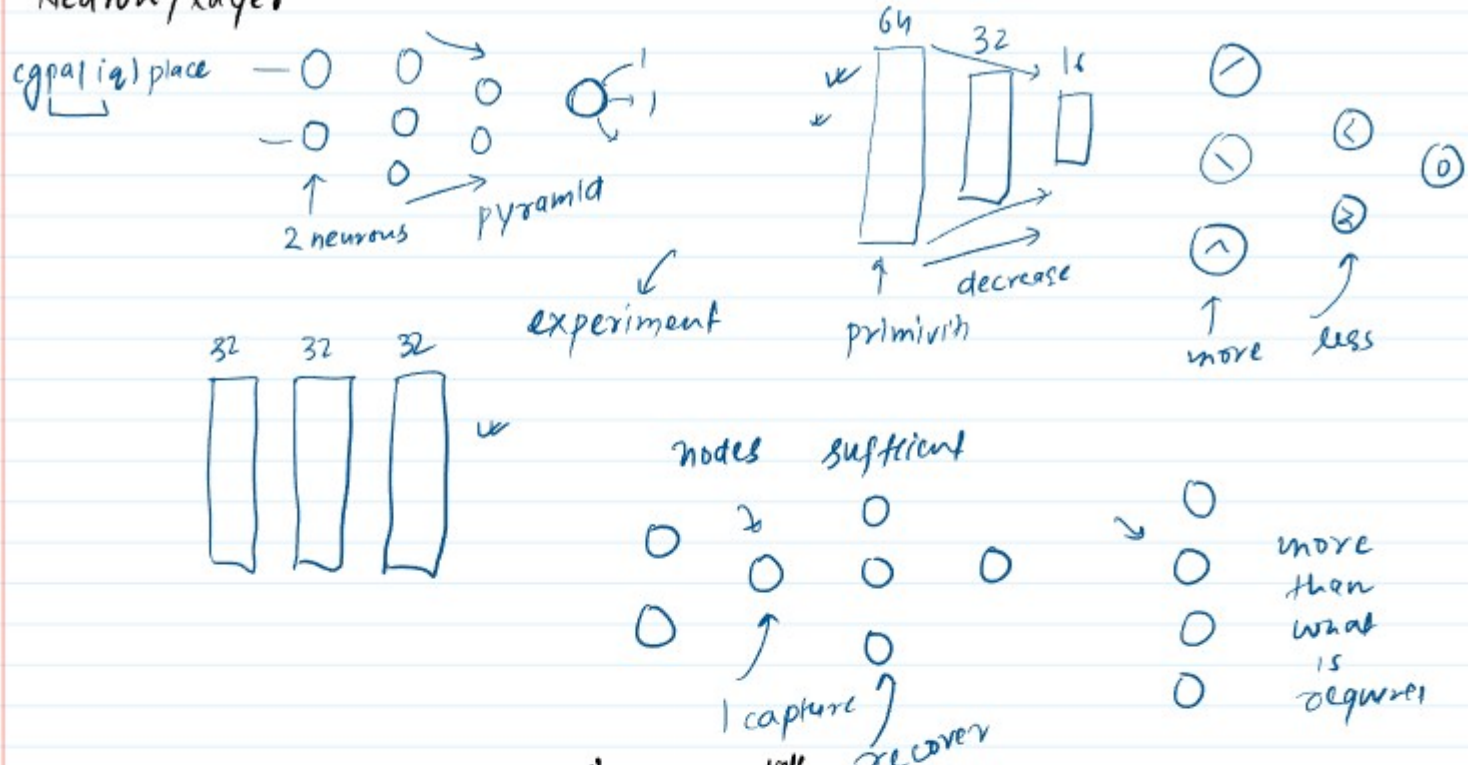
human face

Transfer learning → CNN

reuse

NN ← line, shapes

2. Neuron / layer

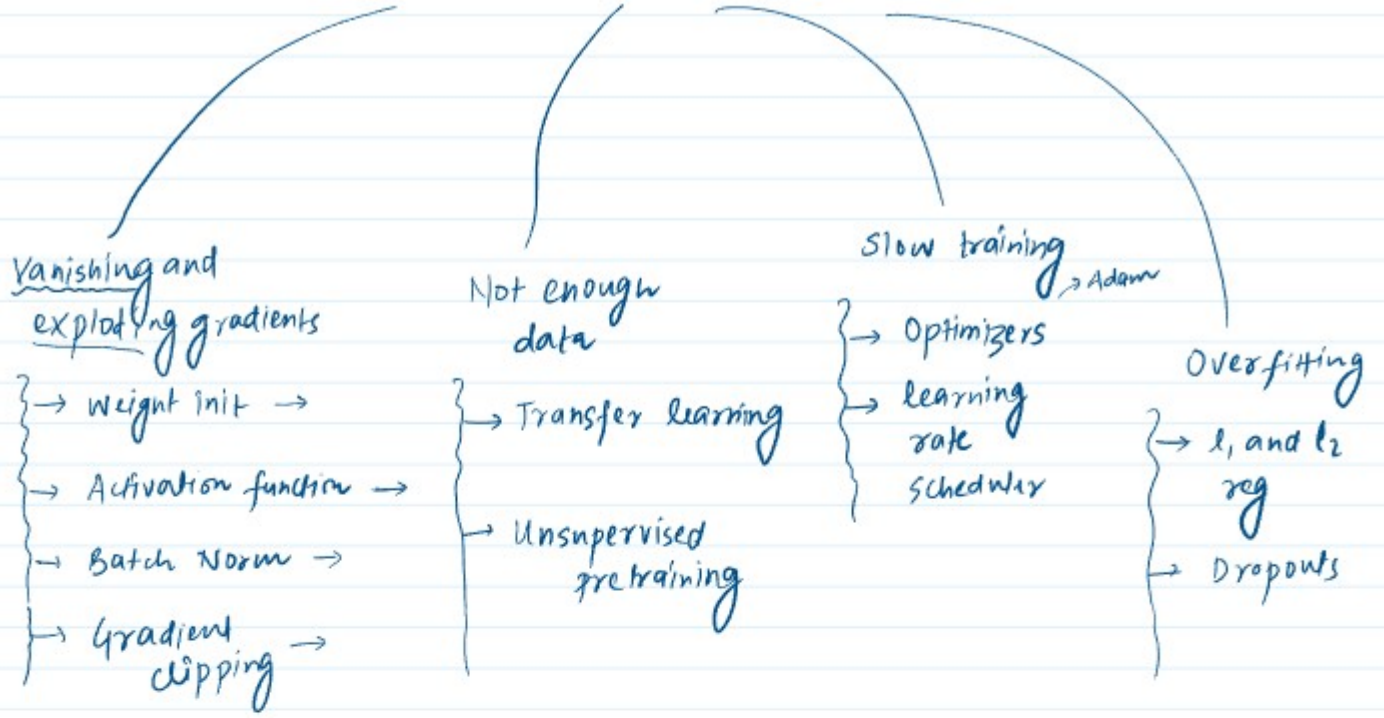


3. Batch size
- Batch → all row → weight update
- Stochastic → 1 row → weight
- Mini Batch → batch → 32 → row → update

hyper parameter

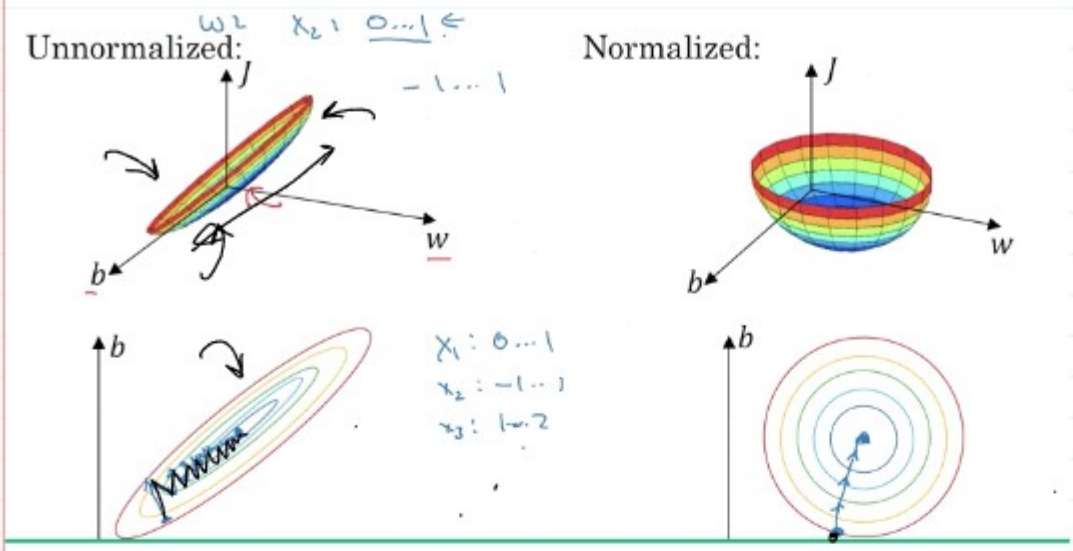
4. Epochs → 100 / 500 / 1000 → early stopping
- Keras → callback
- Keras → stable

Problems with Neural Networks


Vanishing and exploding gradients
→ Weight init →
→ Activation function →
→ Batch Norm →
→ Gradient clipping →
Not enough data
→ Transfer learning
→ Unsupervised pretraining
Slow training
→ Adam
→ Optimizers
→ learning rate schedular
Overfitting
→ l_1 and l_2 reg
→ Dropouts

Feature Scaling

05 May 2022 17:37



(age) x_1 x_2 (salary)

w_1 w_2

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

Standard

$\frac{x_j - \mu}{\sigma}$

Normalize

$\frac{x_j - x_{min}}{x_{max} - x_{min}}$

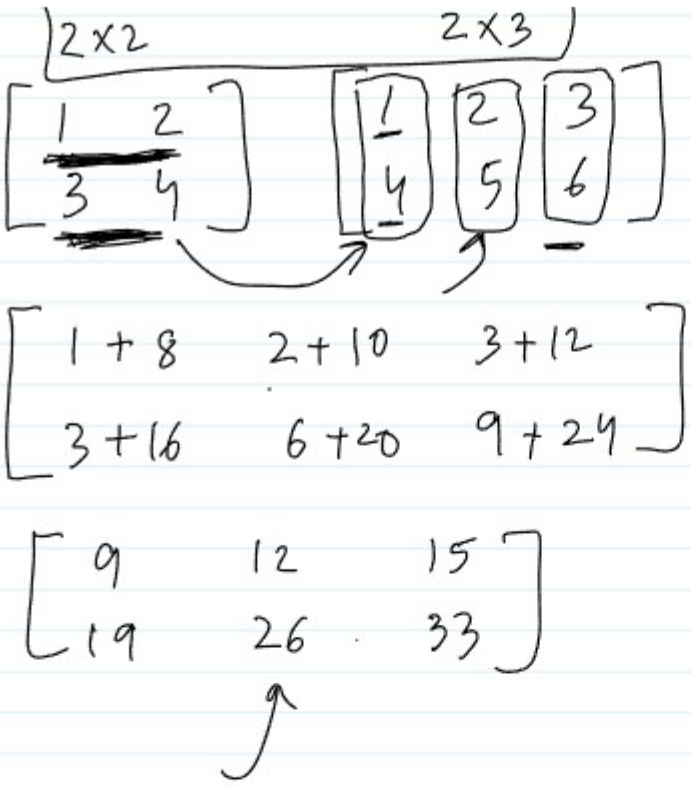
0 to 1

min = 0

max = 1

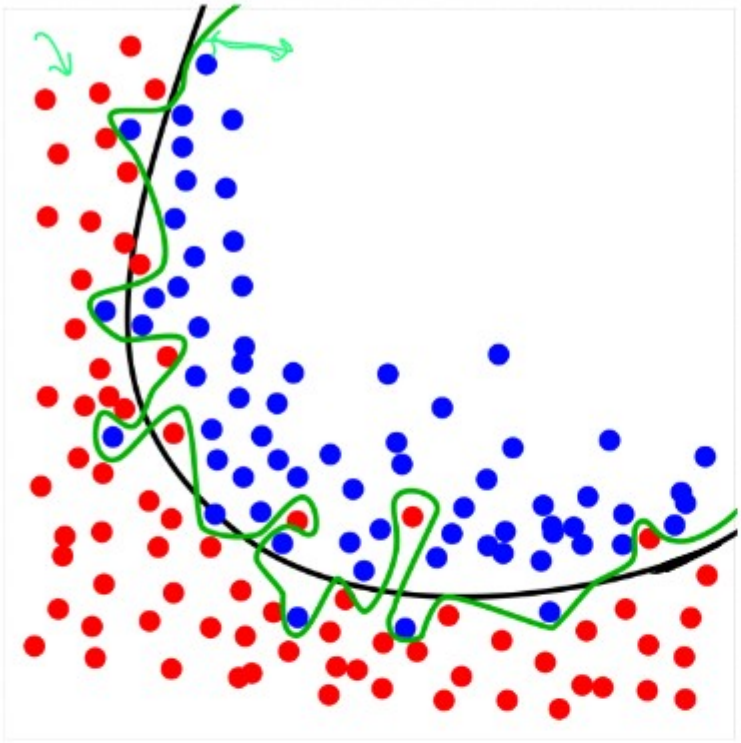
-1 to +1

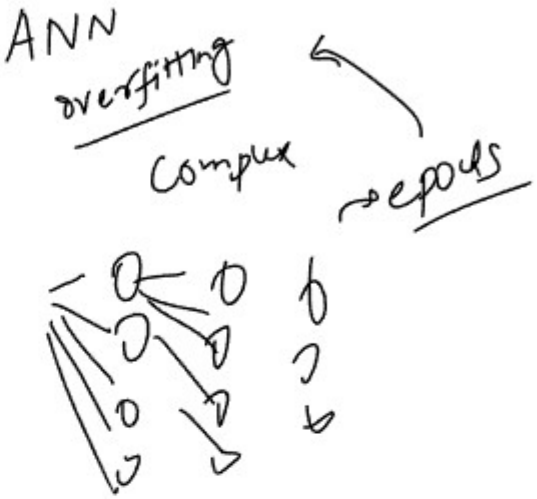
2x2 2x3


2x2
2x3
1 2
3 4
1 2 3
4 5 6
1 + 8 2 + 10 3 + 12
3 + 16 6 + 20 9 + 24
9 12 15
19 26 33

