FINAL VERDICT OF DEEP LEARNING

↳ Assume a Human Body

- NLP: Gave this body basic language & grammar knowledge.
- ~~con~~ CNN → Gave Eyes to this Body.
- RNN / LSTM: Gave Memory capability to this Body. & GENERATION POWER.
- GAN: Gave Imagination Power (High Quality) to this Body ↳ Painting
- AutoEncoder: Techniques to learn Things in shortcut (By making Pattern)

⇒ GAVE

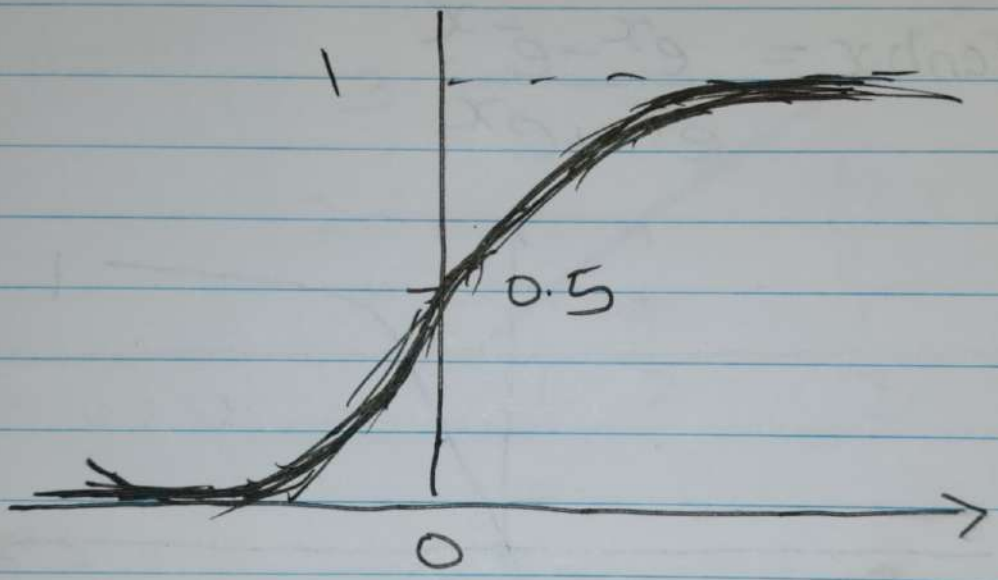
- VAE ⇒ Gave Body Imagination Power ↳ Variation Power like slight changes

apsara

Date: / /

ACTIVATION FXN

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



Merits

↳ (I) Limits output (0, 1)

(II) Differentiable continuous

(III) Used in Binary classification

DeMerits

↳ (I) Vanishing Gradient

↳ (II) Computationally expensive

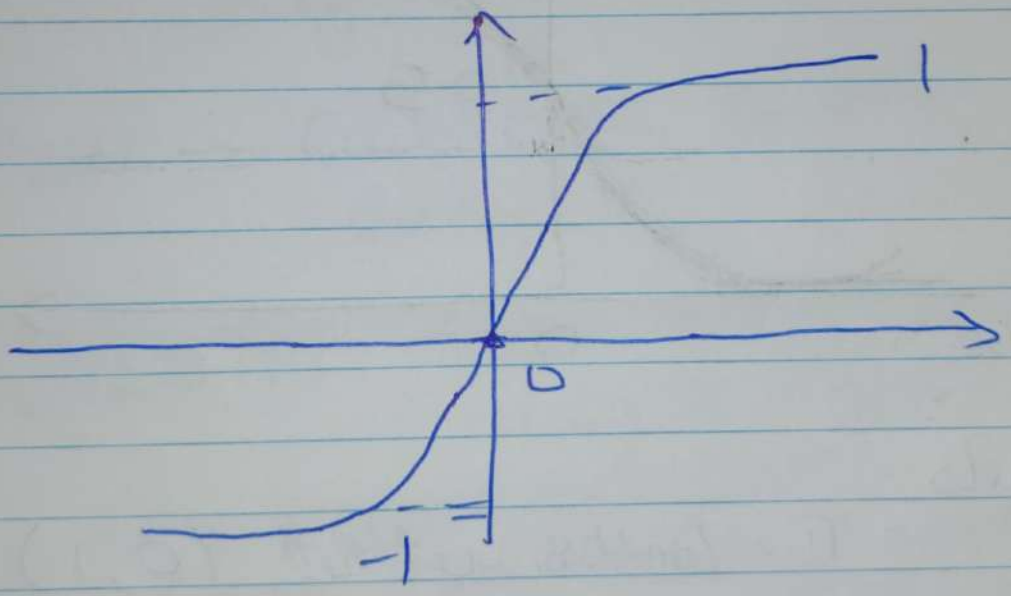
(IV) Not centred at zero

→ Tanh

Tanhx =

$$\tanh x = \frac{\sinh x}{\cosh x}$$

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



Merits

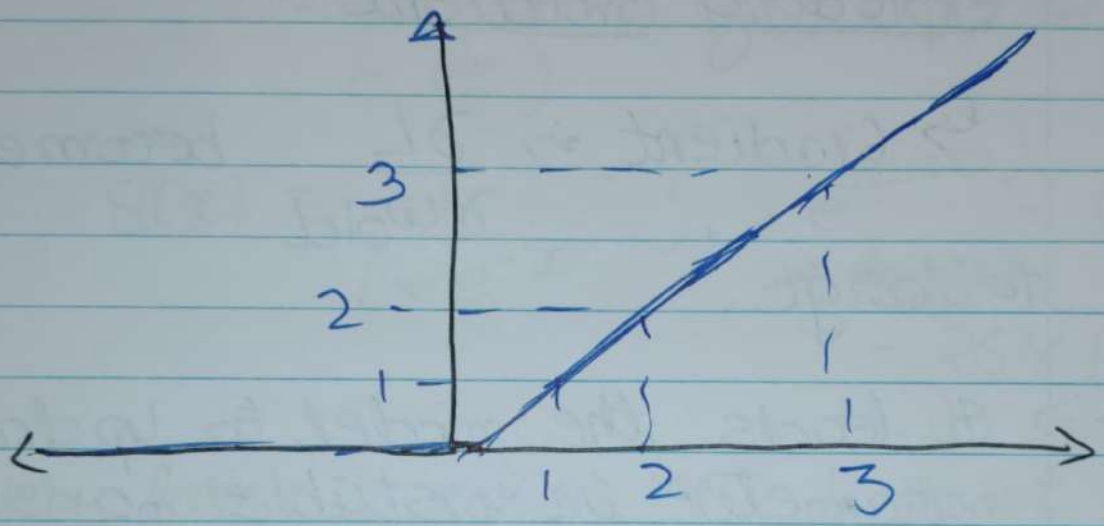
↳ (i) centered at zero
(ii) range (-1, 1)
(iii) continuous & Differentiable

Cons

↳ (i) Vanishing Gradient problem
(ii) Computationaly expensive

→ ReLU (Rectified Linear Unit) Date

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



Merits

↳ (i) Easily to Compute
(ii) Don't cause Vanishing Gradient problem
(iii) outperforms where the output should be non-ve.
(iv) fast & efficient.
↳ as neurons some are not active

Cons

↳ (i) Cause Exploding Gradient problem
(ii) Not centered

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

apsara

Date: __/__/____

(iii) Dying Relu problem

Exploding Gradient

↳ Gradient $\Rightarrow \frac{\partial L}{\partial w_{old}}$ becomes to large.

---> It leads the model to update parameter in unstable manner

Q = How ReLU cause gradient exploding problem?

↳ ReLU allows allows +ve value to pass through w/o modification

↳ This can lead to this to situation

Q = How Sigmoid can lead to vanishing gradient problem?

Ans

↳ In sigmoid the gradient is

$$\sigma'(x) = \sigma(x).(1-\sigma(x))$$

& For x = very large, $\sigma(x) = 1$
x = very small (-ve), $\sigma(x) = 0$

$$S(x) = \frac{1}{1+e^{-x}} ; \quad e^{-(1000)} = 0$$
$$e^{(100)} = 26 \times 10^{42}$$

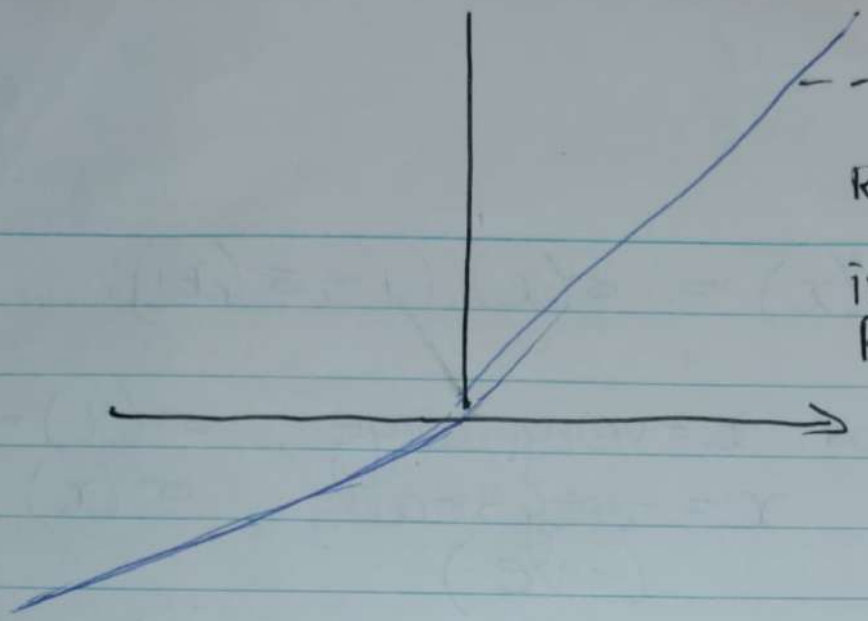
Very large : $\sigma'(x) = 0$ } No wt. update.
Very small : $\sigma'(x) = 0$ }

Q Dying Relu Problem?

↳ When many no. of neurons become inactive & output zero

LEAKY RELU

$$f(x) = \begin{cases} x ; & \text{if } x > 0 \\ 0.01x ; & \text{o/w} \end{cases}$$



↓ apsara

ReLU6

↓

improve Robustness of loss precision computation

→ Solves the problem of ~~Leak~~ Dying ReLU.

Pros : (i) Easy to compute
(ii) Doesn't cause V.G.P.
(iii) " " D.R.P.
Dying → ReLU.