The Problem of Overfitting

12 May 2022 09:32





Possible Solutions

12 May 2022 09:33

1) Add more data

2) Reduce complexity

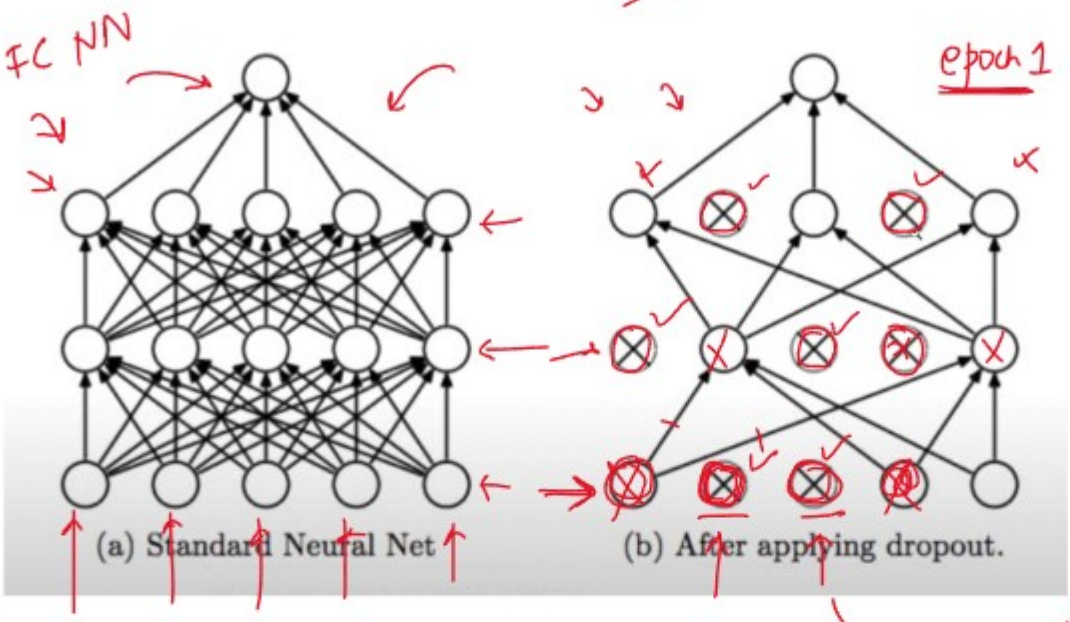
3) Early stopping

4) Regularization → L1 → L2

5) Dropout

The concept of Dropouts

12 May 2022 09:33



Dropout

FC NN

epoch 1

$x_1\ x_2\ x_3\ x_4\ x_5\ y$

input nodes

hidden drop

epoch = 10

epoch 2

epoch 3

2-1. → acc

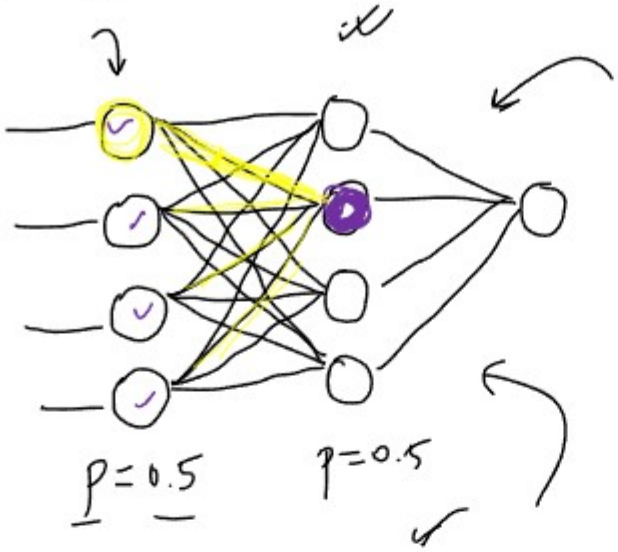
95 → 97

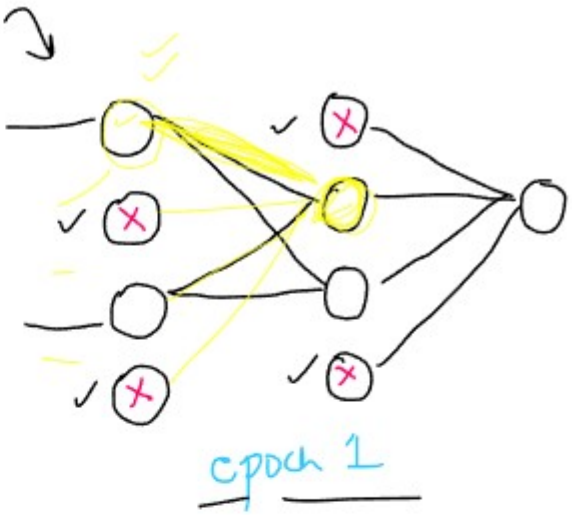
10 epochs

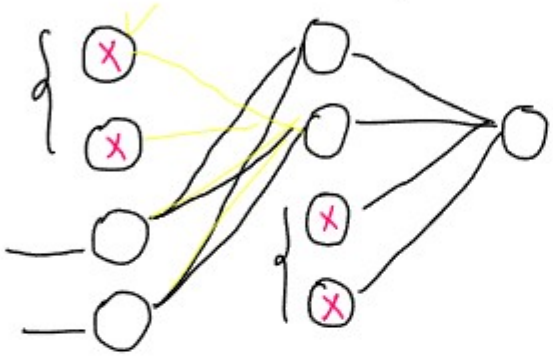
10 diff NNs

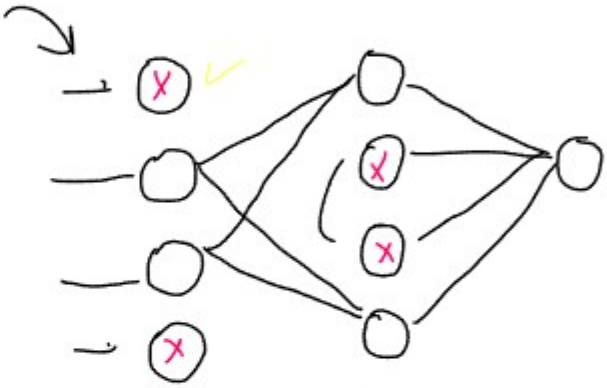
Why this works?

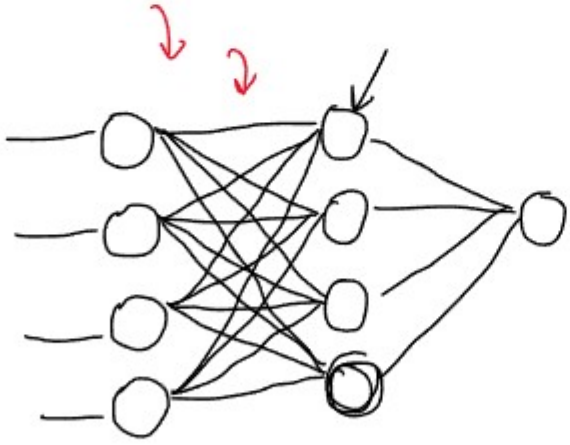
12 May 2022 09:33





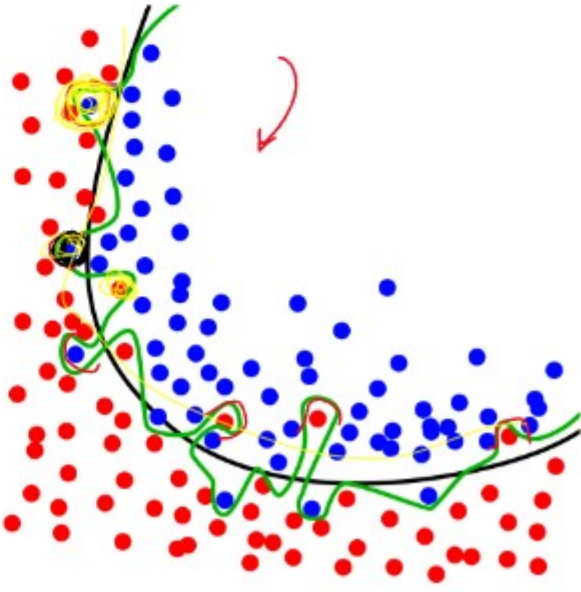






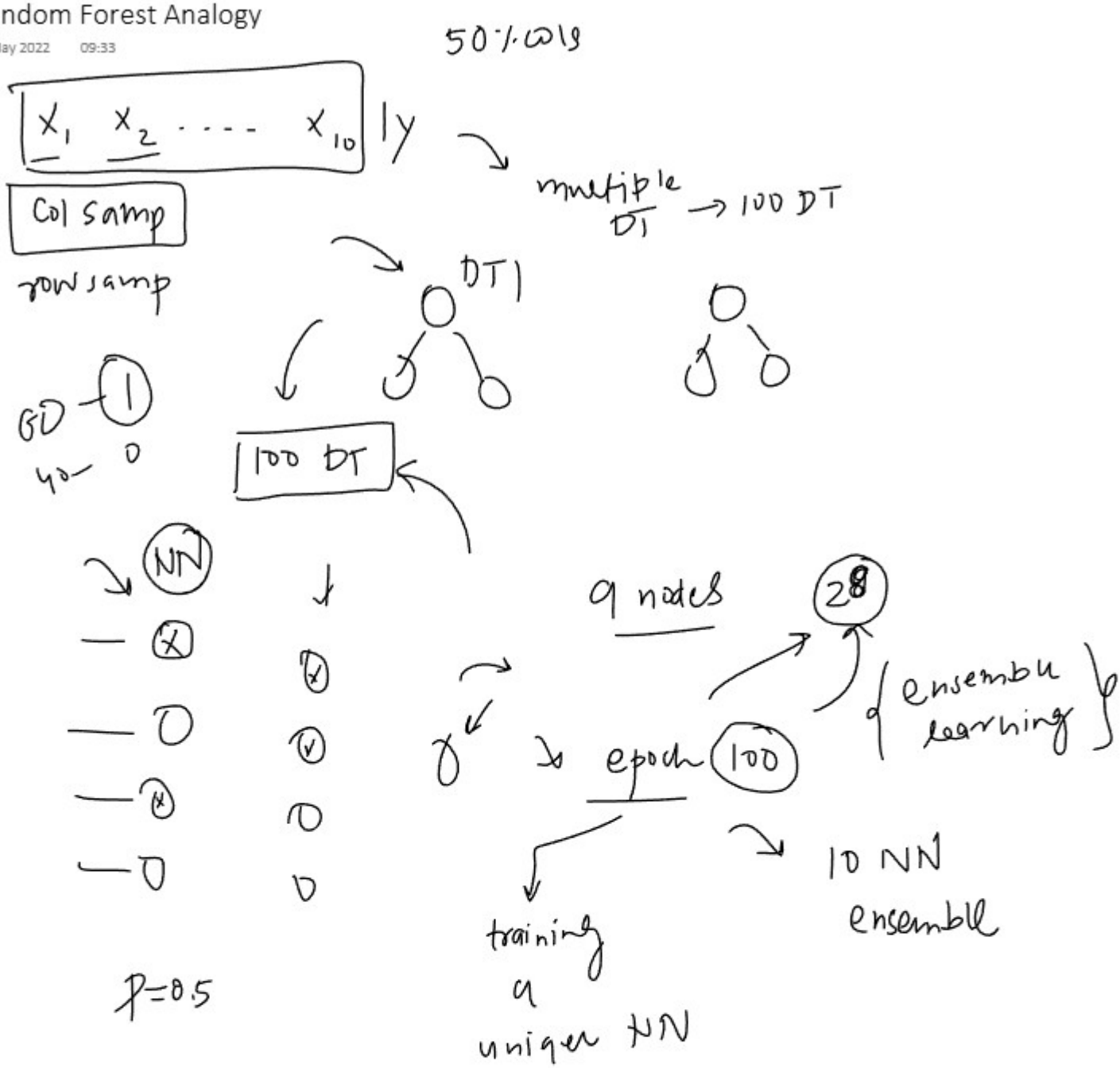
→ Reduce the # nodes

→



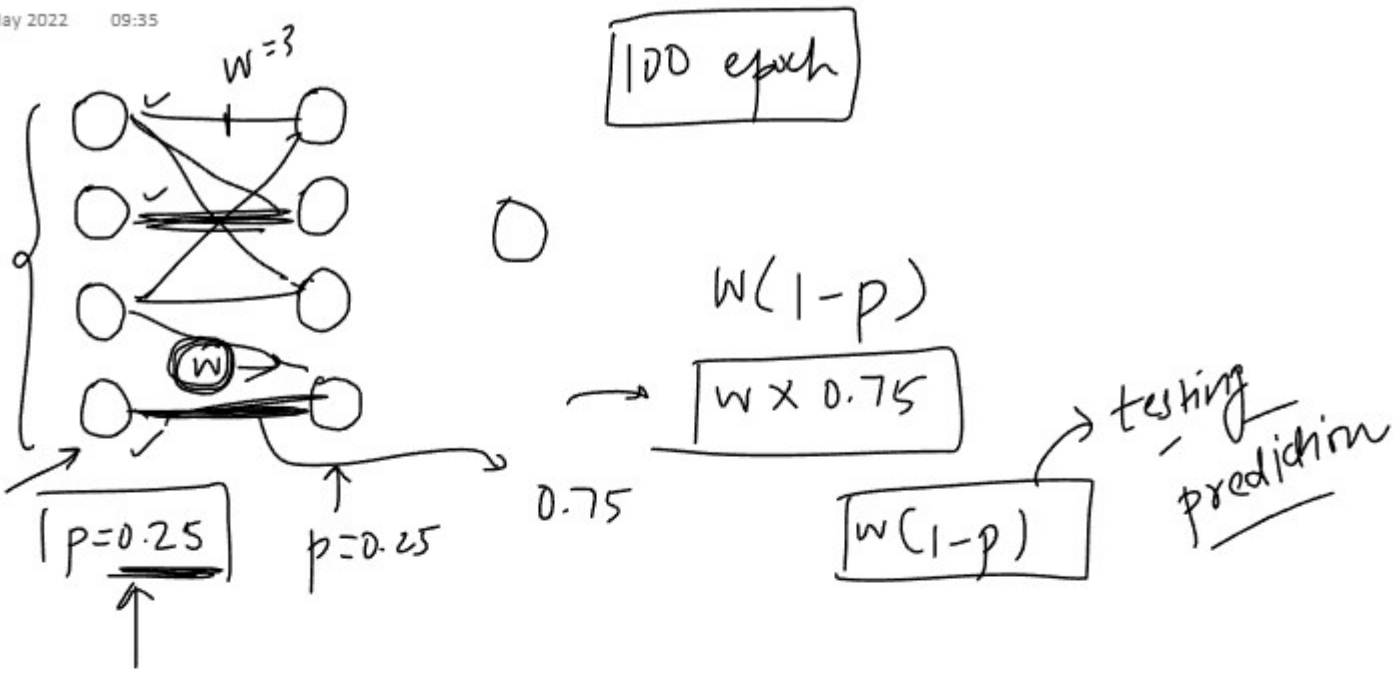
Random Forest Analogy

12 May 2022 09:33



How prediction works?

12 May 2022 09:35



Regression Code Example

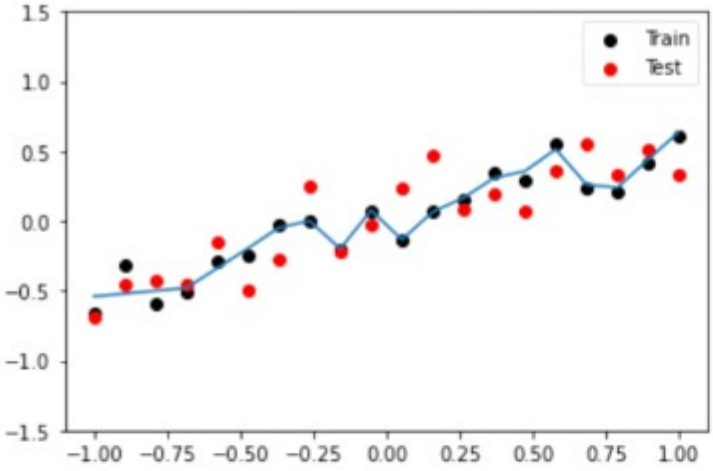
12 May 2022 09:34

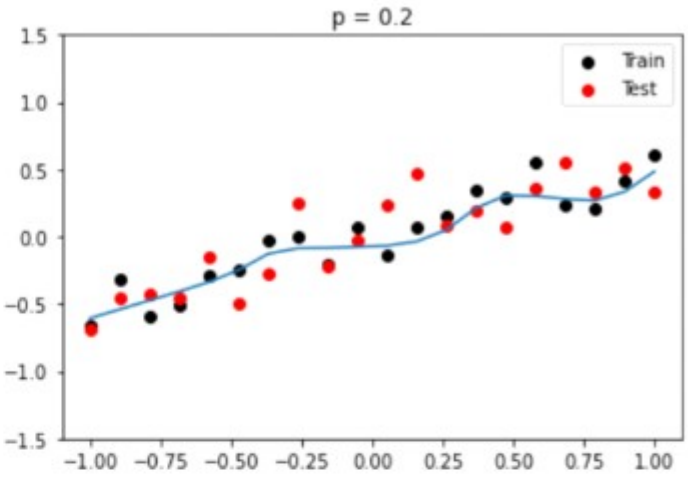
Classification Code Example

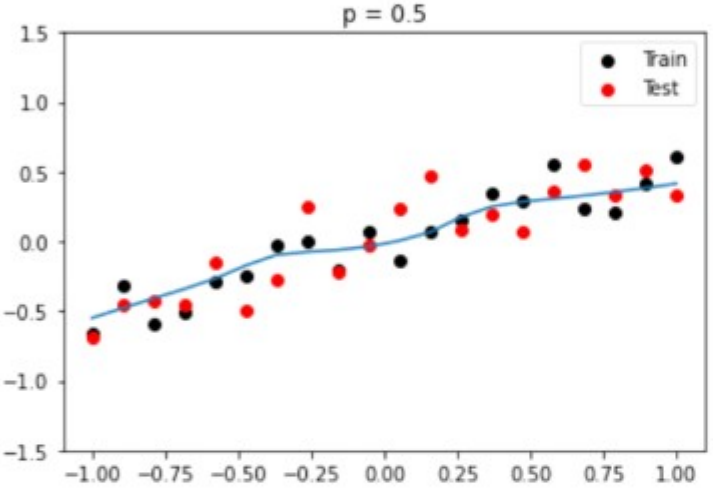
12 May 2022 09:34

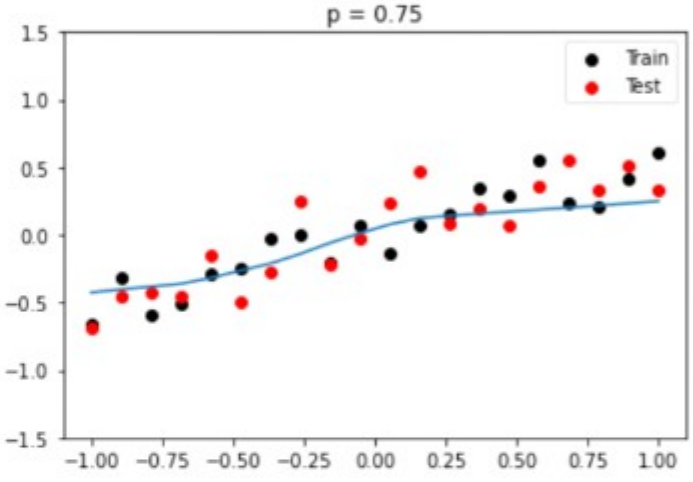
Effect of p

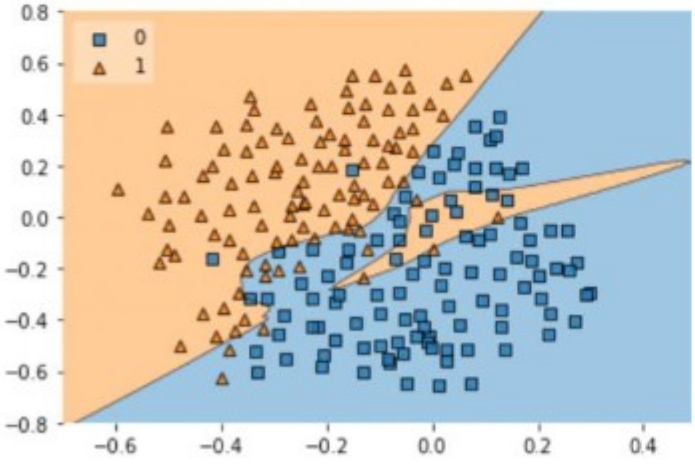
12 May 2022 10:06

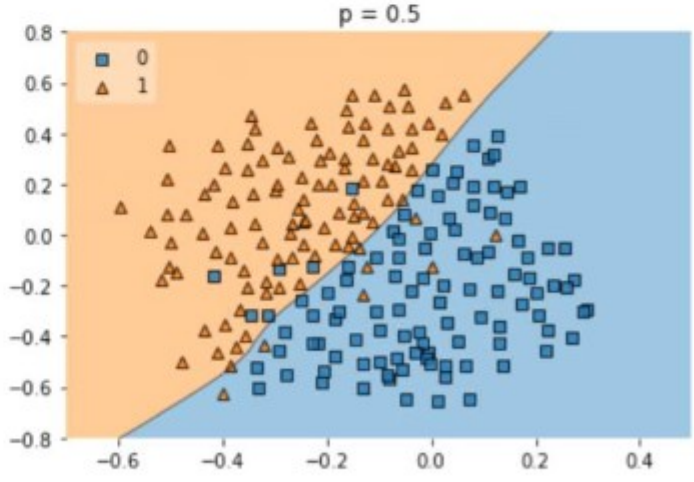










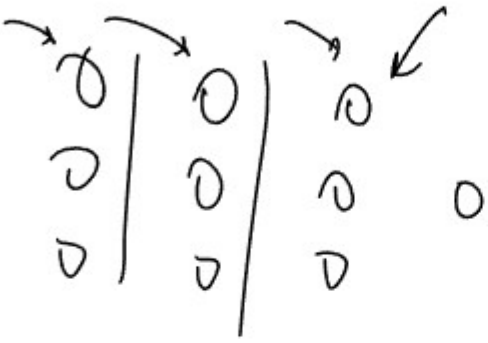


Practical Tips and Tricks

12 May 2022 09:34

1) Overfitting P↑, underfitting P↓

2) Last layer → dropout



3) CNN → 40-50 · (P) → ✓

20-30 RNN 50>

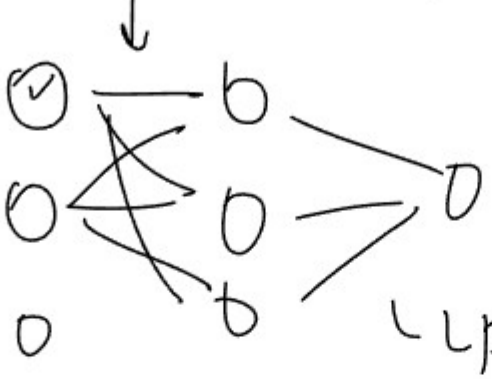
ANN → 10-50

Drawbacks

12 May 2022 09:34

1) convergence → delay

2)



{ loss function } value changes

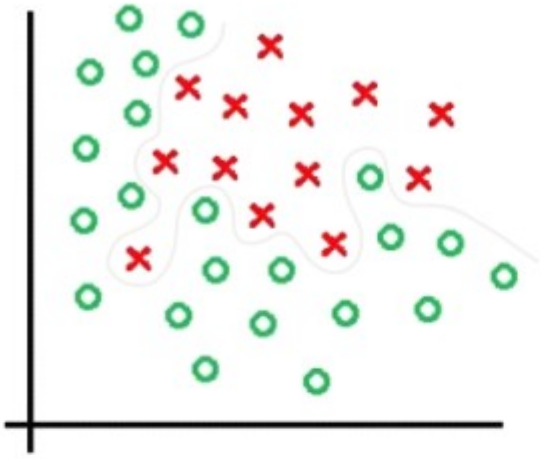
↳ LF – has to change

Resources

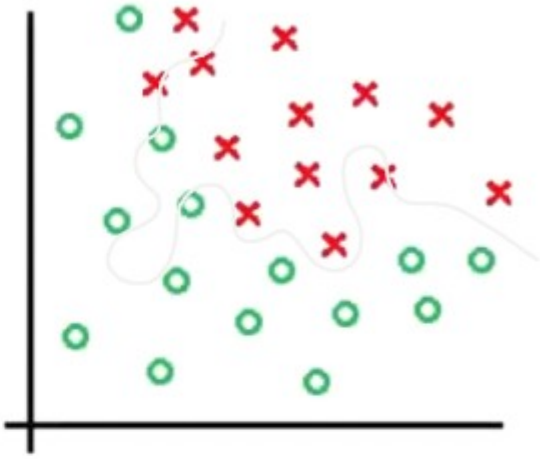
12 May 2022 12:38

Overfitting

20 May 2022 08:14



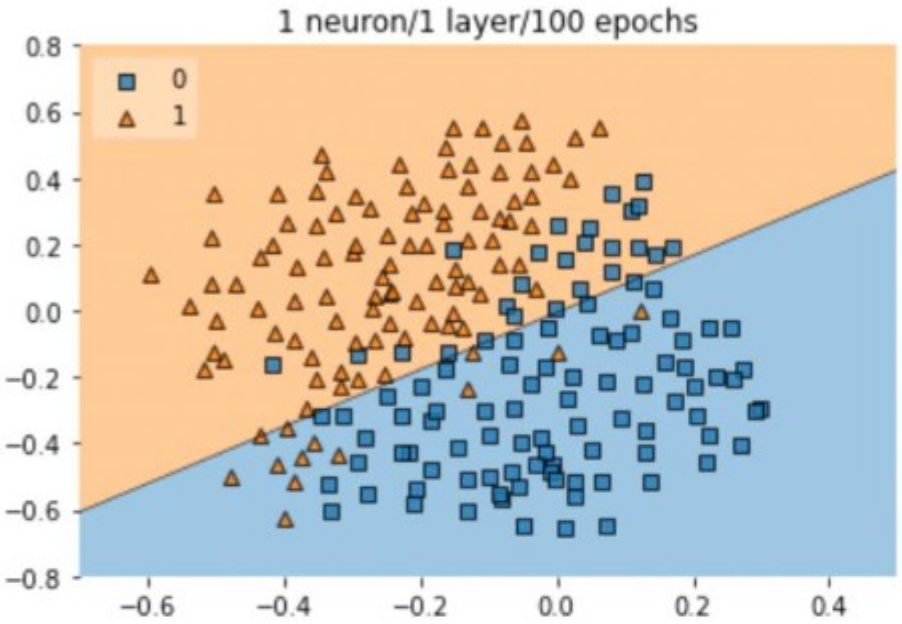
Train Dataset

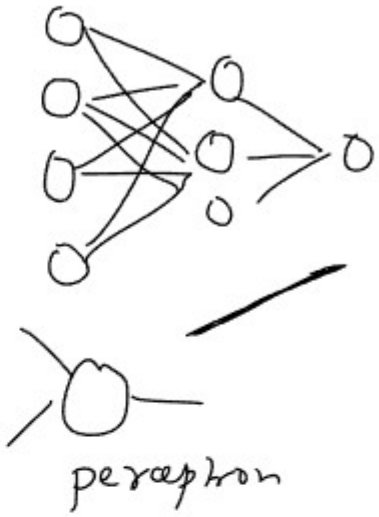


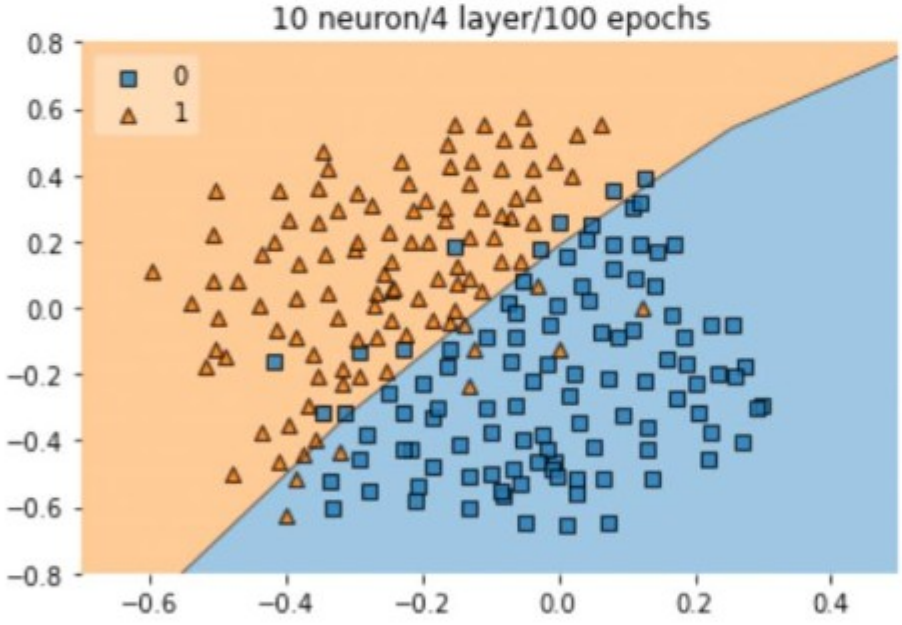
Test Dataset

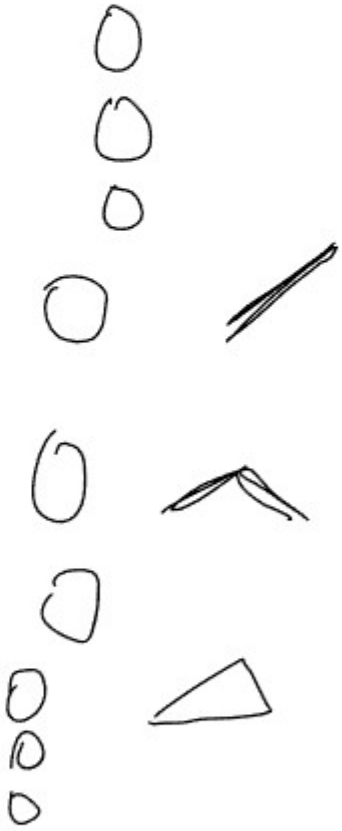
Why Neural Networks Overfit?