Cons
(i) Exploding Gradient
(ii) Not Centered at zero

Parameterized ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{; o/w} \end{cases}$$

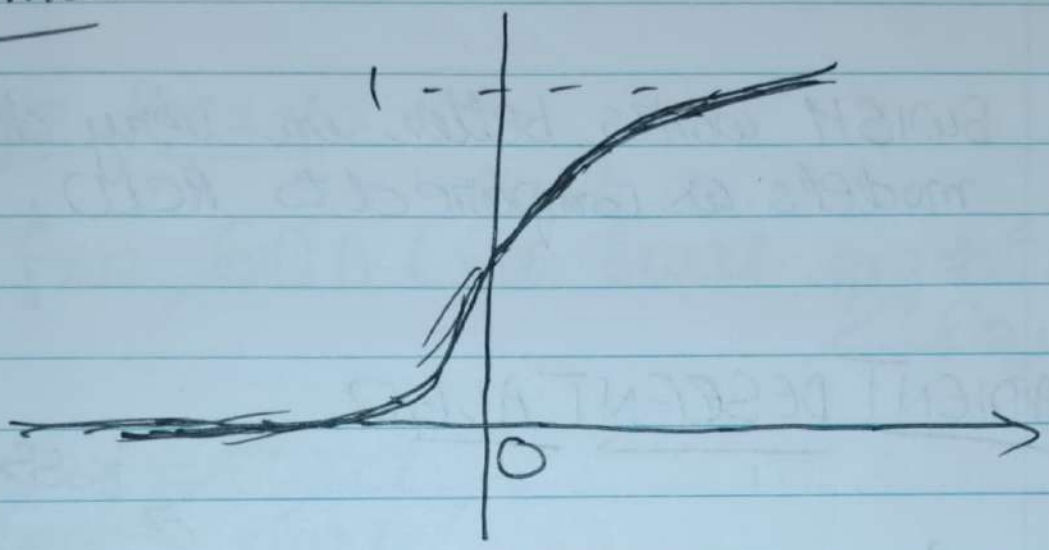
↓
Trainable parameter

Page No.:

ReLU 6

↳ $f(x) = \min(\text{ReLU}, 6)$

Softmax



$$\sigma(z) = \frac{e^x}{\sum e^x}$$

Pros
↳ ① Used for Multiclass classification

Cons (4)
↳ ① Computationally expensive

SWISH

↳ SWISH = x. sigmoid(x)

HARD SWISH

$$h\text{-}swish(x) = x \frac{ReLU6(x+3)}{6}$$

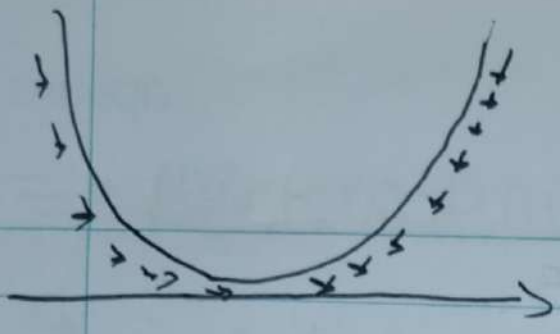
★ SWISH works better in very deep models as compared to ReLU.

GRADIENT DESCENT ALGO

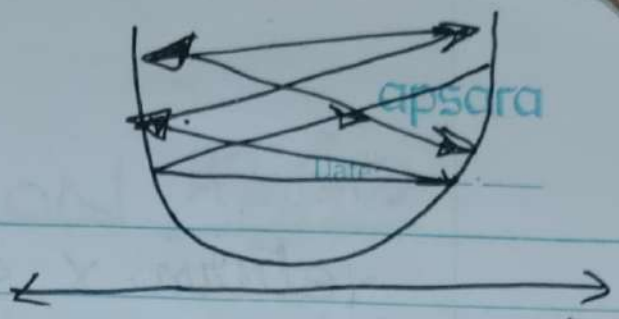
↳ $w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$

(L: loss fxn; η: learning rate)

Loss function = error fxn = cost function



Small learning rate



large learning rate

Algo Pseudo

fxn ~~GDA~~ (~~int~~ float start, gradient (~~float~~ float)

~~int~~ x = start;
~~int~~ steps = start;

~~fxn~~

```
x = start
steps = [start]

for _ in range(max_iter):
    diff = learn_rate * gradient(x)
    if np.abs(diff) < tol:
        break

    x = x - diff
    steps.append(x)
```

→ History Track

return x, steps.

→ BACKPROPAGATION ALGO

(I) Initialize weights randomly

(II) Forward propagate an input to n/w to get the predicted o/p.

(III) Calculate error between predicted o/p & actual o/p.

(IV) Apply BPA to calculate Gradient of loss fxn wrt. each weight

$$\hookrightarrow \frac{\partial L}{\partial \text{weight}}$$

(V) Update wt. using Gradient Descent Algo.

(VI) Repeat 2-5 steps until weights converges

↓

Back Propagation

↙ ↘

BPA + GDA

FOR CALCULATION IN BACKWARD PROPAGATION

apsara

Date: / /

(i) $\frac{\partial E_{total}}{\partial y_{1out}} = y_{1out} - t_1$

(ii) $\frac{\partial y_{1out}}{\partial y_{1in}} = y_{1out}(1 - y_{1out})$

(iii) $\frac{\partial y_{1in}}{w_5} = h_{1out}$

↳ output of hidden layer.

↓

will be used to calculate Gradient for weight attach with outer lay.

$$\frac{\partial E_{total}}{\partial w_8} = \frac{\partial E_{total}}{\partial y_{1out}} \frac{\partial y_{1out}}{\partial y_{1in}} \frac{\partial y_{1in}}{\partial w_8}$$

Page No.:

apsara

For Weight Attach with Input

$$\boxed{\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial h_{1out}} \frac{\partial h_{1out}}{\partial h_{1in}} \frac{\partial h_{1in}}{\partial w_1}}$$

(I) $\frac{\partial E_{total}}{\partial h_{1out}} = h_{1out}(1 - h_{1out})$

(II) $\frac{\partial E}{}$

(II) $\frac{\partial h_{1out}}{\partial h_{1in}} = h_{1out}(1 - h_{out1})$

(III) $\frac{\partial h_{1in}}{\partial w_1} = \frac{\partial}{\partial w_1}(w_1 x_1 + w_2 x_2 + \dots + b)$

(I) $\frac{\partial E_{total}}{\partial h_{1out}} = \frac{\partial E_{01}}{\partial out h_1} + \frac{\partial E_{02}}{\partial out h_1}$

(I) $\frac{\partial E_{01}}{\partial out h_1} = \frac{\partial E_{01}}{\partial y_{1out}} \frac{\partial y_{1out}}{\partial y_{1in}} \frac{\partial h_{1out}}{}$

Page No.:

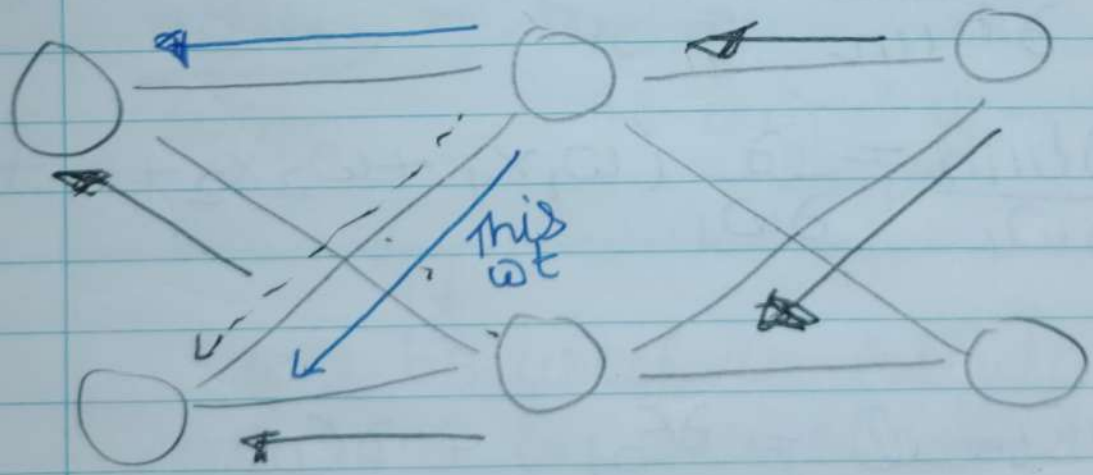
(i) $\frac{\partial E_{01}}{\partial y_{1out}} = y_{1out} - t_1$

(ii) $\frac{\partial y_{1out}}{\partial y_{1in}} = y_{1out}(1 - y_{1out})$

(iii) $\frac{\partial y_{1in}}{\partial h_{1out}}$ ⇒ weight in between them

Illy calculate
↳ $\frac{\partial E_{02}}{\partial out_1}$

this wt



Back Propagation always solve like this

GRADIENT DESCENT

ALGO

SGD
(random some datapoints)
↓
Take one datapoint
↓
Calculate Error
↓ calculate Gradient
Update the weight

BATCH GRADIENT
↓
Take all the sample
↓
Calculate Error
↓
Update the weight

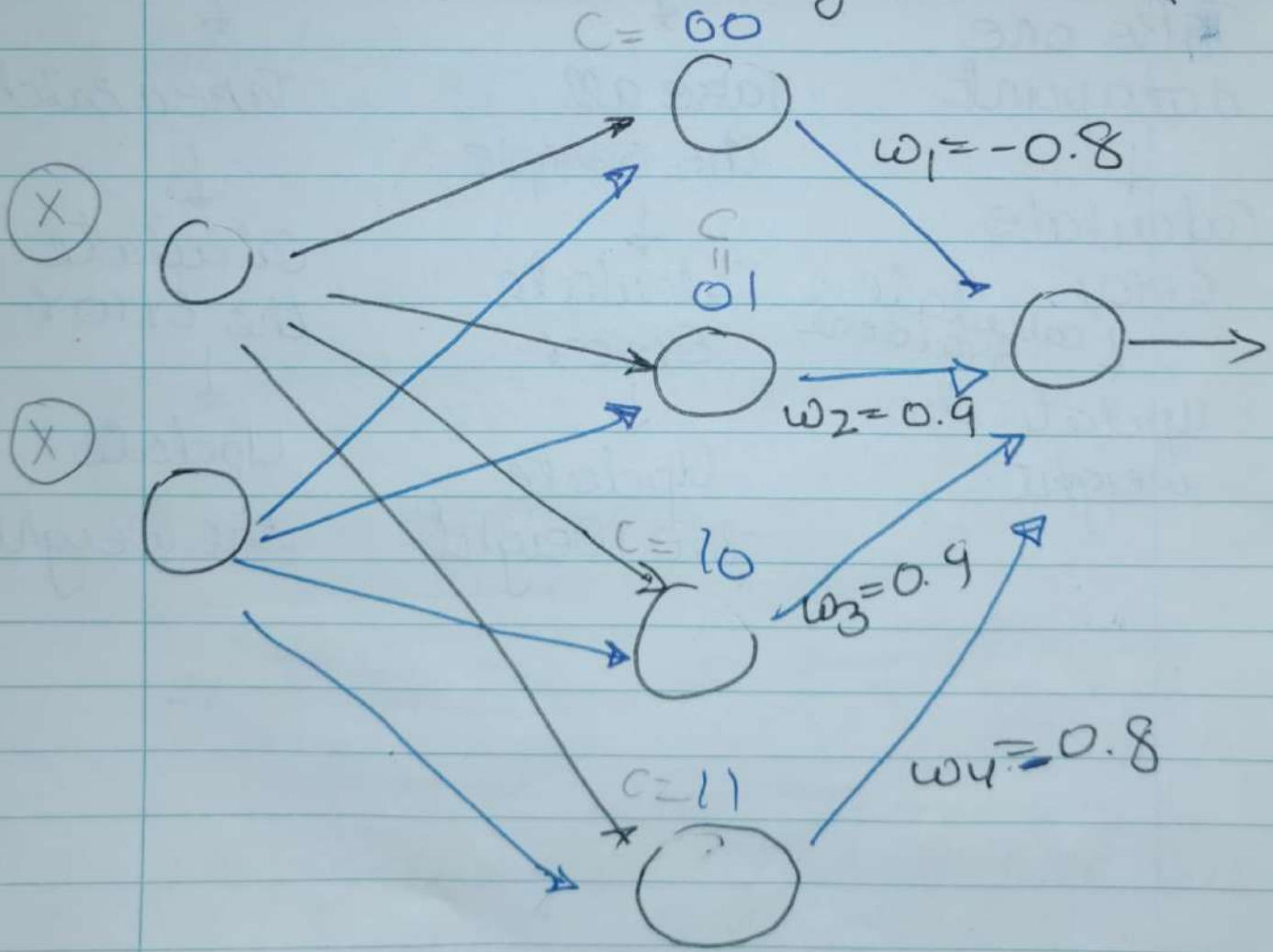
MINI-BATCH GRADIENT
↓
Take a Batch
↓
Calculate the Error
↓
Update the weight

RBF

↳ Gaussian Radial Basis Fxn

↳ $h(x) = e^{-\frac{(x-c_1)^2}{r^2}}$

Construct XOR Gate using RBF Fxn?