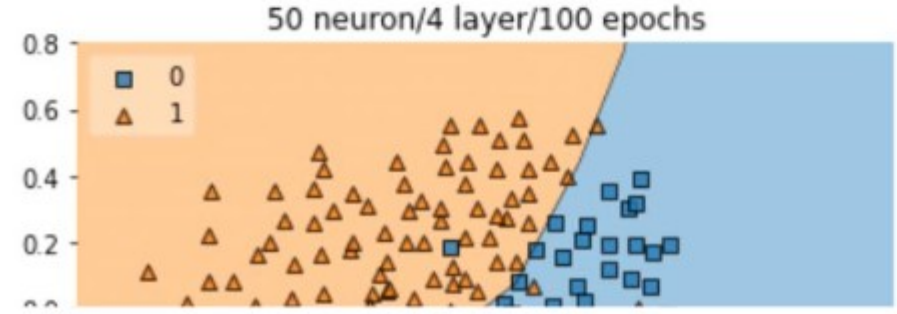
19 May 2022 07:34

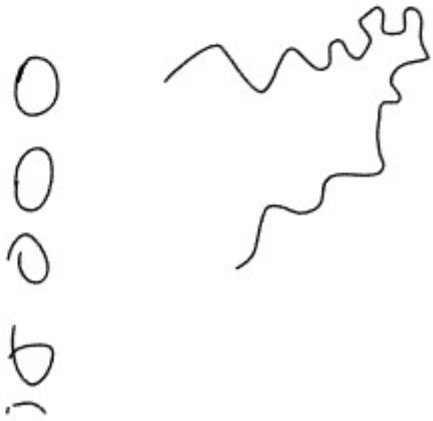


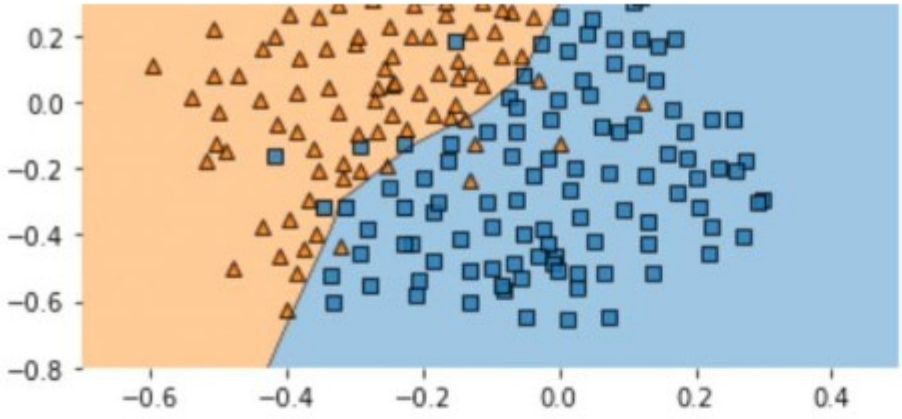


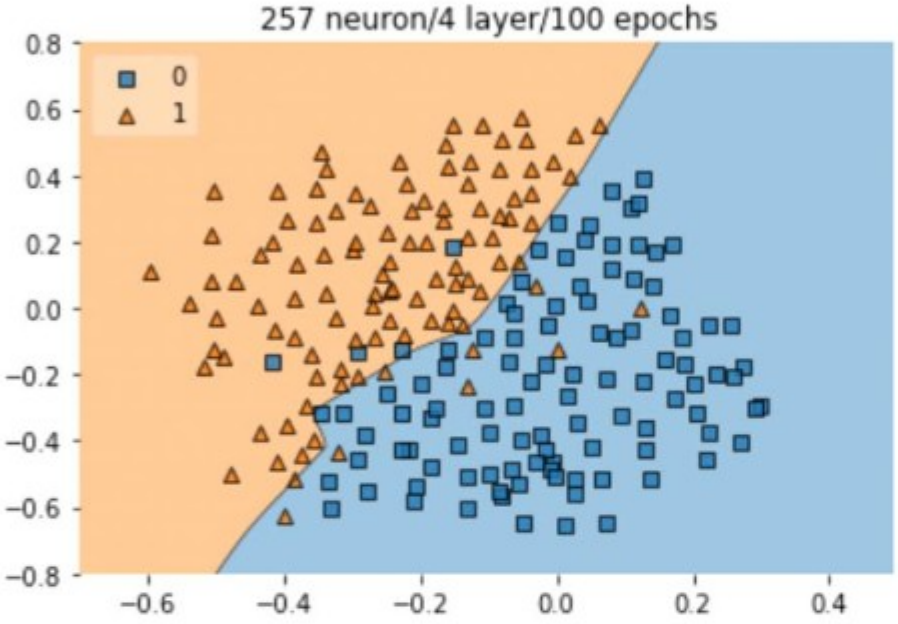


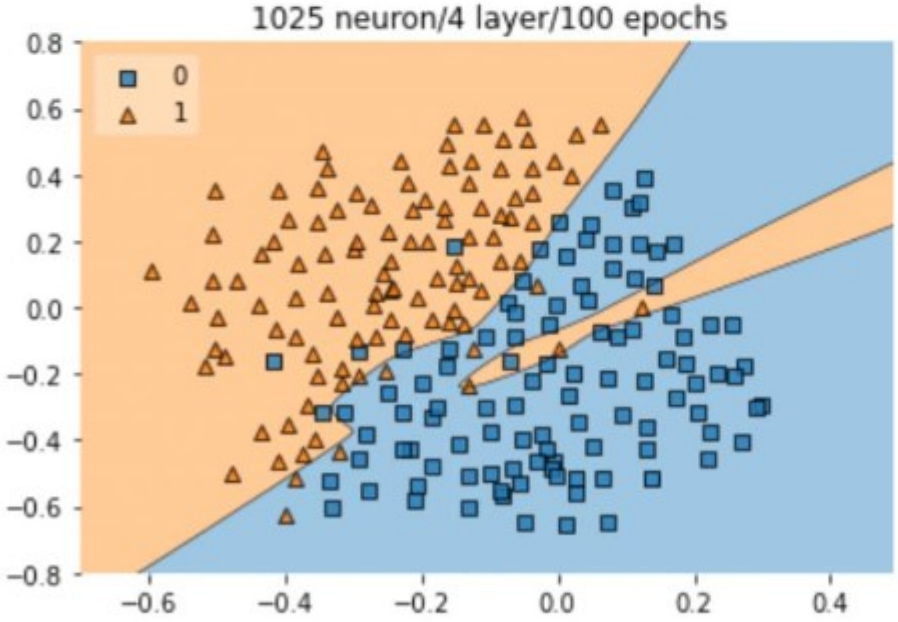


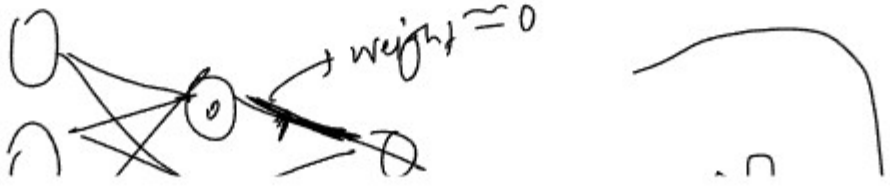


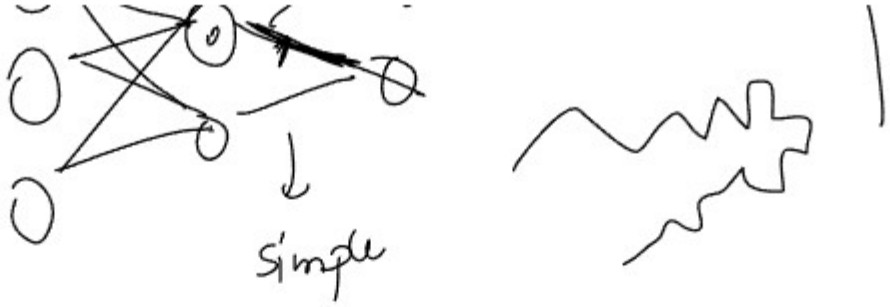





257 neuron/4 layer/100 epochs
0
1

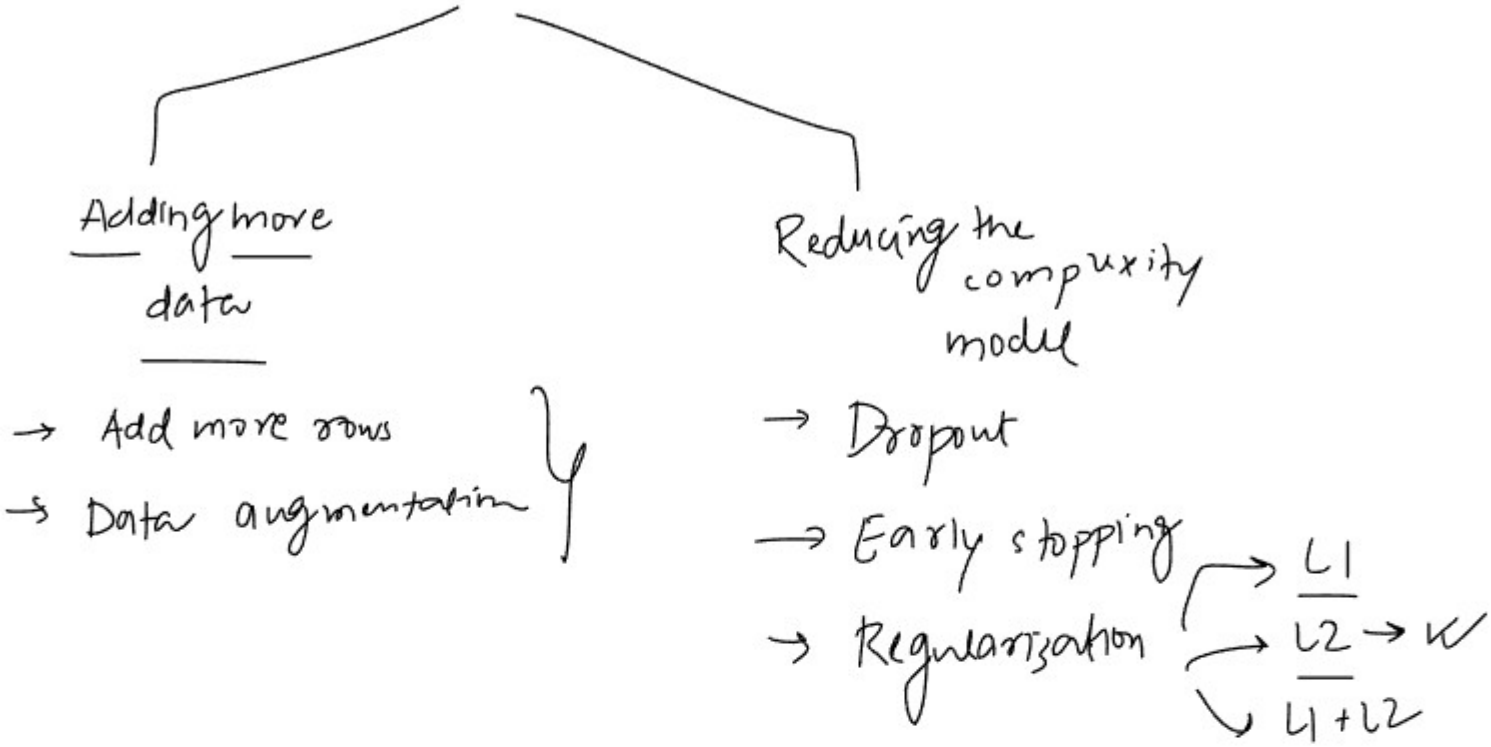

1025 neuron/4 layer/100 epochs
0
1


weight ≈ 0


simple

Ways to solve overfitting

20 May 2022 08:13



Regularization

20 May 2022 08:14

ANN → weights | bias

↳ min Loss function

$$C = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i) + \text{penalty term}$$

L = mse

↳ binary

L2

L1

$$C = L + \frac{\lambda}{2n}\sum_{i=1}^{k} \|w_i\|^2 \rightarrow \text{weightage}$$

$w_1 \rightarrow w_{10}$

$$\frac{\lambda}{2n}\left[w_1^2 + w_2^2 + \dots + w_{10}^2\right]$$

↑ λ = hyperparameter ↑

$\lambda = 0$

$$C = \sum L(y_i, \hat{y}_i) + P$$

L2 → (L1)

$P \rightarrow w \simeq 0$

→ L1 norm

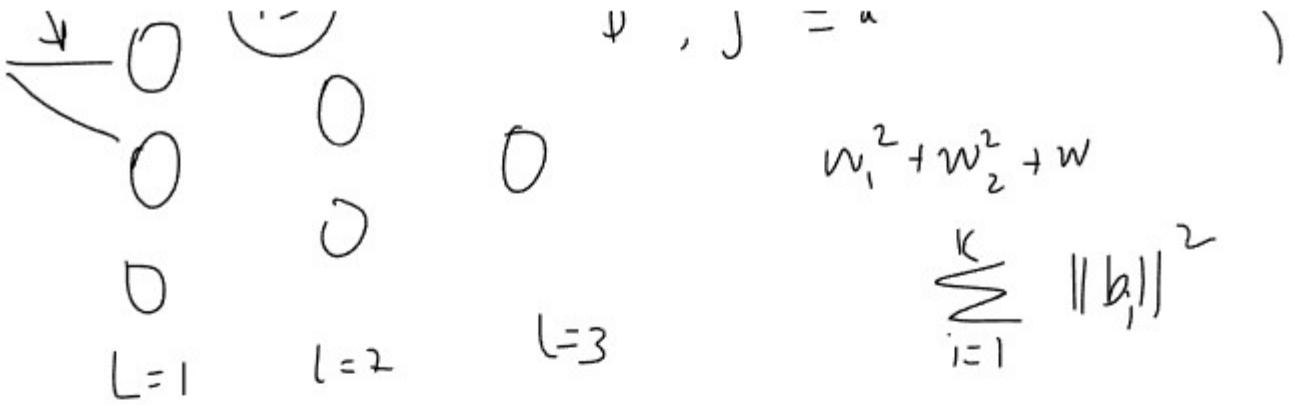
$$C = L + \frac{\lambda}{2n}\sum \|w_i\|$$

$$C = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{l=1}^{L}\sum_{i=1}\sum_{j=1} \|w_{ij}^{l}\|^2$$

$w \simeq 0$

$\rightarrow 0$ (15) $\cap$

$0, j = d$



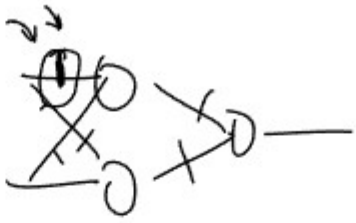
bias X

Intuition behind Regularization

20 May 2022 08:15

$$w_n = w_0 - \eta \frac{\partial L}{\partial w_0}$$

$1-\eta\lambda$ — positive



$$L' = L + \frac{\lambda}{2} \sum \|w_i\|^2$$

$$w_2^1 + w_2^2 + w_3 + \dots$$

$2w$

$$\frac{\partial L'}{\partial w_0} = \frac{\partial L}{\partial w_0} + \frac{\lambda}{2} 2w_0$$

$$= \frac{\partial L}{\partial w_0} + \lambda w_0$$

$$w_n = w_0 - \eta\left(\frac{\partial L}{\partial w_0} + \lambda w_0\right)$$

$$w_n = w_0 - \eta\lambda w_0 - \eta \frac{\partial L}{\partial w_0}$$

$$w_n = (1-\eta\lambda) w_0 - \eta \frac{\partial L}{\partial w_0}$$

$w_0 << w_0$

L2 reg → weight decay

weight decay ↑ w_0

Code Demo

20 May 2022 08:15

Further Resources

20 May 2022 08:15

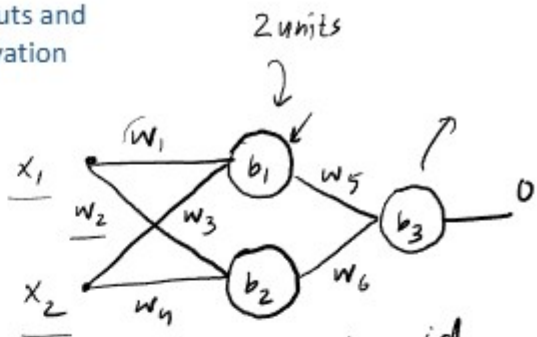
What are Activation Functions?