Input 00

$h_1(x) = e^{-(x}$

tion

$h_1 = e^{-\frac{(x-c_1)^2}{r^2}}$, $c = (0,0)$

$h_2 = e^{-(x-c_2)^2/r^2}$, $c = (0,1)$

$h_3 = e^{-\frac{(x-(3)^2}{r^2}}$, $c = (1,0)$

$h_4 = e^{-(x-(4)^2/r^2}$, $c = (1,1)$

Input (0,0)

Distance of x from c_1

$(x-c)^2 = (0-\square)^2 + (0-0)^2 = 0$

$h_1 = e^{-(0)/2} = 0$

For input (0,1)

Distance of x from c_2

$= (x - c_2)^2 = (0-0) + (0-1)^2$

$= 1$

$h_2 = e^{-(1)^2} = e^{-0.5} = 0.6.$

$\boxed{h_2 = 0.6}$

Distance of x from c_3

$= (x - c_3)^2 = (0-1)^2 + (0-0)^2$

$= 1$

$\boxed{h_3 = 0.6}$

Distance of x from c_4.

$= (x - c_4)^2 = (0-1)^2 + (0-1)^2$

$= 2.$

$h_4 = e^{-1} = 0.4.$

output => $1 \times -0.8 + 0.6 \times 0.9 + 0.6 \times 0.9 + (0.8) \times 0.4$

$\Rightarrow$ ~~0.3~~ -0.04

o/p => □

For input (1,0)

Distance of x from $c_1(0,0)$

$(x - c_1)^2 = (1-0)^2 + (0-0)^2 = 1$

$h_1 = e^{-1/2} = 0.6$

Distance ~~from~~ x from $c_2(0,1)$

$(x - c_2)^2 = (1-0)^2 + (0-1)^2 = 2.$

$h_2 = e^{-1} = 0.4.$

Distance of x from $c_3(1,0)$

$h_3 \quad (x-r)^2 = (1-1)^2 + (1-0)^2 = 0$

$h_3 = 1$

$\boxed{h_3 = 1}$

Distance of $(1,0)$ from $c_4(1,1)$

$(x-c_4)^2 = (1-1)^2 + (0-1)^2$

$= 0+1 = 1$

$h_4 = e^{-1/2} = 0.6$

$o/p \Rightarrow h_1\omega_1 + h_2\omega_2 + h_3\omega_3 + h_4\omega_4$

$= 0.6 \times -0.8 + 0.9 \times 0.4 + 0.9 \times 1$

$+ 0.6 \times -0.8 = 0.3$

$\boxed{o/p = 0.3}$

$\hookrightarrow o/p = 1$

Ily for rest two

apsara

Date: __ . __ . ____

Hypothesis Testing

↳ is a fundamental statistical method used to make inferences or draw conclusion about a population based on data sample data.

i) NULL Hypothesis (=)

↳ It is general given statement that there is no relationship b/w two measured cases difference

ii) Alternate Hypothesis (≠)

↳ Sum given state

↳ There is a effect or difference

iii) Level of Significance

↳ Degree of significance in which, we accept or reject the Null Hypothesis.

---> Level of significance is selected 5%.
↳ ap should be 95% confident

HYPOTHESIS TESTING
(T-TEST)

apsara

Date: __/__/____

Q =

A random sample of size 20 from a normal population gives a sample mean of 42 & sample standard deviation (S.D) of 6. Test the hypothesis that population mean is 44.

i) Z-Test : sample size (n) > 30

✓ ii) T-Test : sample size (n) ≤ 30

Sol

$n = 20$

$\bar{x}$ (sample mean) $= 42$

$s = 6$

(standard deviation)

mean of population (μ_0) $= 44$

Step 1 : Null hypothesis (H_0) : $\mu = 44$

[i.e population mean is 44]

Page No.: ____

Step 2: Alternative hypothesis (H_1):

★ Technique of Identifying one Tailed & Two Tailed Test

(I) Use one Tailed Test

↳ less than, more than, greater than, smaller than, taller than, superior, inferior, increase, decrease, at least, at most, only, improved

(II) Use Two Tailed Test ⇒ otwise

✓ For Two Tailed Test

↳ $H_1: \mu \neq \mu_0 \Rightarrow \mu \neq 44$.

For one Tailed Test

↳ $H_1: \mu < \mu_0$ if $\bar{x} < \mu_0$

else $H_2: \mu > \mu_0$ if $\bar{x} > \mu_0$

So,

$\mu \neq 44$ [i.e population mean is not 44]

Step 3:

Test statistics: under H_0 test statistics is given by

T-Test Formula

if S.D not given

$$t_{cal} = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}}$$

or

$$t_{cal} = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n-1}}}$$

S.D given Then apply

Variance is given $\Rightarrow$

$S.D = \sqrt{variance}$

apsara

Date: __ . __ . ____

$$\therefore t_{cal} = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n-1}}} = \frac{42-44}{\frac{6}{\sqrt{19}}}$$

$$t_{cal} = -1.45$$

↓

$$|t_{cal}| = 1.45$$

Step 4: Level of Significance (α) = assume

$$5\cdot/. = \frac{5}{100} = 0.05$$

∴ Degree of Freedom (DOF) = $20 - 1 = 19$
$(n-1)$

$|T_{tab}| = 2.093$

↓

from Table
↳ require DOF
& α

Page No.: _____

Date:

Step 5: Comparision & Decision

$|t_{tab}| > |t_{cal}|$

Hence, it is not significant
So, we accept null hypothesis (H_0)
& we reject alternative hypothe
-sis. (H_1)

→ one-Tail Test

Q A Company claims that the mean life of the electric bulbs is 28 months. A random sample of 10 bulbs has the following life in months:

22, 24, 26, 32, 28, 20, 23, 34, 30 & 43

Test the claim of the company at 5% level of significance

Sol $n = 10$ (T-Test)

$\mu_0 = 28$

Calculation of $\bar{x}$ & S

$$\Sigma x = 282 \quad , \quad \Sigma x^2 = 8378$$

mean -> $\bar{x} = \frac{\Sigma x}{n} = \frac{282}{10} = 28.2$

sample

$$S = \sqrt{\frac{1}{n-1}\left[\Sigma x^2 - \frac{(\Sigma x)^2}{n}\right]}$$

mple

$$= \sqrt{\frac{1}{9}\left[8378 - \frac{(282)^2}{10}\right]}$$

$$\boxed{S = 6.87}$$

$n = 10$, $\bar{x} = 28.2$, $S = 6.87$

$u_0 = 28$.

ep 1 : same

One

Step 2: ~~Statistic~~ Two Tailed Test

Step 3:

$$u > u_0$$

$(\because \bar{x} > u_0)$

$$u > 28$$

[i.e mean life of electric bulb greater 28 month]

Step 3: Test Statistics: Under H_0 Test statistics is given by,

$$t_{cal} = \frac{\bar{x} - u_0}{\frac{s}{\sqrt{n}}}$$

$$= \frac{28.2 - 28}{\frac{6.87}{\sqrt{10}}} = 0.09$$

$|t_{cal}| = 0.09$

Step 4: Level of significance : 5%.

Degree of freedom (df) $= n-1$
$= 9$

$|t_{tab}| = 1.833.$

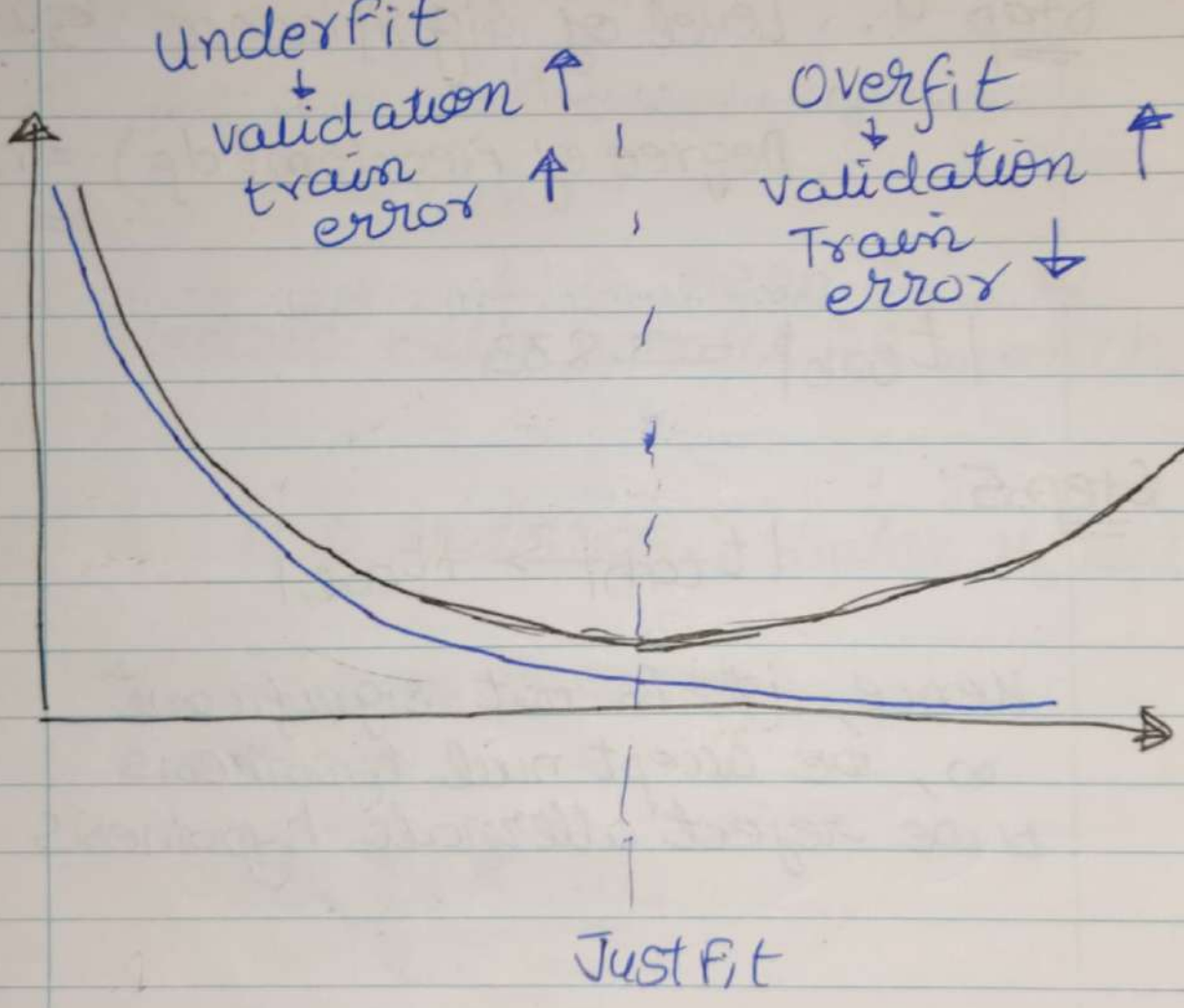
Step 5:

$|t_{tab}| > |t_{cal}|$

Hence, it is not significant
so, we accept null hypothesis.
& we reject allernate hypothesis

ERROR GRAPH

FOR UNDERFIT, JUST FIT & OVERFIT



- In regularization, we reduce magnitude of features by keeping same no. of features

REGULARIZATION

apsara

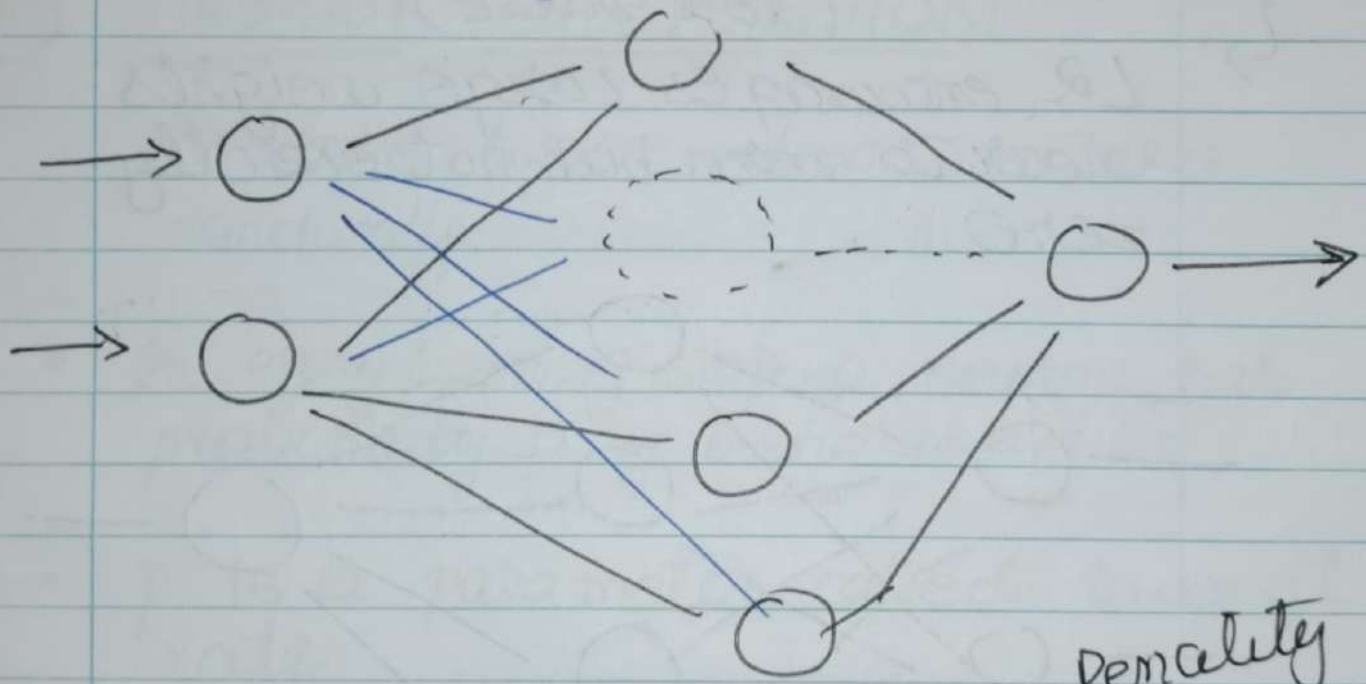
Date: __.__.____

① L1 REGULARIZATION (LASSO)

(Helps in Feature selection)

↳ In this we add absolute value of weights to the loss function

↳ L1 encourages weights to be as low as 0.0, resulting in more sparse n/w.



Penality

Equation:

$$\text{cost} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} |w_i|$$

m = No. of features
n = No. of examples
y_i = Act target value

Page No.:_____

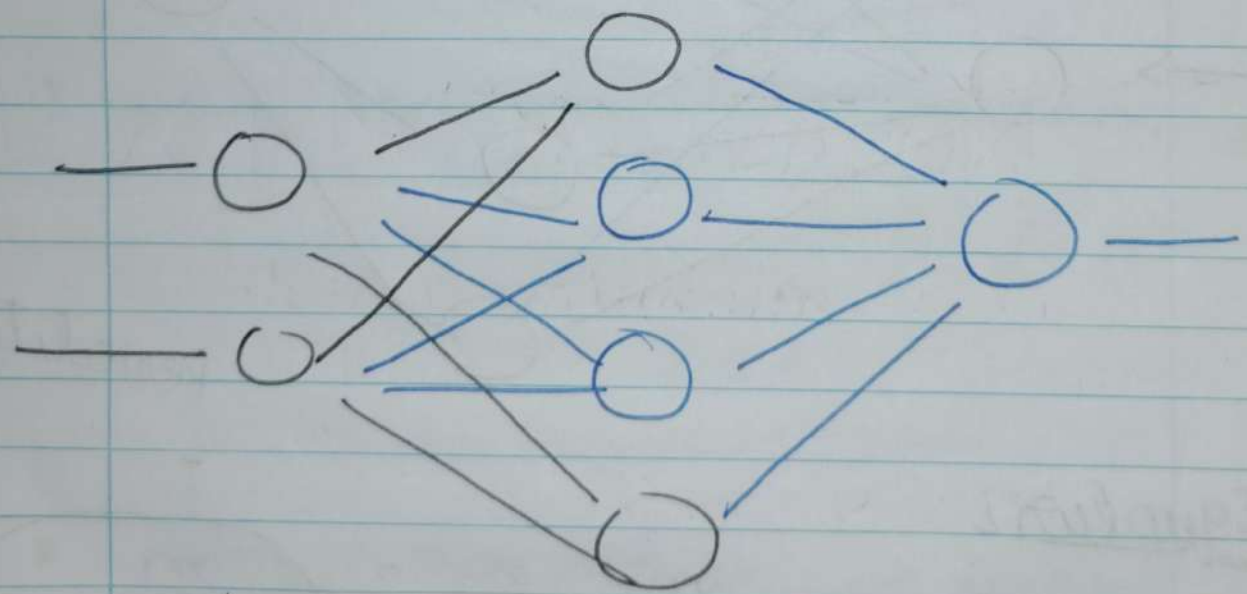
(11) L2 REGULARIZATION

(RIDGE REGRESSION or weight DECAY)

↳ Add the sum of square values of the weight to the loss function

↳ You will not have sparse n/w. (less sparse)

↳ L2 encourages large weights near to zero but not exactly zero



Equation

$$\text{cost} = \frac{1}{n} \sum_{n=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} \omega_i^2$$

no penality
R
-> value
apsara

L Parameter (0 to 1), fully penality

↳ How much attention to pay to this penalty.

- Penality Too strong -> Underfit
- Penality too weak -> Overfit.

(III) DROPOUT REGULARISATION

↳ Make some neurons inactive randomly.

- In every training step, a neuron has probability p of being inactive
- p is a parameter called Dropout rate

(IV) EARLY STOPPING

↳ Stop the learning once the validation error is minimum

(V) DATA AUGMENITATION

↳ Create variations of the data points to enrich the training data

★ BIAS VARIANCE TRADEOFF

↳ Finding a Proper Balance b/w Bias & variance is known as Bias variance Tradeoff.

(2015 introduce)

(VI) BATCH NORMALIZATION

(STANDARDISATION)

NORMALIZATION

↳ Collapse i/p between 0 & 1

Standardisation

↳ making mean 0 & variance 1.

- Standardisation is part of Normaliz-ation

apsara

Date: __/__/____

Types of Normalisation

(I) **Min-Max Normalisation**: Scales the data b/w 0 & 1

$$X_{Norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

(II) **Z-Score Normalization** (Standar-dization): Scales the data to have mean of 0 & S.d of 1.

$$X_{standardized} = \frac{X - \mu}{\sigma}$$

(III) **Max Abs Normalization** → Scales the data between 0 - 1 & 1.

Page No: ____

$$X_{max\text{-}abs} = \frac{X}{|X_{max}|}$$

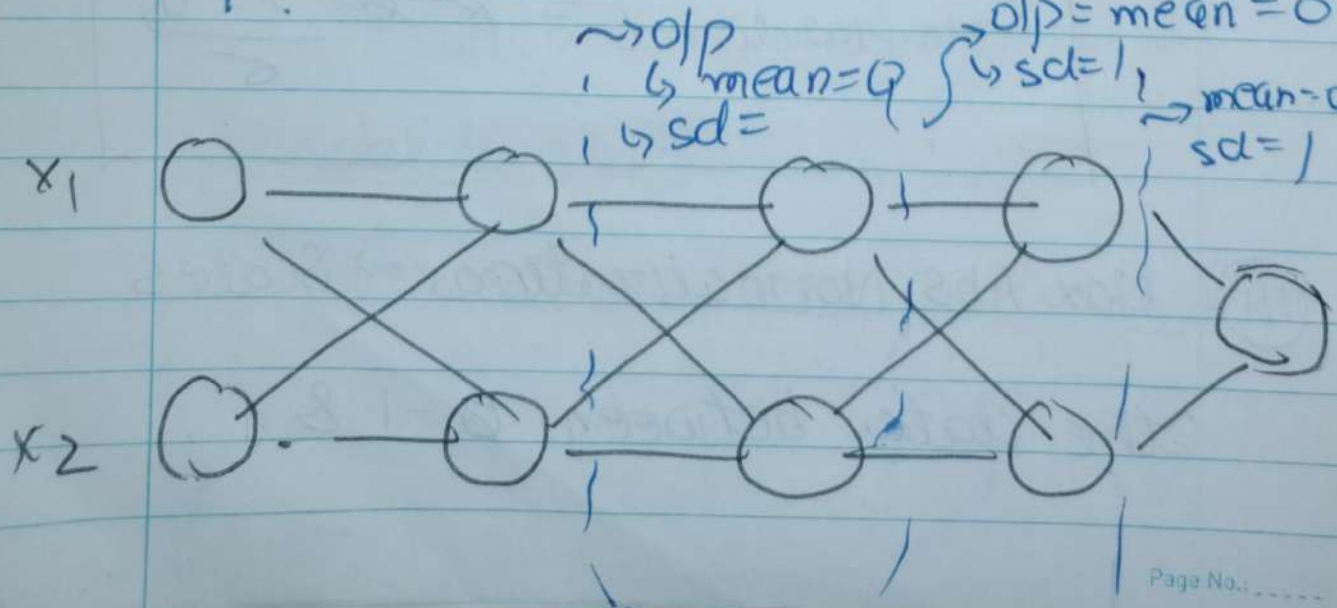
— x — x — x — x — x — x

→ Here is a unique type of Normalization that is used in Batch Normalization i.e

Batch Normalization

$$\Rightarrow \hat{x}(i) = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma^2 + \epsilon}}$$

→ Batch Normalisation have mean 0 & standard deviation 1.



How BATCH NORMALIZATION WORKS

→ Output of each hidden layer is standardized to have mean of 0 & variance 1 in order to achieve faster convergence

Steps

↳ (I) Takes a mini-batch
↳ (i) Calculate mean
(ii) Calculate variance

(IV) $$\mu_B = \frac{x_1 + x_2 + x_3 + \dots}{n}$$

variance ⇒ $$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$$

(II) Normalize the batch

(III) Scale & Shift to ensure model can still learn effectively

$$\boxed{y_i = \gamma \hat{x}_i + \beta}$$

γ -> gamma } are trainable
β -> Beta. } parameter

apsara

Date:

★ weights need regularization where biases don't

L1 Regularization (Lasso)	L2 Regularization (Ridge)
(I) Penality term is based on the abso-lute value of model parameter	(I) The penality term is based on the squares
(II) Produce Spare Solution	(II) Produce non-sparse solution
(III) Selects the subsets of the most imp. features	(III) All features are used by the model
(IV) Sensitive to outliers	(IV) Robust to outliers → that's why used significantly

Q Why L1 Regularisation is sensitive to outlier whereas L2 is not?

Page No.:

NORM PENALTIES AS CONSTRAIN -ED OPTIMIZATION

- Constraint Optimization is about find the best solution to problem while sticking to certain rules or constraint

Constraint Problem

↳ Want to minimize cost but also need to make sure you don't exceed budget

→ To solve these types of problem we use lagrange fxn

$$L(\theta, \lambda) = J(\theta, x, y) + \lambda(S(\theta) - k)$$

const.

$J(\theta, x, y)$: only cost fxn

$S(\theta)$: Penalty constraint

$$\theta^* = \arg\min_{\theta} \max_{\alpha, \alpha \geq 0} \mathcal{L}(\theta, \alpha)$$

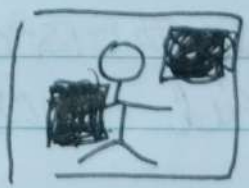
- ## DATA AUGMENTATION

↳ Basically adding more synthetic data for training

- Main Data Augmentation Method

↳ (I) Flip
(II) rotate (1° to 3°) => may
(III) crop
(IV) Color shift
(V) Noise Addition (Breast Cancer)
(VI) Constrast change
(VII) Information loss.

↳



- ## NOISE ROBUSTNESS => (Noise Injection)

i) Adding Noise to input
ii) Adding Noise to Hidden layers
iii) Adding Noise to Weights → used in RNN
iv) Injecting Noise

- Semi Supervised LEARNING

↳ Technique that combines a small amount of labelled with larger amount of unlabelled data during training

↳ This is a form of regularization as its improve the model generalization ability

- Unsupervised learning can provide useful clues how to group

EARLY STOPPING

- Maxout networks, being more flexible than ReLU, can learn more complex features but are still susceptible to overfitting

Q How EARLY STOPPING WORKS?

Epoch 1 : validation loss ↓ : Save the parameter

Epoch 2 : Val loss ↓ : Update & save Parameter

Epoch 3 : Val loss ↓ : Update & save Parameter

Epoch 4 : Val loss ↑ : Don't Update

Epoch 5 : Val loss ↑ : " "

Epoch 6 : Val loss ↑ : Early stopping.

★ Mostly used in Deep learning.

apsara

Date: __/__/____

--> Early stopping is a regularization technique used in ML to prevent overfitting by halting the training process once model validation loss start ping.
It monitor the performance during training, & when no significant improvement is observed for certain no. of iterations or epochs, training stopped & the model is then restored to the state where it performed best on val. dataset.

↳ During training, it saves parameter whenever val. loss ↓.

USE OF 2nd TRAINING STEP IN EARLY STOPPING

↳ Early stopping requires validation set.
  ↳ Thus some training data is not fed into model

★ Better utilisation of entire dataset

apsara

Date: __.__.____

To make use of this extra data, a second training step can be conducted after initial training with early stopping.

- 2nd training phase will include previous + valid. data (entire dataset)

TWO BASIC STRATEGIES FOR 2nd TRAINING PROCEDURE

(I) Fine Tuning (II) Retraining from Scratch

(I) We use pretrained parameters obtain-ed from initial training & then model is trained on few more epochs on full dataset for further improved.

(✗) Fine Tuning allow model to leverage the knowledge gained from the initial training while adjusting to new datapoints

★ Better utilisation of entire dataset

apsara

Date: ___.___.___

To make use of this extra data, a second training step can be conducted after initial training with early stopping.

- 2nd training phase will include previous + valid. data (entire dataset)

TWO BASIC STRATEGIES FOR 2nd TRAINING PROCEDURE

(I) Fine Tuning (II) Retraining from Scratch

(I) We use pretrained parameters obtain-ed from initial training & then model is trained on few more epochs on full dataset for further improved.

(✓) Fine Tuning allow model to leverage the knowledge gained from the initial training while adjusting to new datapoints

→ Not really used

apsara

Date: __/__/____

(11) Retrain the model from scratch means discarding the parameter update from first training & retrain the new model on whole dataset

↳ if validation dataset is too small (is one case, where we do this)

Early Stopping

↳ (I) Early stopping effectively controls capacity of the model by limiting the number of training iterations

↳ (II) It provide straight forward to regularize model w/o altering its str. or adding complexity

PARAMETER TYING REGULARISATION

↳ In this, instead of each layer in neural n/w having its own set of parameters (weights), certain

layers share same parameters.

PARAMETER SHARING

↳ refers to practice of using the same set of parameters (weights) accross different layers of a network.

In CNN, the kernel parameters are shared

BAGGING REGULARISATION

↳ Bagging involve training multiple models on different subsets of the training data & then aggregating their predictions to produce final output.

Bagging → Bootstrap aggregating

- Use concept of Model Averaging.

~~DROPOUT REGULARIZATION~~

PARAMETERS	HYPERPARAMETERS
(i) These are the entities learned via training from training data.	(i) These are the entities set by the designer before training
(ii) They are set manually by designer	(ii) Example → learning rate No. of layers No. of neurons in each layer Activation fxn
(iii) Example → Weight & Biases	

HYPERPARAMETER TUNING

↳ (i) optimizer
↳ (ii) Learning rate
↳ (iii) Batch size

→ Tuning these manually makes you to try out 1000 more combination

Approaches for Hyperparameter Tuning

(i) Grid Search
- ↳ Try out every single option
- ↳ very slow

(ii) Random Search (used mostly)
- ↳ Try out a subset of all the options
- ↳ Works well for many problems

(iii) Zoom In

CODE → Little Bit Difficult.

Evaluation of model

TRAINING SET	CROSS - VALIDATION SET	TEST SET

Weight & Biases

Model Selection (Hyperparameter Tuning)

apsara

ENSEMBLE TECHNIQUE

Bagging
(Feeding Data Ualy to Different model)

↓

Random Forest

BOOSTING
(Feeding data sequentially to a model)

data → O → O → O → O

↓

i) AdaBoost
ii) Gradient Boost
iii) XG Boost
iv) CATBOOST

Q = Why we use the momentum in the overall updations of weight -s in N.N?

↳ So, according to momentum is used in SGD or weight updation for speed up & faster convergence what momentum do is add previous fraction of update) to the current update

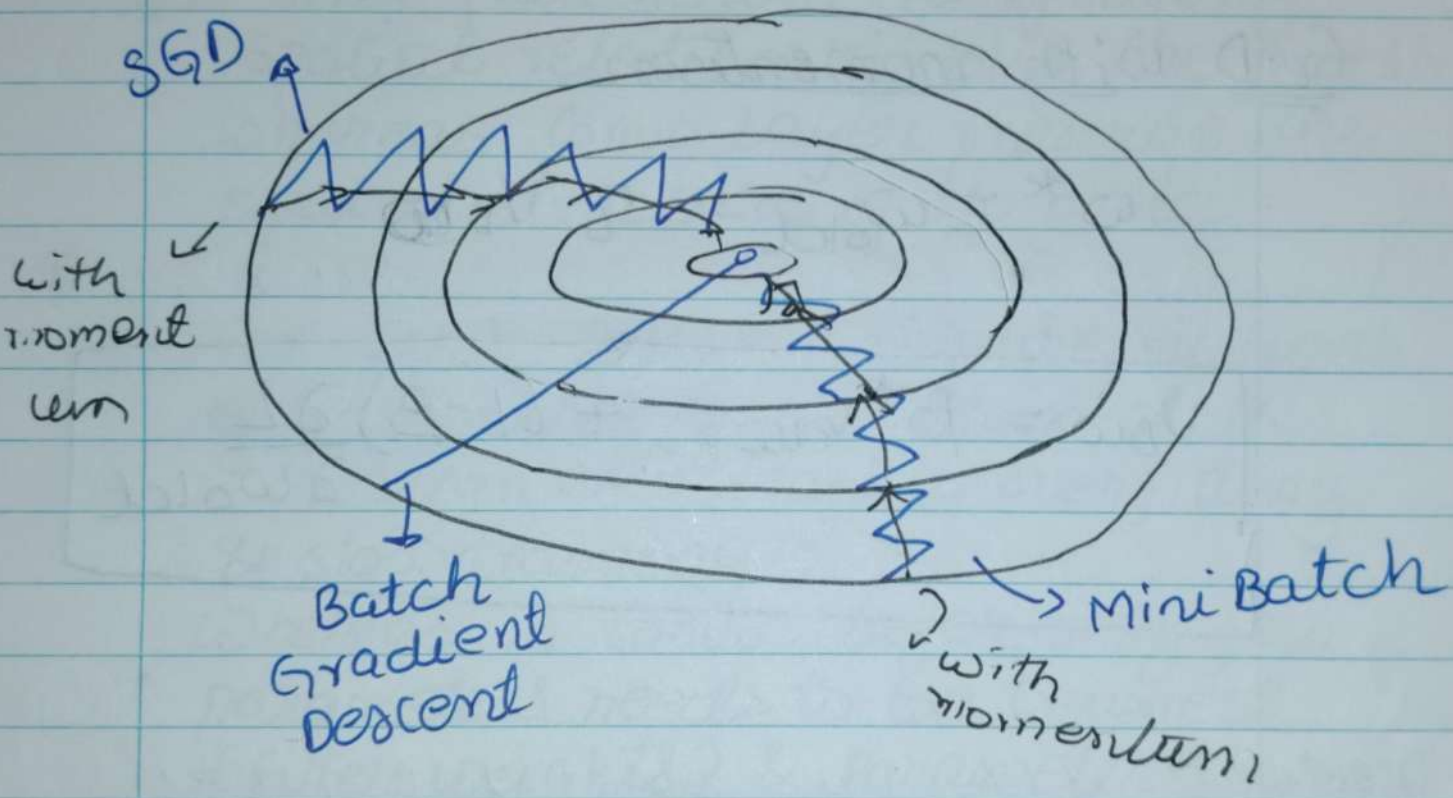
SGD

~~SGD~~ with Momentum

apsara

Date:

- ↳ aim is to remove vertical movem -ent & improve the horizontal movement.