31 May 2022 14:49

In artificial neural networks, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation function or transfer function. If a neuron has n inputs then the output or activation of a neuron is

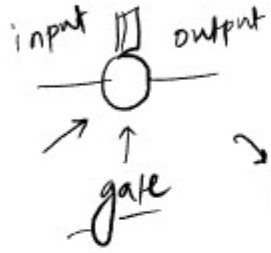
$$a = g(w_1x_1 + w_2x_2 + w_3x_3 + \dots w_nx_n + b)$$

This function g is referred to as the activation function.



$g(w_1x_1 + w_2x_2 + b_1)$ → sigmoid, relu, softmax

g → activation

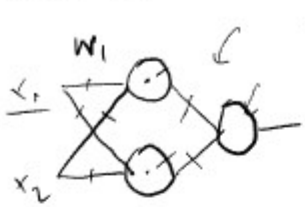


activated (Y/N)

how much

Why Activation Functions are needed?

31 May 2022 14:50



forward prop

$Z_1 = W_1 A_0 + b_1$

$A_1 = g(Z_1) = Z_1$ — linear

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

ANN

↓

Non-linear

Non-linear

output = input

$A_2 = g(W_2 A_1 + b_1)$

(output = input)

$= W_2 A_1 + b_2 = W_2 (W_1 A_0 + b_1) + b_2$

$= W_2 W_1 A_0 + W_2 b_1 + b_2$

$A_2 = \hat{y} = W' A_0 + b'$

linear

$= W' A_0 + b'$

Ideal Activation Function

31 May 2022 14:50

$y = f(x)$ ✓ linear x non-linear

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

1) Non-linear ✓ → Universal Approx Theorem

2) Differentiable ✓ → gradient des → derivatives → relu (0)

3) Computationally inexpensive ✓ → derivative → simples easy fast → training slow

4) Zero centered ↑ ← zero cent normalized → mean = 0

normalized data input → later layers → tanh

5) Non-saturating → Sigmoid → [0, 1] → faster

tanh → (-1, 1)

$f(x) = \max(0, x)$ → Relu

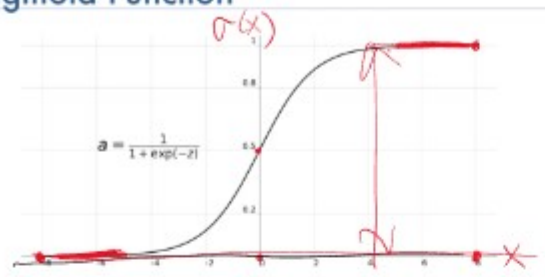
update x [0, 1] → backprop → saturating AF → Vanishing gradient Problem

0.01 → 0.0001 → 0.0001

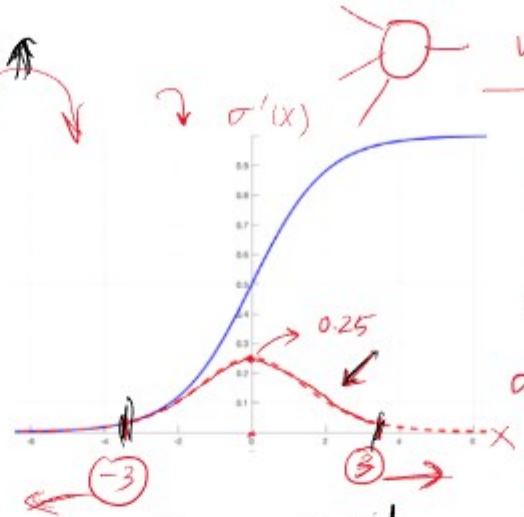
Sigmoid Activation Function

31 May 2022 14:50

Sigmoid Function



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



$w_1x_1 + w_2x_2 + b$

input ↑ → Norm

$x \simeq 0$

$\sigma'(x) \simeq 0$

$w, w_2\ b$

must init

Yes

NO

sigmoid → 0-1 — yes

↳ No

Advantages

1) [0,1] → probability → output layer → Binary classification

2) Non-linear → Non-linear data → good option

3) Differentiable → Backprop → $\frac{\partial L}{\partial w}$ ← X

Disadvantages

1) Saturating function → $[-\infty, \infty] \rightarrow [0,1]$

↳ Vanishing gradient problem

Backprop → update

$w_n = w_0 - \eta \frac{\partial L}{\partial w}$

$w_n = w_0$

training X

No update will take place

Hidden layer

sigmoid → training X

output → binary classification

2) Non Zero centered

Normalized → mean = 0

+ve / -ve → Training slow

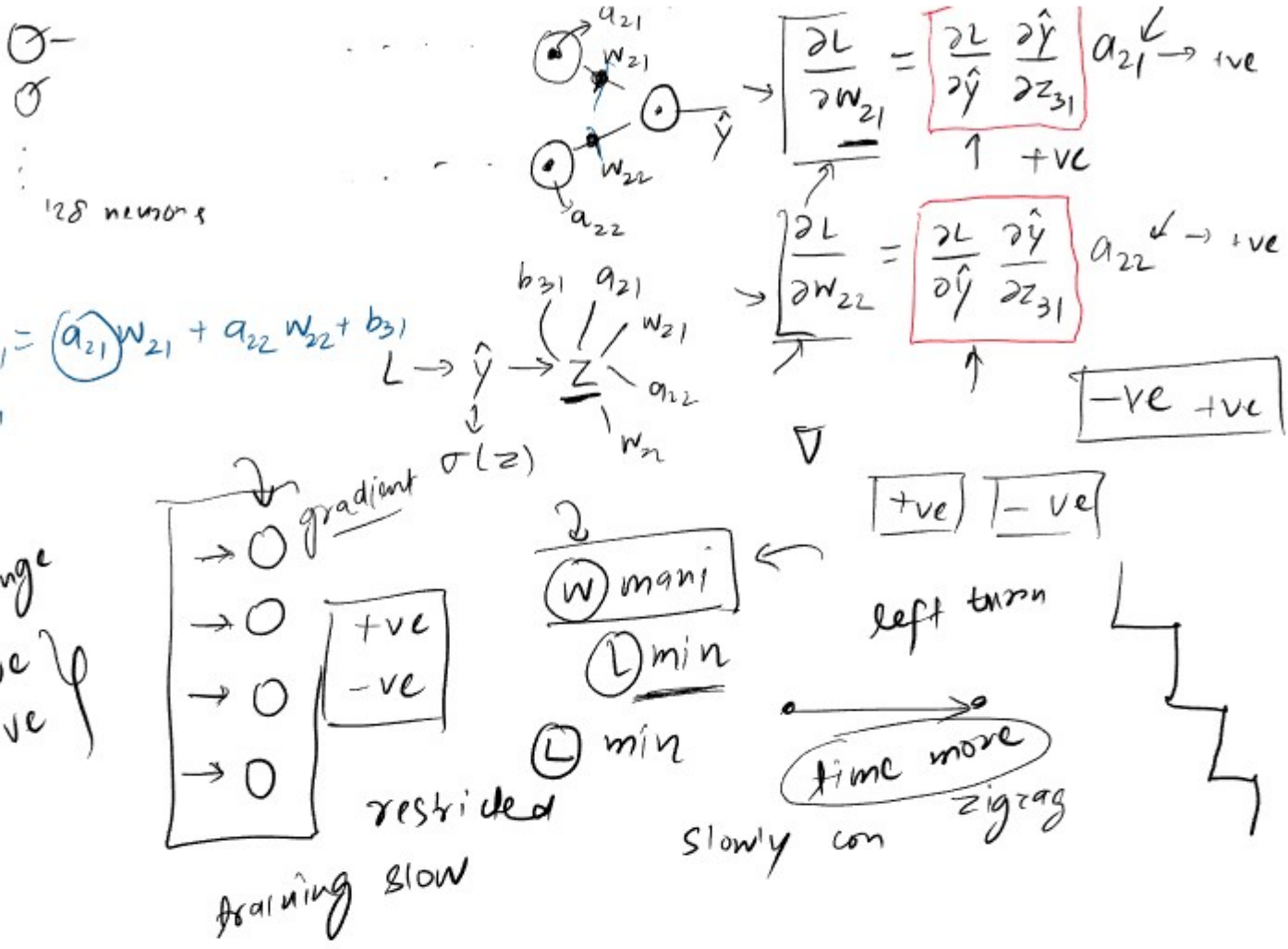
Convergence problem

zero

+ve -ve

0 → 1

$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} a_{21}$ → +ve



3) $\sigma(x) = \frac{1}{1+e^{-x}}$ → computation time

exp

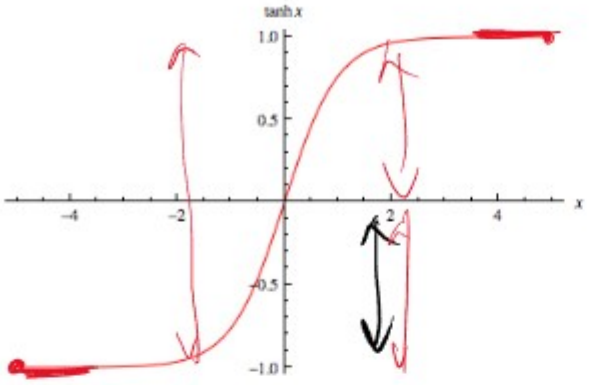
hidden layer → sigmoid → relu

output → b cp

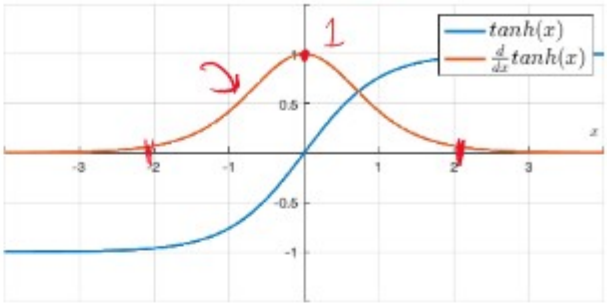
Tanh Activation Function

31 May 2022 14:50

(0,1) (-1,1)



0.25



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

$$f'(x) = (1 - \tanh^2(x))$$

Advantages

1) Non-linear
2) Differentiable
3) zero centered (+ve, -ve) ↳ training faster sigmoid

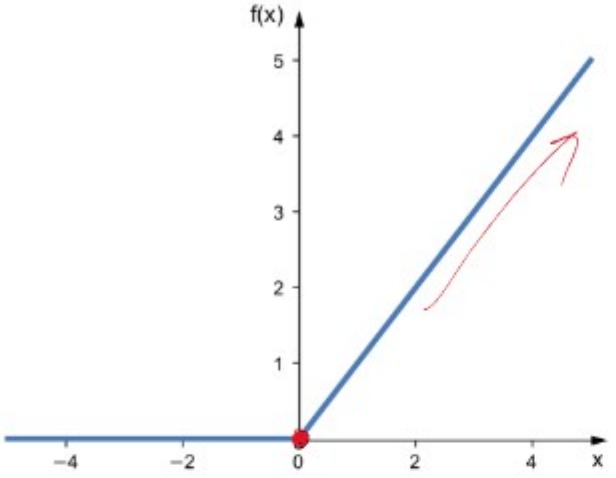
Disadvantage

→ saturating function → vanishing gradient prob

→ computationally exp [exp] slow

Relu Activation Function

01 June 2022 16:43



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

Advantage

$f(x) = \max(0, x+1) - \max(0, x-1)$

1) Non-linear
2) Not saturated in the +ve region
3) Computationally inexpensive
4) Converge → faster → sigmoid ↘ tanh

Disadvantage

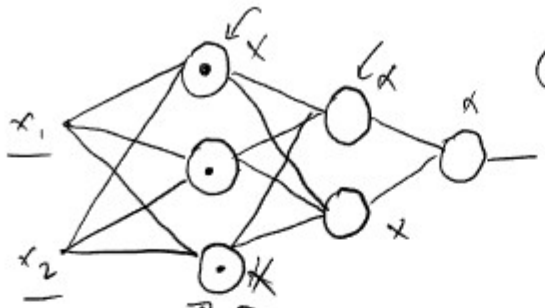
→ Differentiable $x < 0 \rightarrow 0$, $x \geq 0 \rightarrow 1$

→ Non Zero centered → sigmoid → Batch Normalization normalize →

↳ Dying ReLU problem

Dying Relu Problem

08 June 2022 22:53



Relu → Vanishing grad

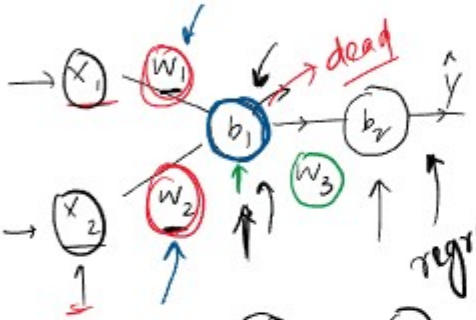
Best choice hidden layers

→ Dying Relu → 50% >

0 → dead neuron → forever dead (0)

100% dead → worst

modified data → low level regress



$a_1 = \max(0, z_1) = 0$

$z_1 = w_1x_1 + w_2x_2 + b_1$ < 0

$\frac{\partial a_1}{\partial z_1} = 0$

regression

$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$ -ve

$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$ -ve

+ve

$L \to \hat{y} \to w_3, b_2$

$a_1 \to z_1 \to w_1, w_2, x_1, x_2, b_1$

$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$

$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_2}$

dying relu

$z_1 = w_1x_1 + w_2x_2 + b_1$ < 0 (-ve)

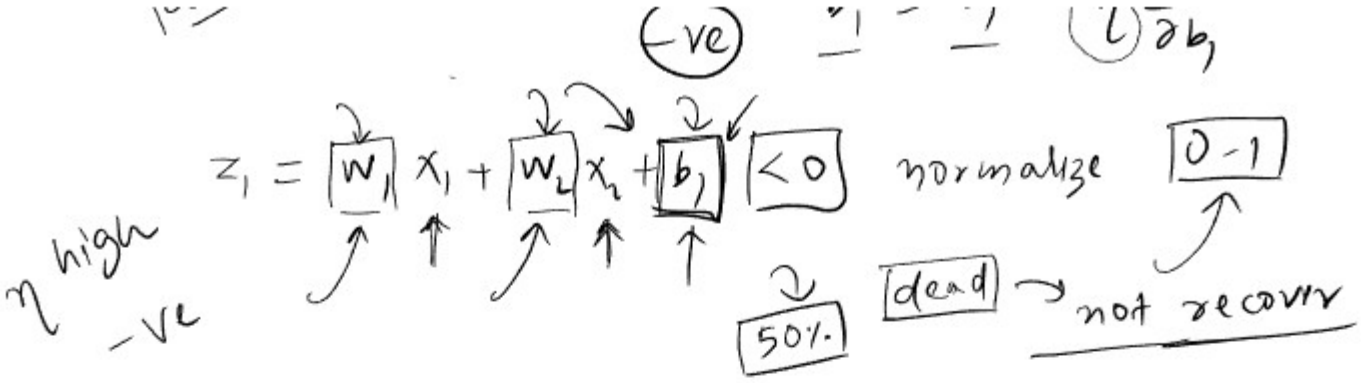
$b_1 << 0$ (-ve)

dead

1) High learning rate (η)

2) High -ve bias

$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}$ (-ve)

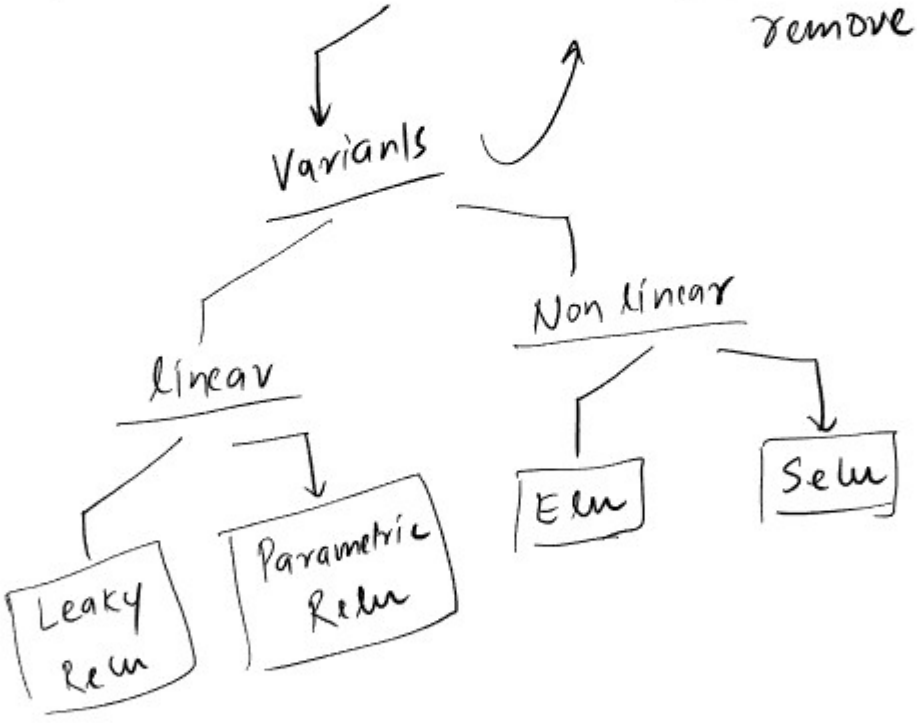


Solutions

→ Set low learning rate

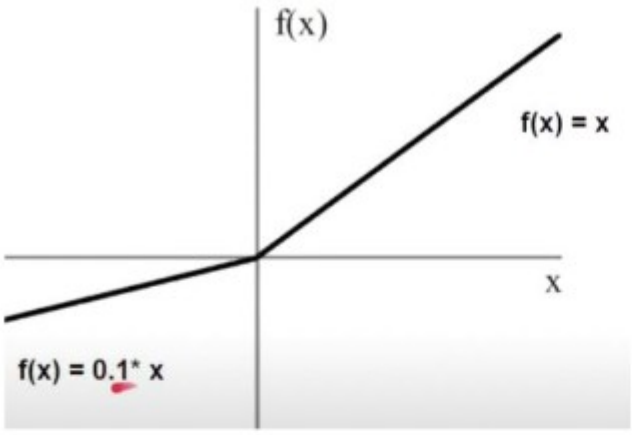
→ bias → +ve value → 0.01

→ Don't use relu → variants (+ve keep) / remove



Leaky Relu

08 June 2022 22:53



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

$z \geq 0 \rightarrow z$

$z < 0 \rightarrow \frac{1}{100} z$ (fraction of z)

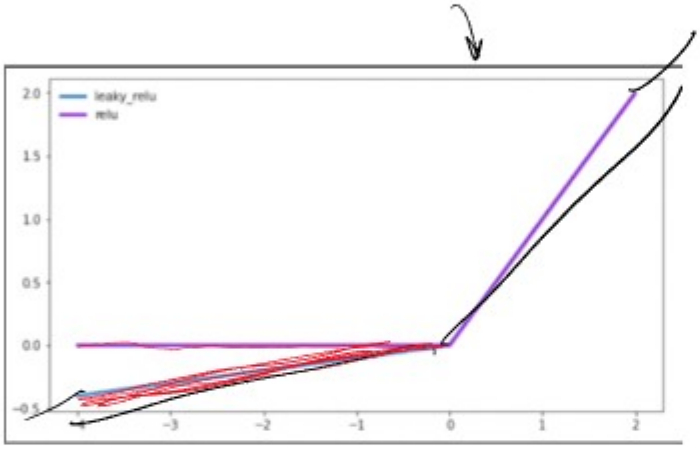
$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$

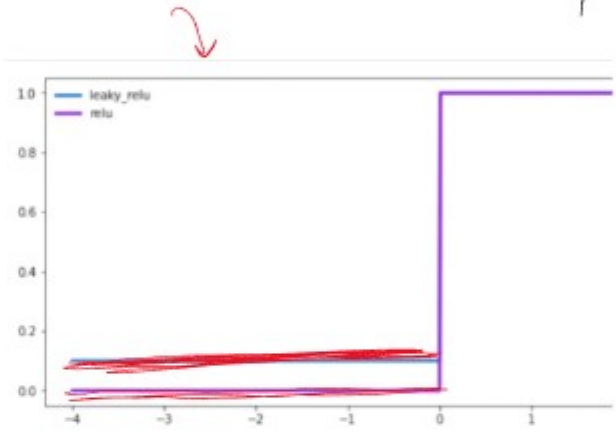
-ve (z) $\rightarrow \neq 0$ [0.01] z

↳ derivative

$f'(z)$ $z \geq 0 \rightarrow$ [1]

$z < 0 \rightarrow$ [0.01]





Advantages

→ Non-saturated → Unbounded

→ Easily computed

→ No dying relu problem

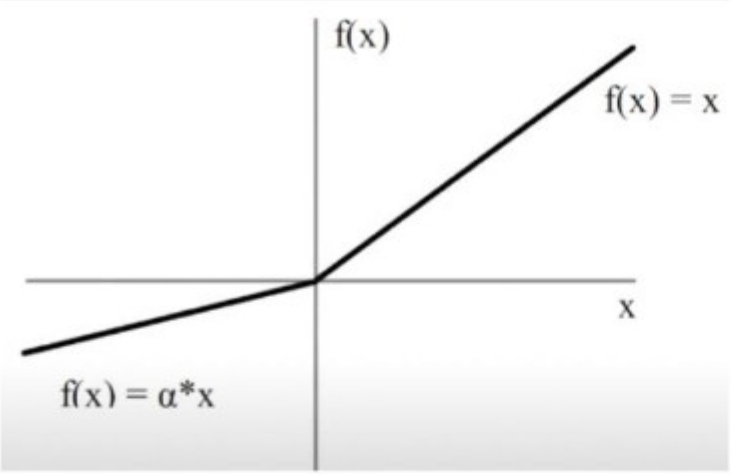
→ Close to 0 centered

-ve/+ve

Disadv

Parametric Relu

08 June 2022 22:53



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

$a \rightarrow$ trainable parameter

0.01x

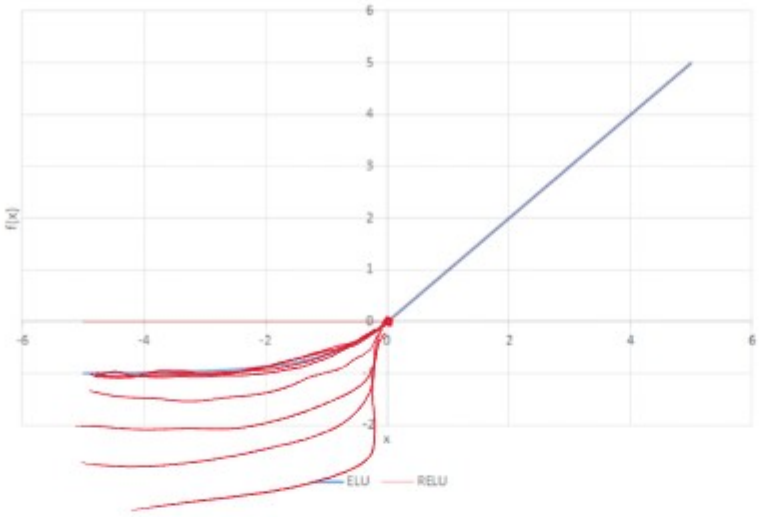
depend on data

flexibility

Elu - Exponential Linear Unit

09 June 2022 00:29

performance better Relu

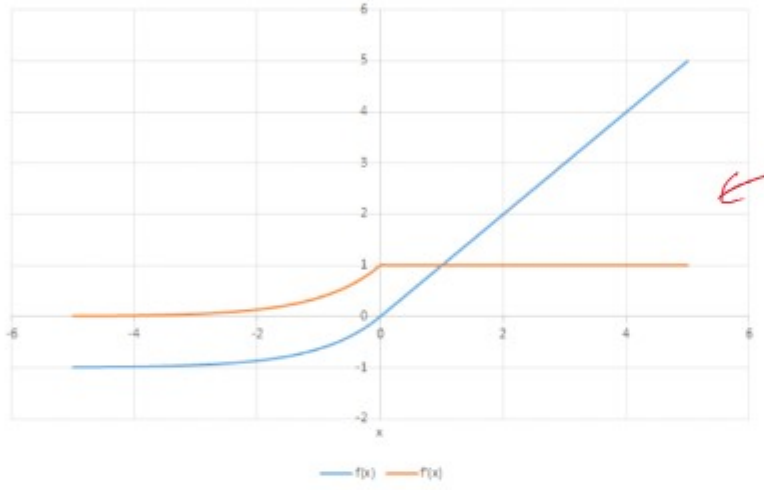


$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

constant

0.1 to 0.3

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



Advan

→ Close zero centered
↳ convergence faster

→ Better generalized

→ Dying relu

→ Always continous as well diffen

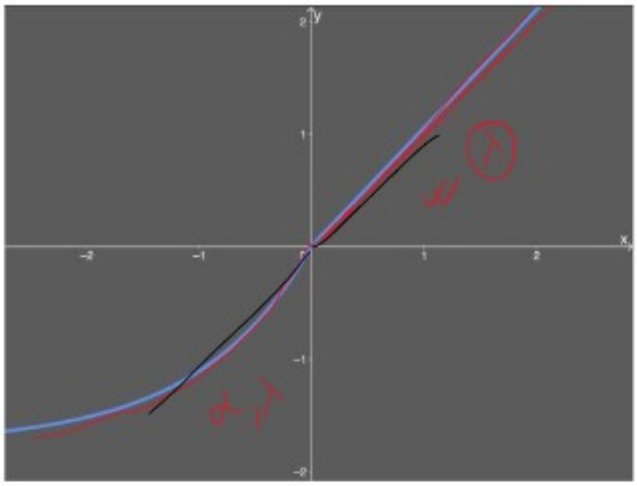
Disadv

→ computation expensive (ex)

Selu - Scaled Exponential Linear Unit

09 June 2022 00:29

Recent → new → 90+ pages



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

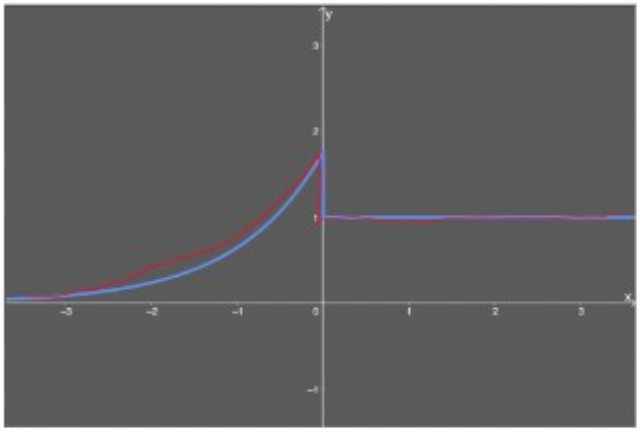
$$a \approx 1.673263242354377284817042991671 7$$

$$\lambda \approx 1.050700987355480493419334985294 6$$

fixed ✓

trainbu ✗

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



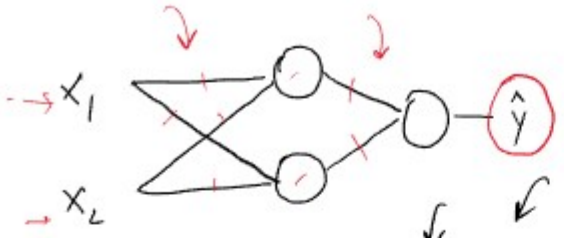
self-normalizing → activation → normalized

$m = 0 \quad \sigma = 1$

converge faster

Why Weight Initialization is Important?

22 June 2022 12:48



→ Vanishing GP
→ Exploding GP
→ Slow convergence

Wrong WI techniques

↓

What to do?..

2000's Deep learning

1. Initialize the parameters
2. Choose an *optimization algorithm*
3. Repeat these steps:
   1. Forward propagate an input
   2. Compute the cost function
   3. Compute the gradients of the cost with respect to parameters using backpropagation
   4. Update each parameter using the gradients, according to the optimization algorithm

Sigmoid Activ

Wrong weight init

What not to do?