↳ lower weightage is given to previous datapoint & higher to current.

- $\boxed{V_t = \beta * V_{t-1} + (1-\beta)O_t}$ ← current data point

$V_0 = 0$

$V_1 = \beta V_0 + (1-\beta)O_1$

$V_2 = \beta V_1 + (1-\beta)O_2$

Page No:

Gradient Descent w/o Momentum

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

G.D with momentum

$$w^* = w_{old} - \eta V_{dw}$$

$$V_{dw} = \beta^* V_{dw_{t-1}} + (1-\beta) \frac{\partial L}{\partial w_{old}}$$

apsara

Date:

Why Convo. layer preferred over fully connected NW (ANN) in CNN?

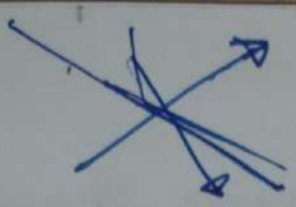
(1) ANN flattens the i/p ignoring spatial relationship b/w the pixels whereas Convo. Layer preserve the spatial structure of i/p data.

(2) For CNN, ANN will require millions of parameter to be trained & that can also lead to overfitting & slow training
Whereas in Convo. layer only few parameters needs to be trained (filter weights) & Parameter sharing also leads to massive reduction in no. of parameters

CNN => computationally efficient
ANN => Not

Why Batch Normalization is required for both input as well as hidden layers?

Page No.:



(i) Improve Stability
(ii) Performance
(iii) Slow Convergence
(iv) Vanishing / Exploding Gradient Problem
(v) Overfitting

Q How to get original data back from PCA?

↳ Eigen vector * (Eigen value)^T

(i) zero mean* = Eigen vector ⊗ (Eigen value)^T
P.C value

(ii) zero mean = zero mean^{*T}.

(iii) zero mean
original data = zero mean + mean value

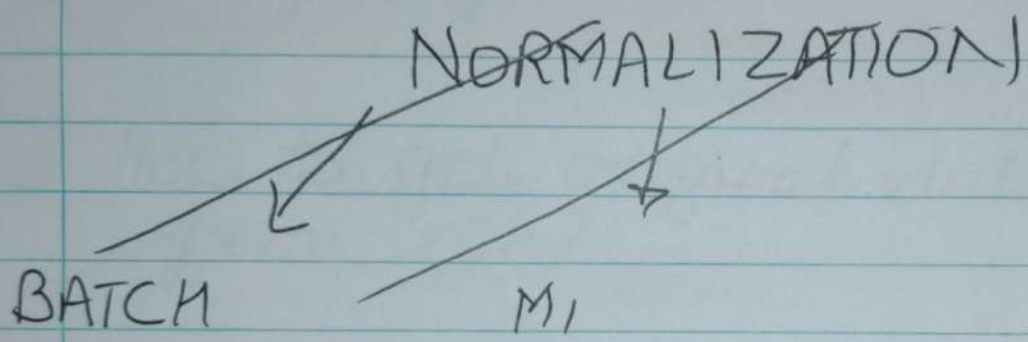
apsara

Date: __ . __ . ____

INTERVIEW

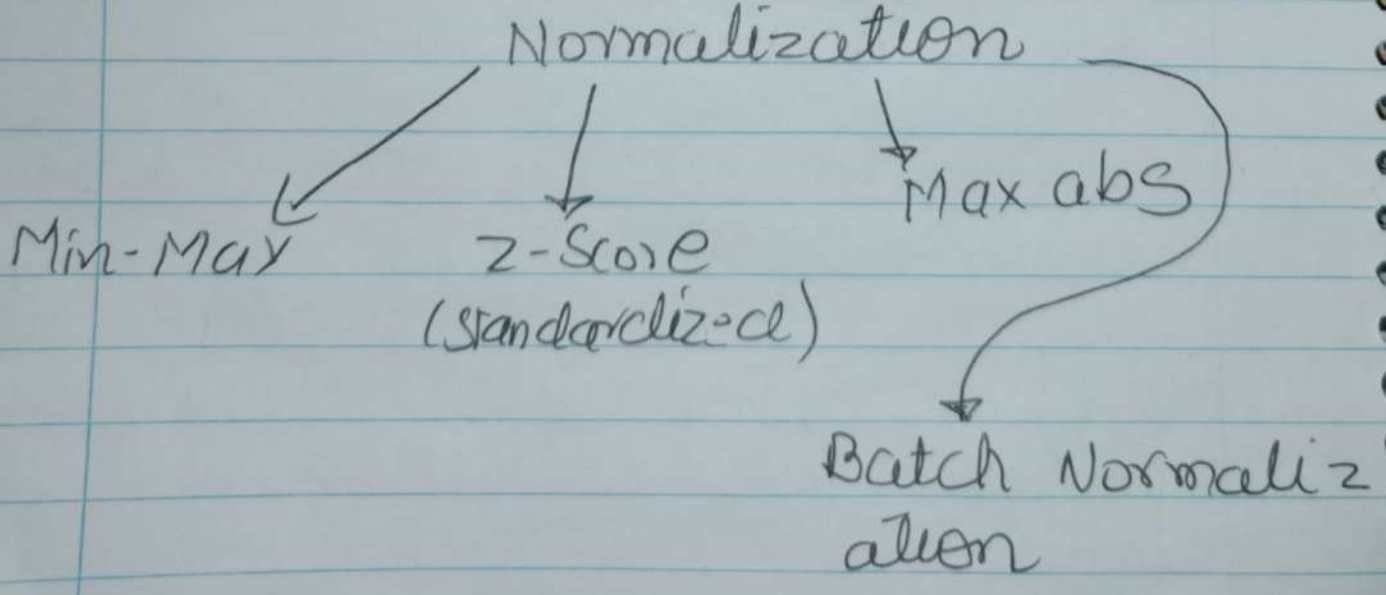
Zero Padding : layer used in CNN

to add zero values around the border of a 2D input.



- Normalization : Process that

adjust data to a standard scale or distribution



apsara

Date: __.__.____

---> Batch Normalization => Mean 0 & standard Deviation 1.

↳ o/p of each neuron is standardize to have mean 0 & variance 1 for faster convergence

★ Reduces the number of iterations required during training

- Gaussian Blur: Smoothens the image by averaging pixel intensities with their neighbour.

- What is F1 score?

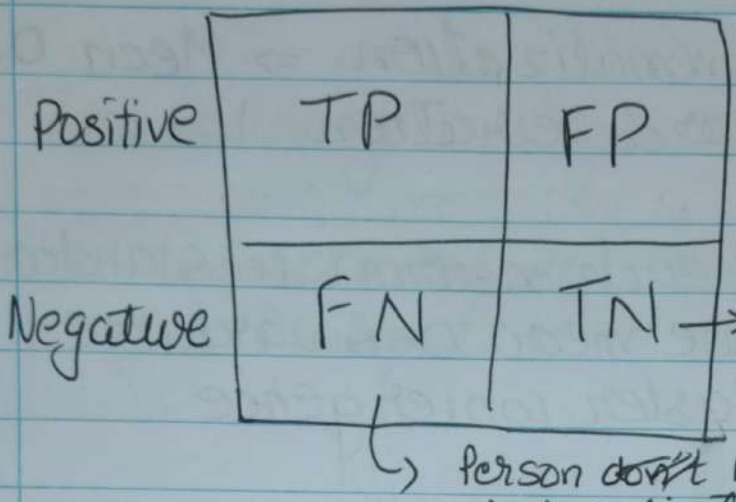
Ans It is a metric used to evaluate performance of a classification model

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Page No.: ____

★ FP is very dangerous
↳ Person don't have cancer but detected

• CM



(I) Accuracy ⇒ $\frac{TP + TN}{TP+TN+FN+FP}$ (not)

(II) Precision: Performance metric used to evaluate the accuracy of model's +ve prediction

Precision: $\frac{TP}{TP+FP}$

(III) Recall: $\frac{TP}{TP+FN}$

↳ T + F

★ FN is dangerous.

Hyperparameter

Learning Rate: Learning rate defines how fast or slow model converges to global minima.

> Step size => control the size of steps taken to update the model's parameter

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

(η: learning rate)