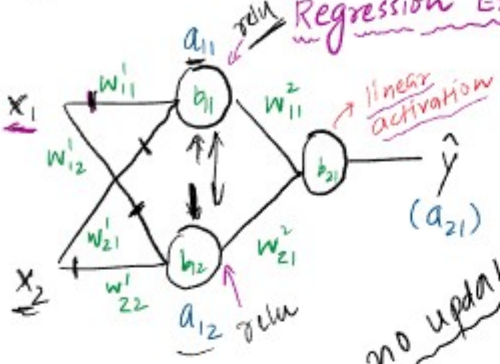
22 June 2022 12:49

Case1 → Zero Initialization

relu / tanh / sigmoid

$W = 0, b = 0$

Regression Example



$$a_{11} = \frac{e^{z_{11}} - e^{-z_{11}}}{e^{z_{11}} + e^{-z_{11}}} = 0$$

$$a_{11} = \max(0, z_{11}) = 0$$

$$z_{11} = w^1_{11} x_1 + w^1_{21} x_2 + b_{11} = 0$$

$$a_{12} = \max(0, z_{12}) = 0$$

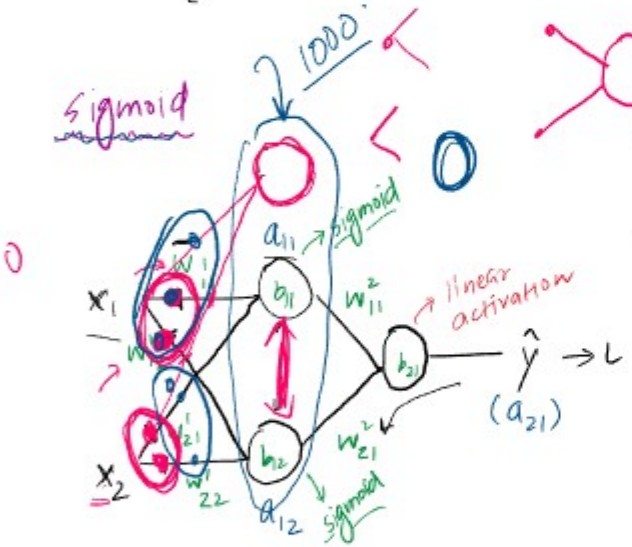
$$z_{12} = w^1_{12} x_1 + w^1_{22} x_2 + b_{12} = 0$$

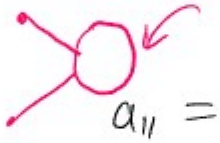
$a_{11} = a_{12}$ → equal and 0

no update

$$w^1_{11} = w^1_{11} - \eta \frac{\partial L}{\partial w^1_{11}}$$

$W = 0$ ← No training will place





0

$a_{11} = \sigma(z_{11}) = 0.5$

$z_{11} =$

$a_{12} = 0.5$

$a_{11} = a_{12}$

Non-linear prop

$$W = W - \eta \frac{\partial L}{\partial W}$$

$a_{11} = a_{12}$

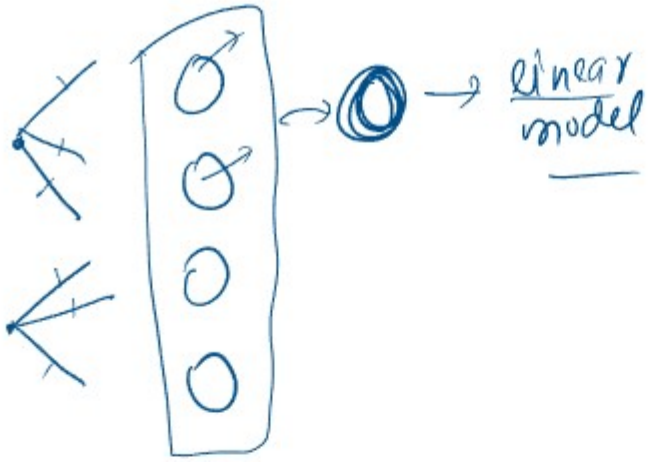
$z_{11} = z_{12}$

$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_1$$

$$\frac{\partial L}{\partial w^1_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_1$$

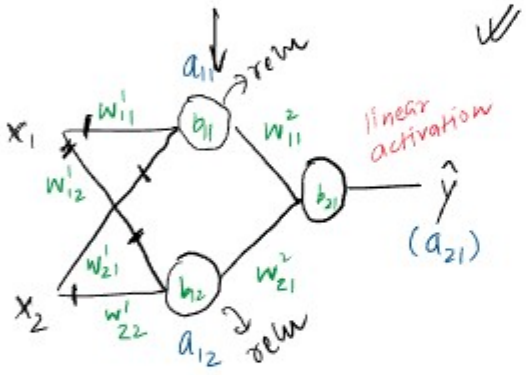
$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_2$$

$$\frac{\partial L}{\partial w^1_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_2$$



Case2 → Non-0 constant value

$w = 0.5 \quad b = 0.5$ sigmoid → 0



$a_{11} = \max(0, z_{11})$ → non-zero

$z_{11} = w^1_{11} x_1 + w^1_{21} x_2 + b_{11} \neq 0$

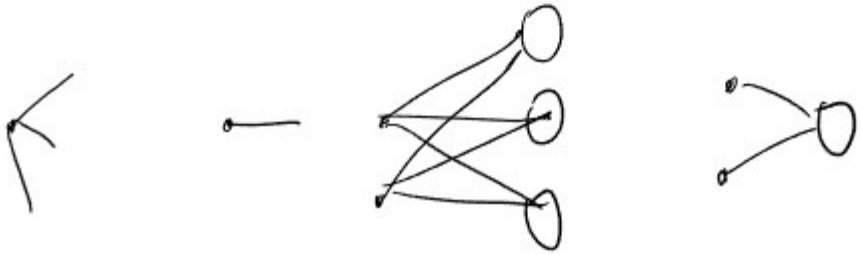
$a_{12} = \max(0, z_{12}) \neq 0$

$z_{12} = w^1_{12} x_1 + w^1_{22} x_2 + b_{12} \neq 0$

$z_{12} = z_{11}$ $\quad \boxed{a_{11} = a_{12}}$

$$\boxed{\frac{\partial L}{\partial w^1_{11}}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_1 \qquad \boxed{\frac{\partial L}{\partial w^1_{12}}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_1$$

$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_2 \qquad \frac{\partial L}{\partial w^1_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_2$$

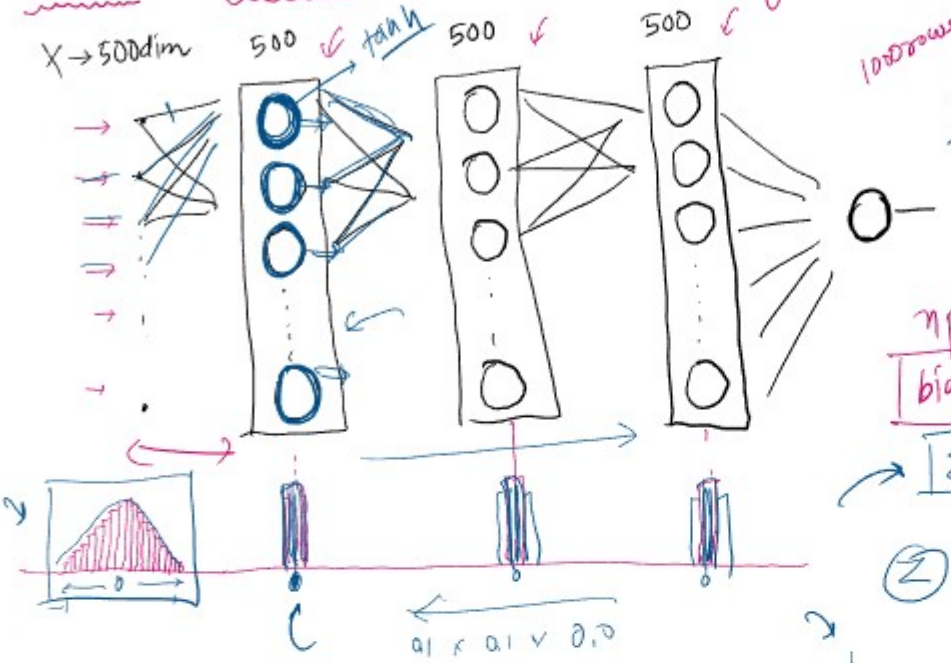


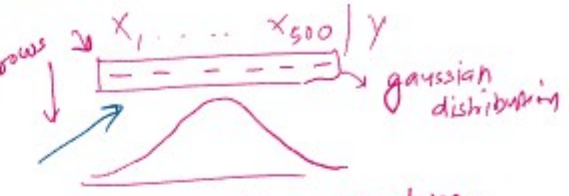
random init

Case3 – Random Initialization

small → tanh / sigmoid
large

VGP sigmoid





weight init → random
small values

np.random.randn (500, 500) * 0.01

bias = 0

$x_i \rightarrow$ 0-1 → -1 to 1

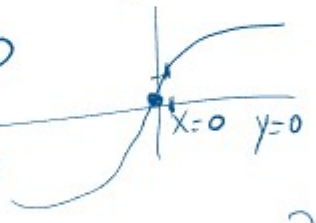
$\sum w_i x_i$ → 0.0007

small number tanh

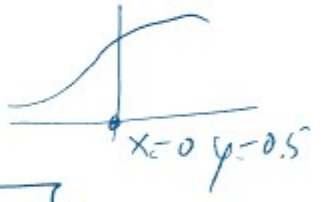
Z is a small

small small ≈ 0

$W = W - \eta \frac{\partial L}{\partial W} \approx 0$



very close to 0



vanishing gradient problem

VGP → deep

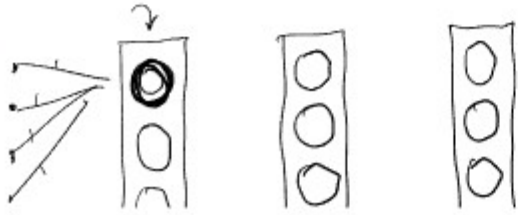
small weights → 0.0007

Relu → small values → slow convergence

tanh → VGP

sigmoid → VGP

Random Initialization (Large values)

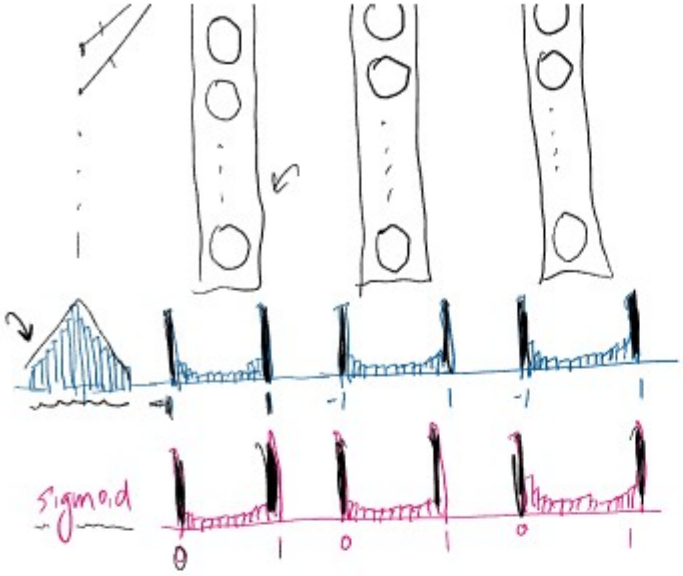


np.random.randn (500, 500) → 0-1

$\sum w_i x_i$ → 500 → 100 → 500

0-1

tanh / sigmoid



0-1 tanh/sigmoid

$g(z)$ → saturation → saturate

$\sum w_i x_i \to 100$

$\sum w_i x_i \to$ Relu → 250 → 260

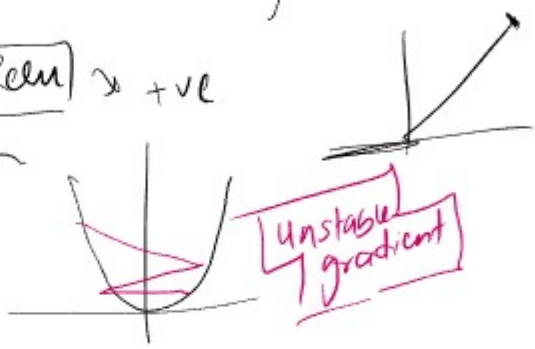
↓

gradients

- Slow training
- Vanishing gradient problem

$$w = w - \eta \frac{\partial L}{\partial w}$$

Relu → +ve



Unstable gradient

→ Zero ✗

→ non-zero same ✗

→ small random ✓

→ large random ✗

Key Insights

22 June 2022 12:49

Things not to do—

1) Zero initialization → no training
2) Non-zero constant initialization → linear
3) Random initialization with small weights → vanish
4) Random initialization with large weights → exploding

random

np.random.randn(shape) * 0.01

(2,2)

-0.01 to 0.01

Vanishing grad

Slow convergence

np.random.randn(2,2)

-3 to 3

What can be done

→ Heuristics (Jugad) → practical solution

Xavier/Glorot Init (tanh, sigmoid) → Normal / Uniform

He Init (relu) → Normal / Uniform

Keras

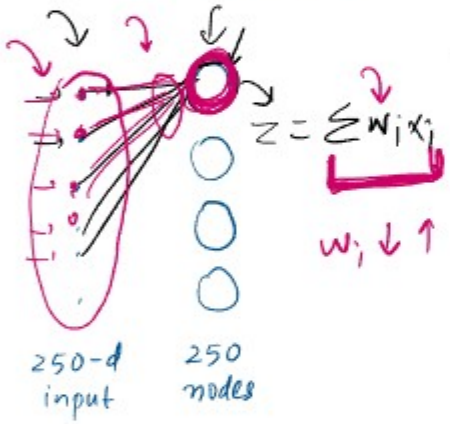
formulas

-1 to 1

0-1

-0.01 to 0.01

Intuition



small ←

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

250

-250 to 250

large

np.random.randn(250,250) * 1 → -3 to 3

Variance = $\frac{1}{n} = \frac{1}{250}$ * $\sqrt{\frac{1}{250}}$ → sd

$n \rightarrow \uparrow \downarrow \frac{1}{n} \downarrow \uparrow w$

Xavier init (Normal)

$\sqrt{\frac{1}{fan\_in}}$ ← # of inputs coming to the node

np.random.randn(2,2) * $\sqrt{\frac{2}{2}}$

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

He Normal ← relu

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

Uniform Distribution

Xavier Uniform

$[-limit, limit]$

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

Keras

He Uniform

$[-limit, limit]$ $\quad limit = \sqrt{\frac{6}{fan\_in}}$

Weight Initialization Techniques

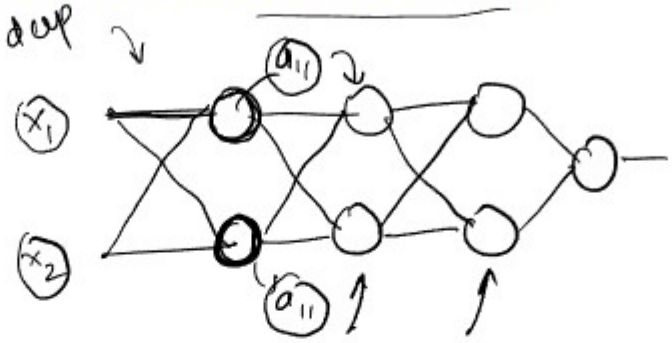
22 June 2022 12:49

What is Batch Norm?

27 June 2022 11:00

Batch-Normalization (BN) is an algorithmic method which makes the training of Deep Neural Networks (DNN) faster and more stable.

↳ It consists of normalizing activation vectors from hidden layers using the mean and variance of the current batch. This normalization step is applied right before (or right after) the nonlinear function.

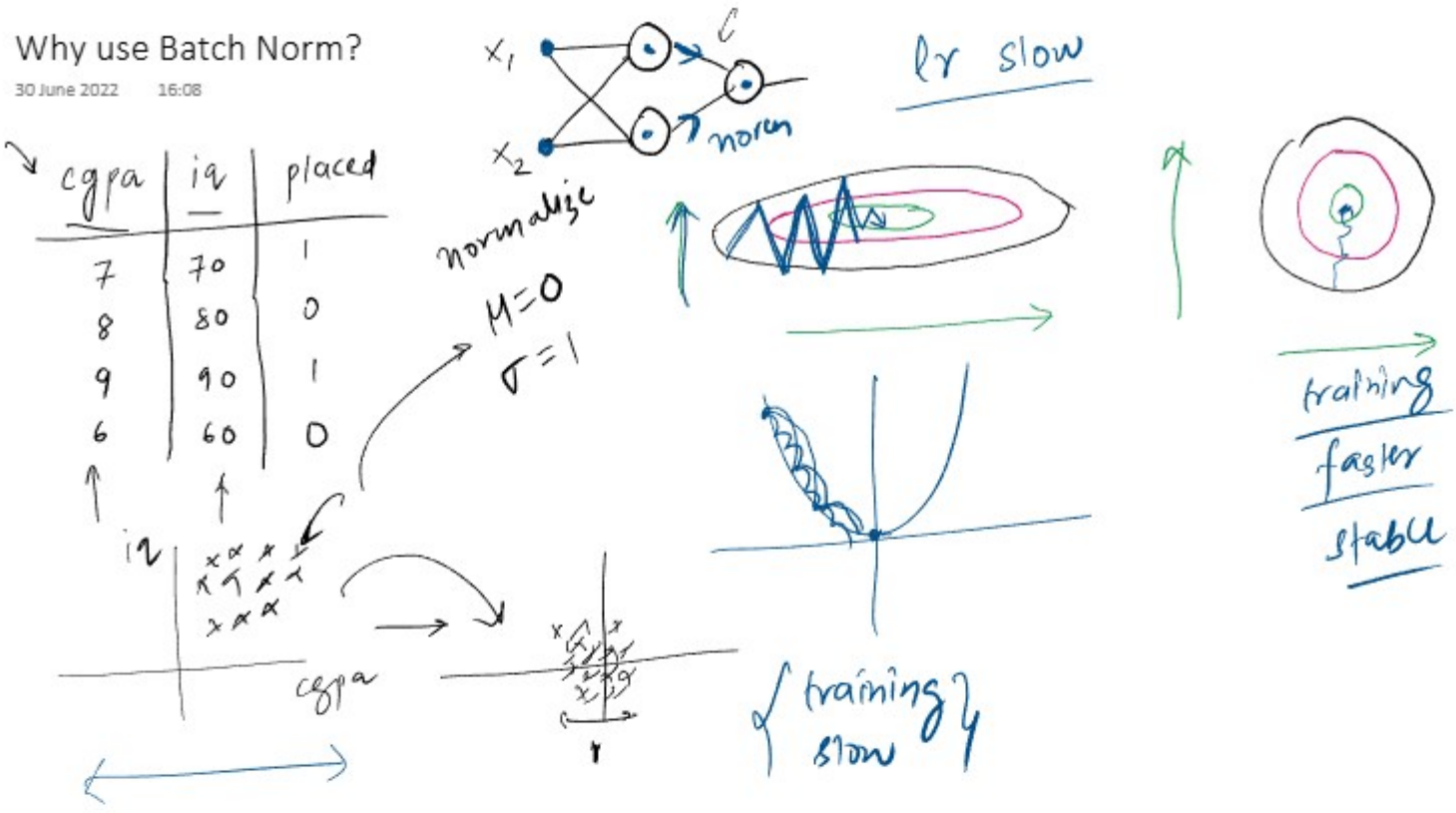


normalized
m=0
sd=1

normalize
m=0 sd=1

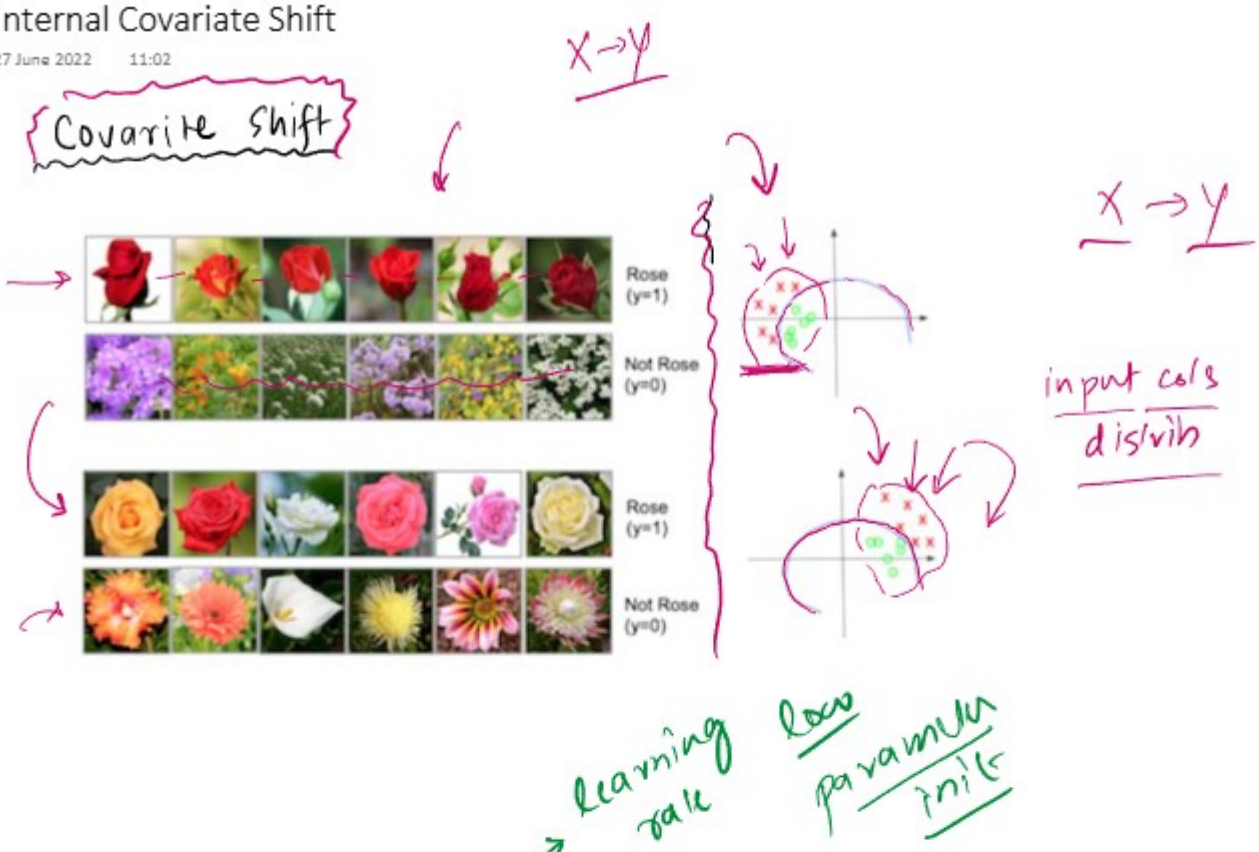
Why use Batch Norm?

30 June 2022 16:08



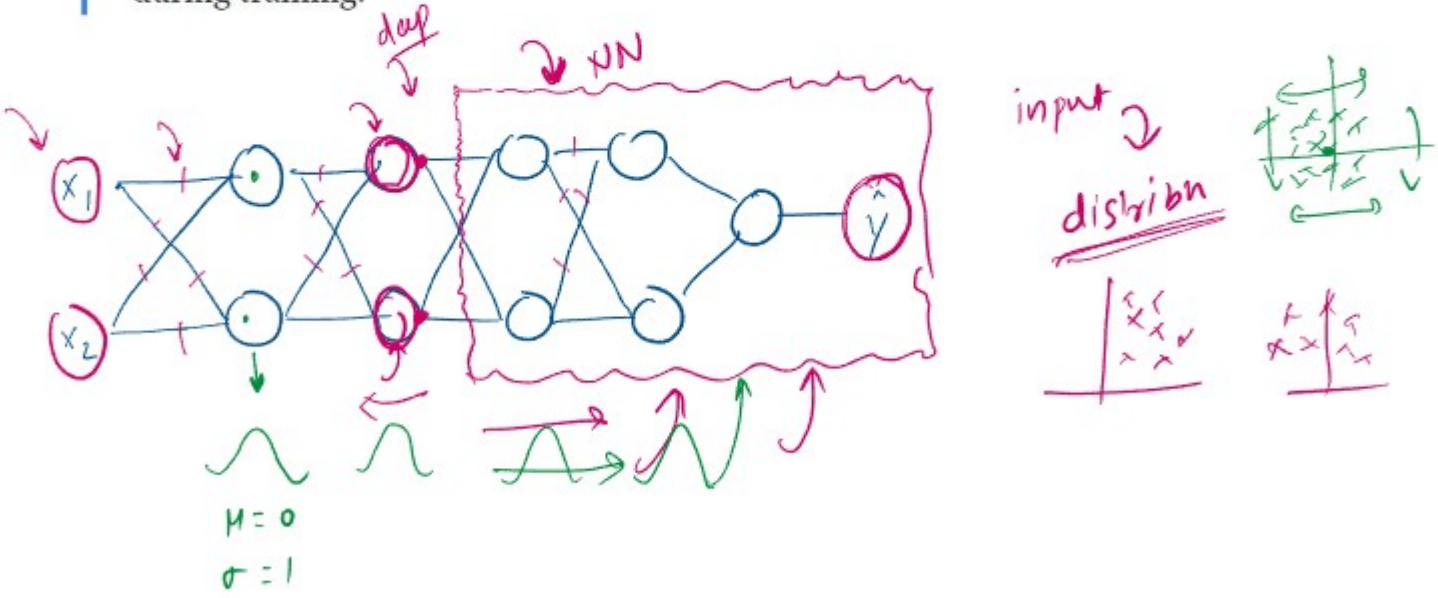
Internal Covariate Shift

27 June 2022 11:02



The authors' precise definition is:

> We define Internal Covariate Shift as the change in the distribution of network activations due to the change in network parameters during training.




bees
knees
cheese
fleas?

Batch Norm - The How

27 June 2022 11:02

cgpa	iq	placed
8.9	100	1
6.2	89	0
9.1	91	0
7.7	76	1
⋮	⋮	⋮
6.7	91	0

batch_size : 4

$z_{11} = w_1 \text{cgpa} + w_2 \text{iq} + b$

$g(z_{11}) = a_{11}$

(4,2) * (2,2) → (4,2) + (1,2)

fp → unumb

→ Mini-Batch GD

→ Layer by layer

unumb normalize $\mu=0$ $\sigma=1$

(1) $z_{11} \to z_{11}^N \to g(z_{11}^N) = a_{11}$

(2) $z_{11} \to g(z_{11}) = a_{11} \to a_{11}^N$

$$\frac{z_{11} - \mu}{\sigma} = z_{11}^N \quad (4,2)$$

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} z_{11}^i \qquad m = 4$$

$$z_{11}^i = \frac{z_{11}^i - \mu_B}{\sigma_B + \epsilon}$$

$$\sigma_B = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (z_{11}^i - \mu_B)}$$

$\gamma\beta$ → $\mu=0, \sigma=1$ → $\gamma\beta$

$$z_{11}^{BN} = \gamma z_{11}^N + \beta$$

0-1 $\mu=0$ $\sigma=1$

learnable parameters w, b

$\gamma=1$ $\beta=0$

$z_{11} \to z_{11}^N \to z_{11}^{BN} \to g(z_{11}^{BN}) = a_{11}$

z_{11} → normalize $\mu=0$ $\sigma=1$

$\gamma = \sigma + \epsilon$ $\beta = \mu$

flexibility

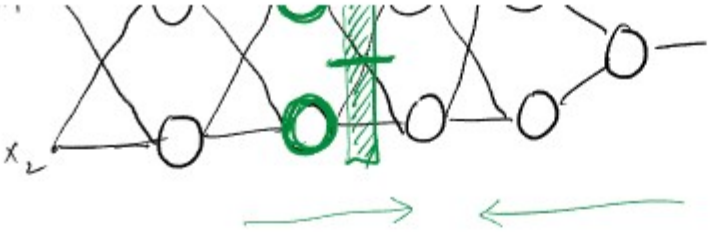
$z_{11} \to z_{11}^N \to z_{11}^{BN}$

Batch Norm → Layer

4 learnable $\gamma\beta$ $\gamma\beta$

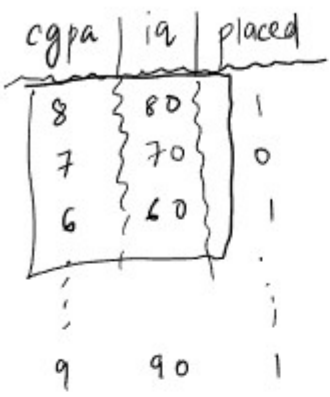
$$\gamma = \gamma - \eta \frac{\partial L}{\partial \gamma}$$

Keras



Batch Norm during test

27 June 2022 11:03



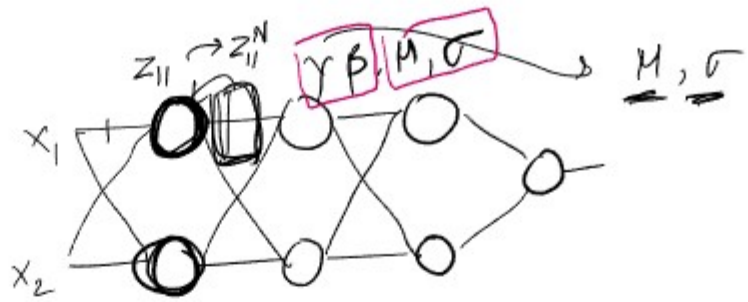
100 students

b = 4

25 times

1 batch

2



testing / query point

Exp weighted moving avg.

EWMA

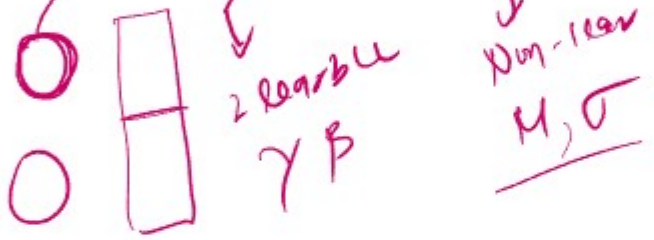
μ_{B1} , μ_{B2} . . . . μ_{B25} → μ

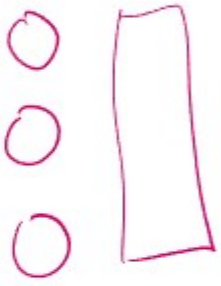
ex γ WMA

4 parameters

2 learnable γ β

Non-learn μ, σ





$3 \times 4 = 12$

6 6

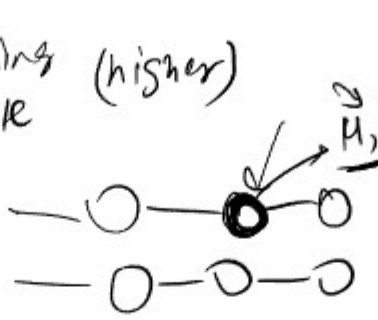
Advantages

27 June 2022 11:02

1) Stable → hyperpara → wider range of values

2) faster → learning rate (higher)

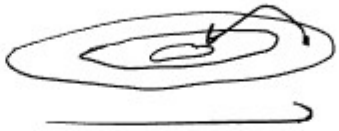
3) Regularizer →



randomness ↓ rise

overfitting

dropout



4) Weight init impact reduce

Keras Implementation

27 June 2022 11:03

```
model = Sequential()

model.add(Dense(3,activation='relu',input_dim=2))
model.add(BatchNormalization())
model.add(Dense(2,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1,activation='sigmoid'))
```