Loss function: MSE

For Regression, Task generally one output neuron is required

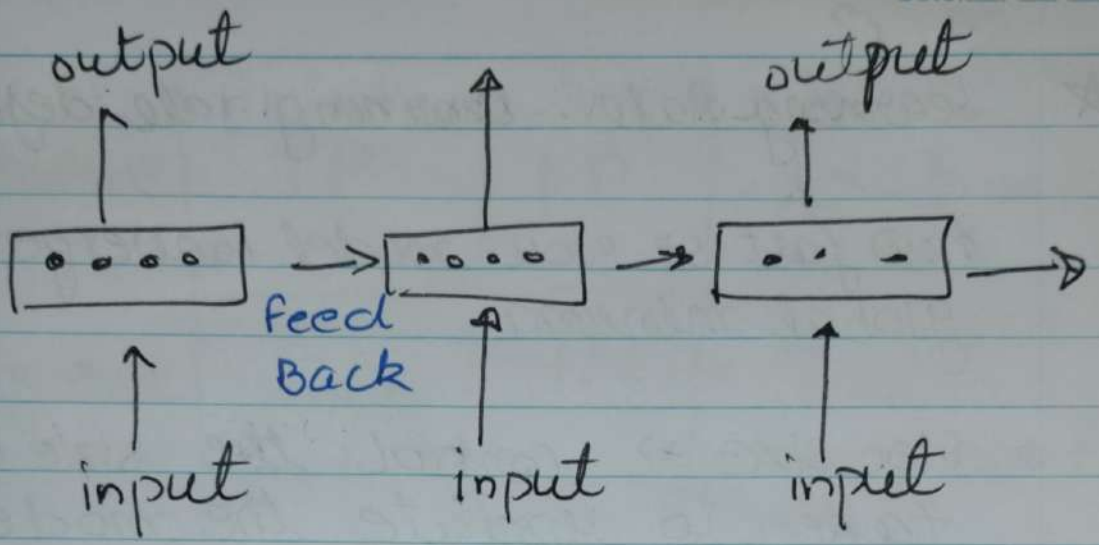
ANN => is a Fully connected NN

↓

output of each previous layer is given input to every next neuron

RNN

RNN are also trained just like NN

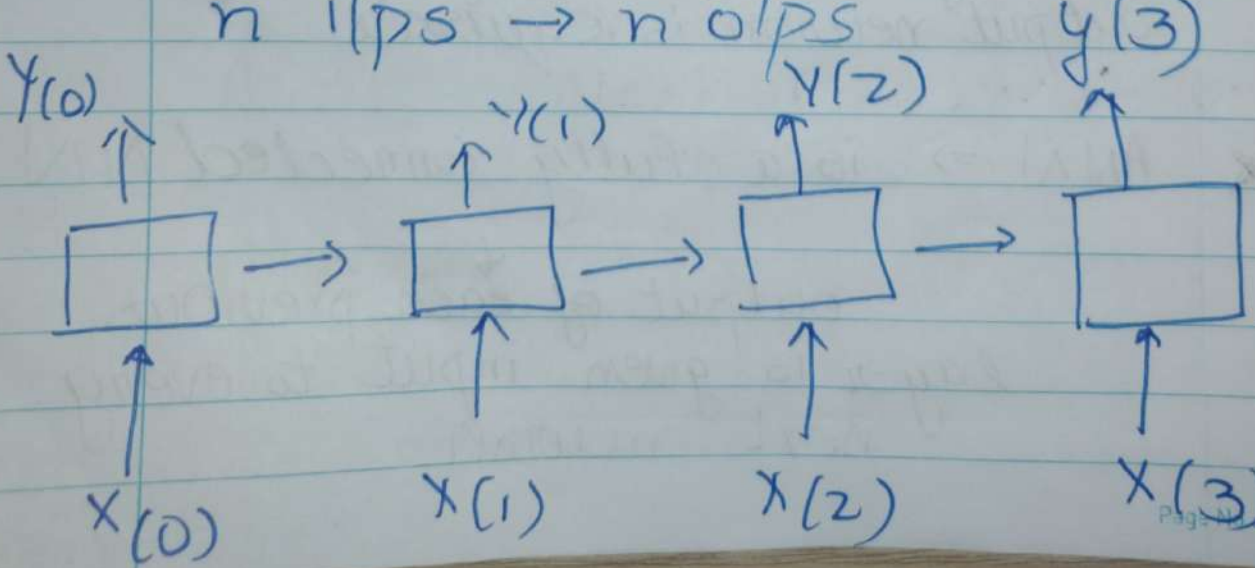


★ Output of previous layer is given as feedback to next layer.

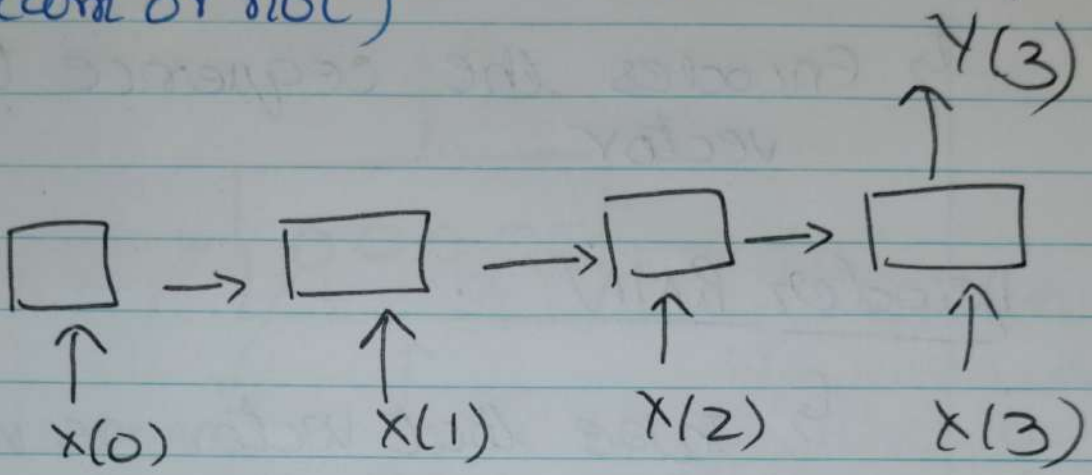
Types of RNN

① Sequence to Sequence:
(Price forecasting)

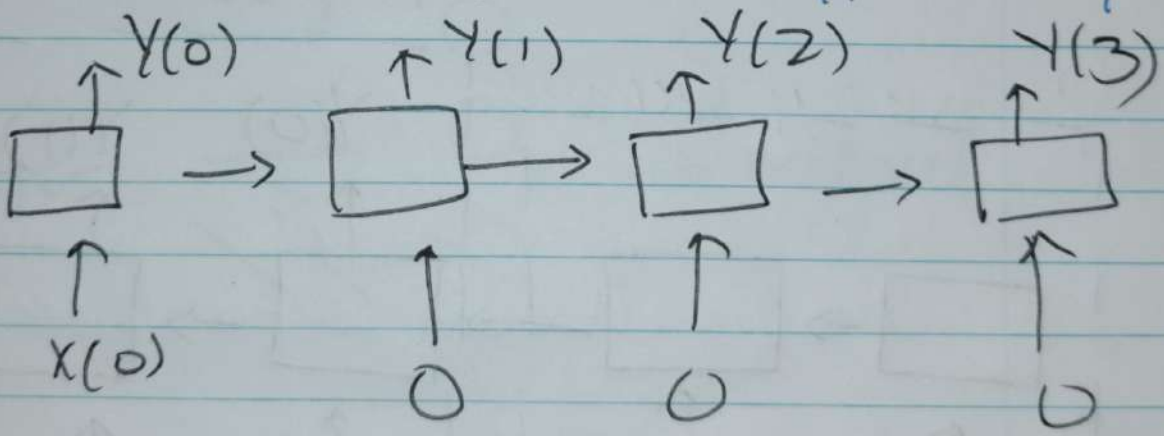
n i/ps → n o/ps



(II) Sequence to vector: n i/ps → 1 o/p.
(Scam or not)



(III) Vector to Sequence: 1 i/p → n o/p



(IV) Encoder - Decoder:
(Translation

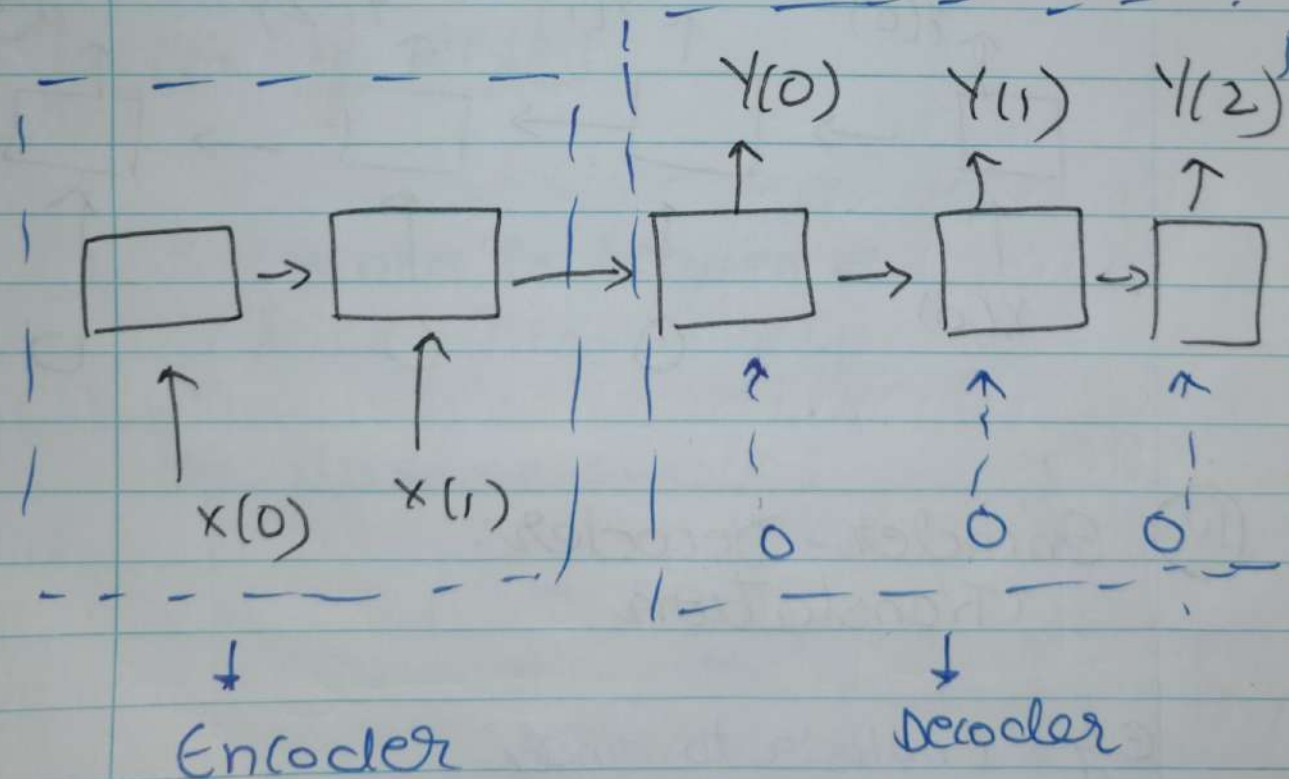
E.g → English to Hindi

Encoder RNN

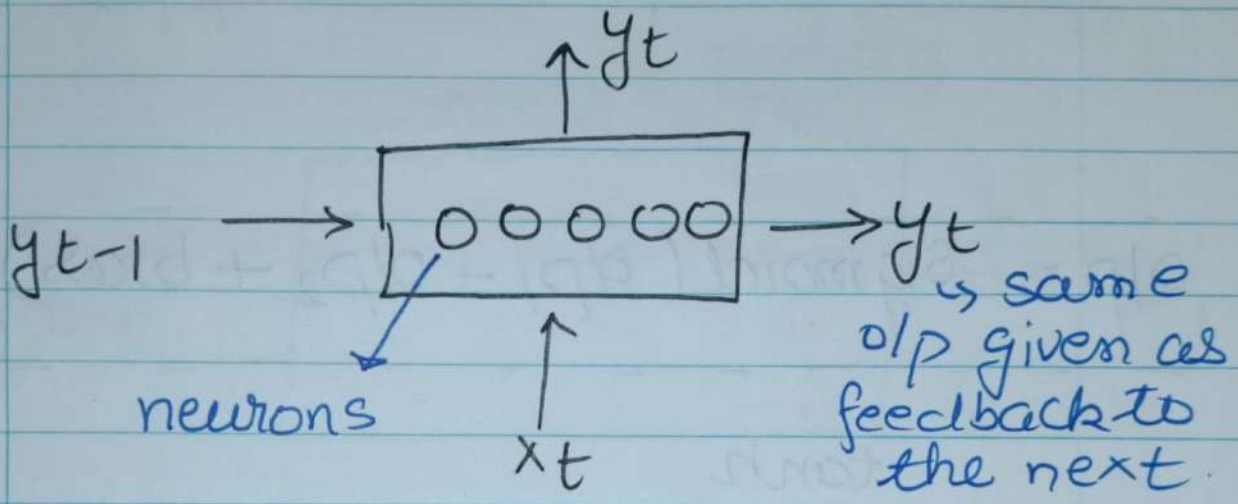
↳ Encodes the sequence to a vector

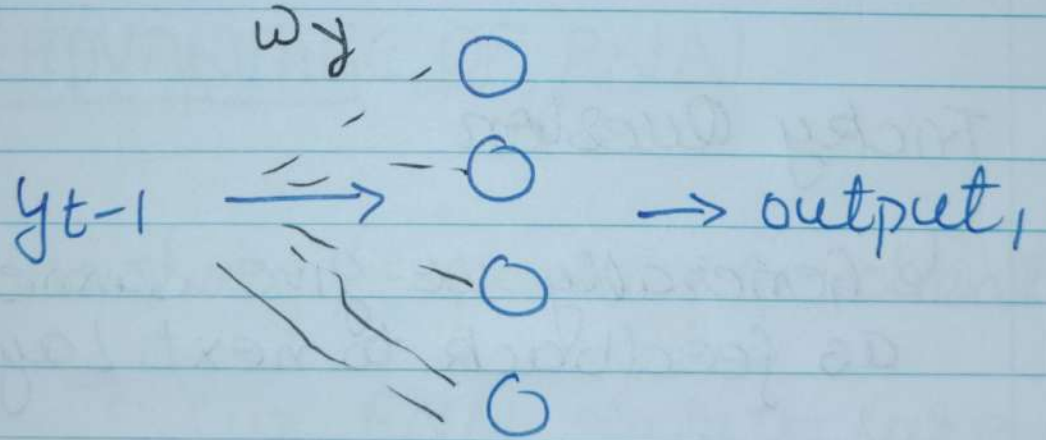
Decoder RNN

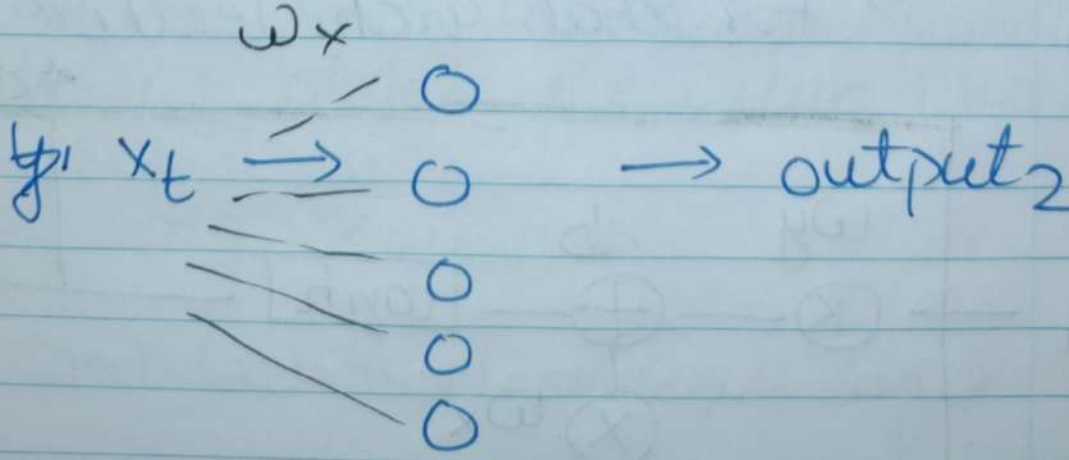
↳ Take that vector as input
↳ Reconstruct the original sequence or generate a related o/p sequence



Q How output is calculated in RNN?







★ You can think it as double neuron connection, one separate connection with i/p & one with feedback

output = activation fxn (output$_1$ + output$_2$ + bias)

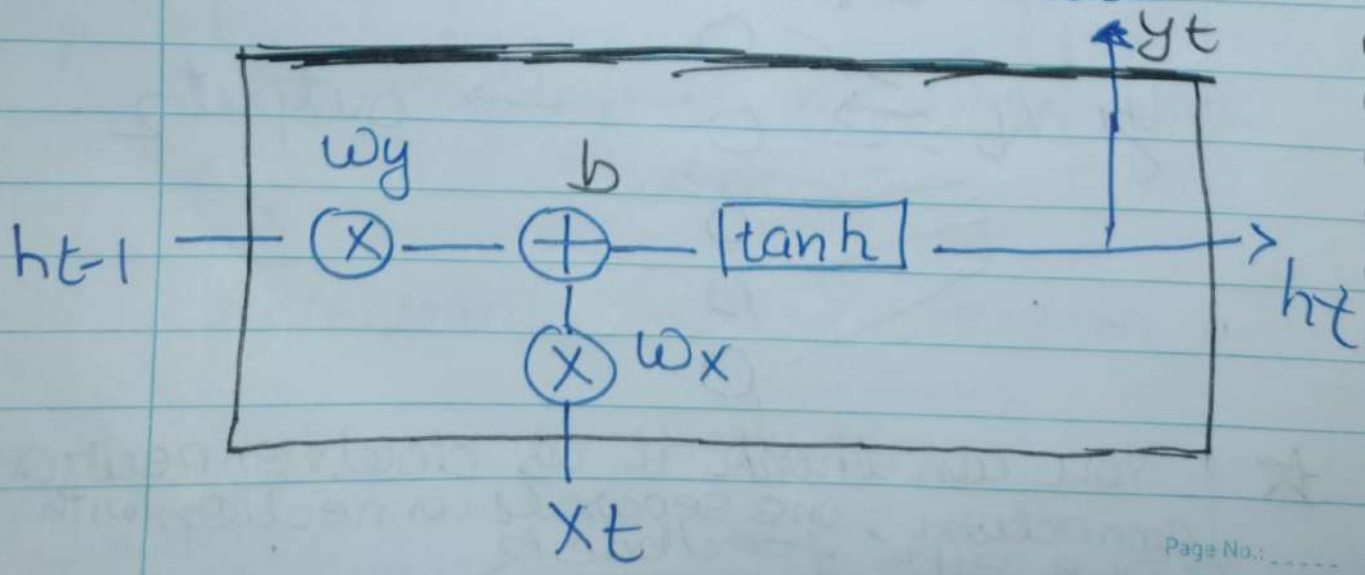
o/p = sigmoid (o/p$_1$ + o/p$_2$ + bias)

tanh
ReLU.

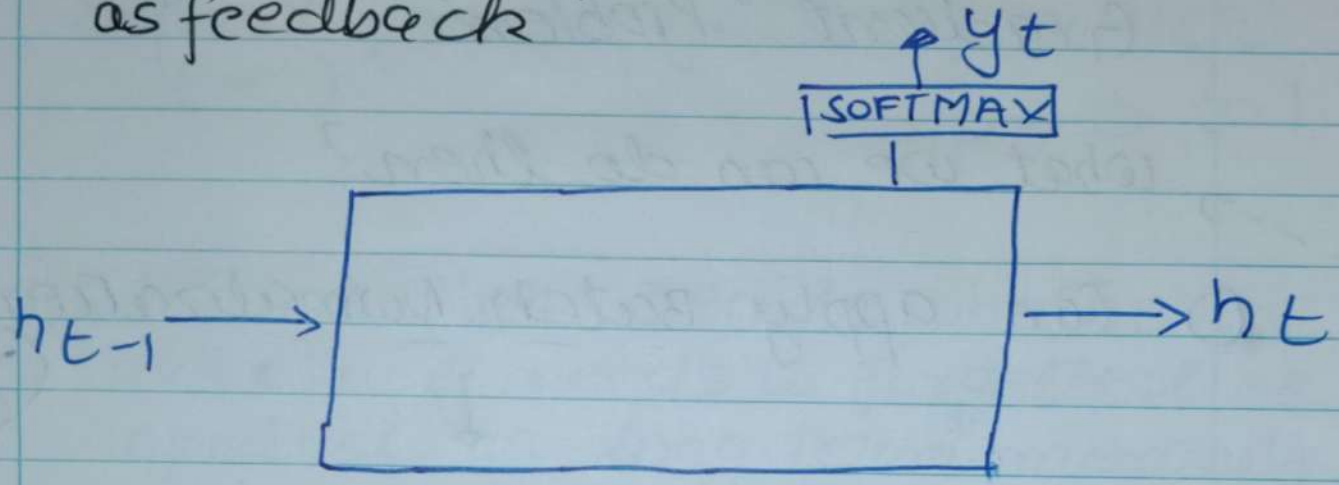
Tricky Question

↳ Generally we give same o/p as feedback to next layer

↳ For that architecture



→ In this case o/p will not be same as feedback



DISADVANTAGE OF RNN

① Vanishing / Exploding Gradient:

↳ Cuz RNN architecture have Tanh activation function which shrink the value b/w -1 & 1.

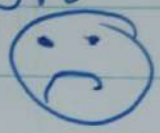
↳ Long Sequences.

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

→ RNN also face exploding Gradient Problem

--→ What we can do then?

① Can apply Batch Normalization



↓

ⓘ normalize i/p across each layer

↓

mean 0
variance 1

ⓘⓘ Improve Training Speed

ⓘⓘ Long Term Memory does not work well:

They forget the beginning of long sequences:

--→ What we can do?

↳ use LSTM or GRU.

↱ o/p of LSTM => ht

LSTM

↱ every thing attach with σ is Gate

→ Sigmoid fxn → gate controller
(σ) ↓
o/p => 0 to 1

gate controller: Decide whether something needs to forgetted or inputed in long term memory.

- Forget Gate: which part of long term memory should be removed.
- Input Gate: which part of the new info we got should be added to long term memory.
- output gate: Which part of long term state should be added to the o/p at this timestamp.
- Sigmoid → 1 → Let everything pass
- Sigmoid → 0 →

apsara

Date: / /

---> Check on You.

LSTM can be used in:

(i) Natural language

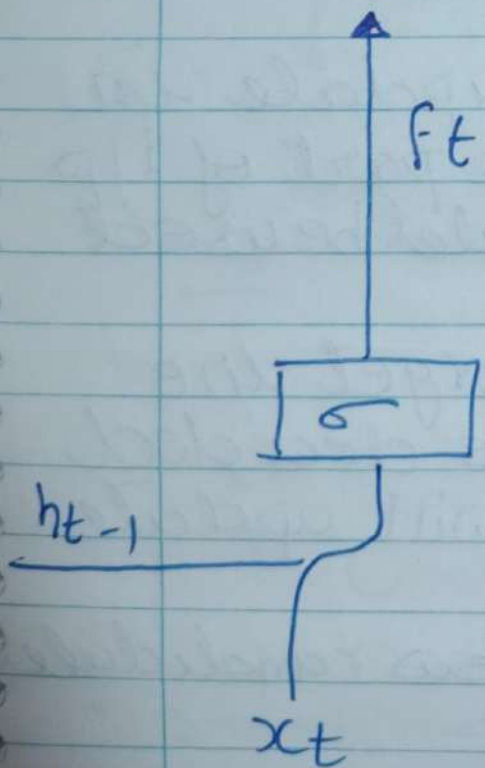
(ii) Time Series Data:
- Stock Price
- Energy usages

★ where the gap blw the relevant info & predict position is small, RNN can work.

Q where RNN can't work but LSTM can?

I grew up in France.....
... I speak fluent ?

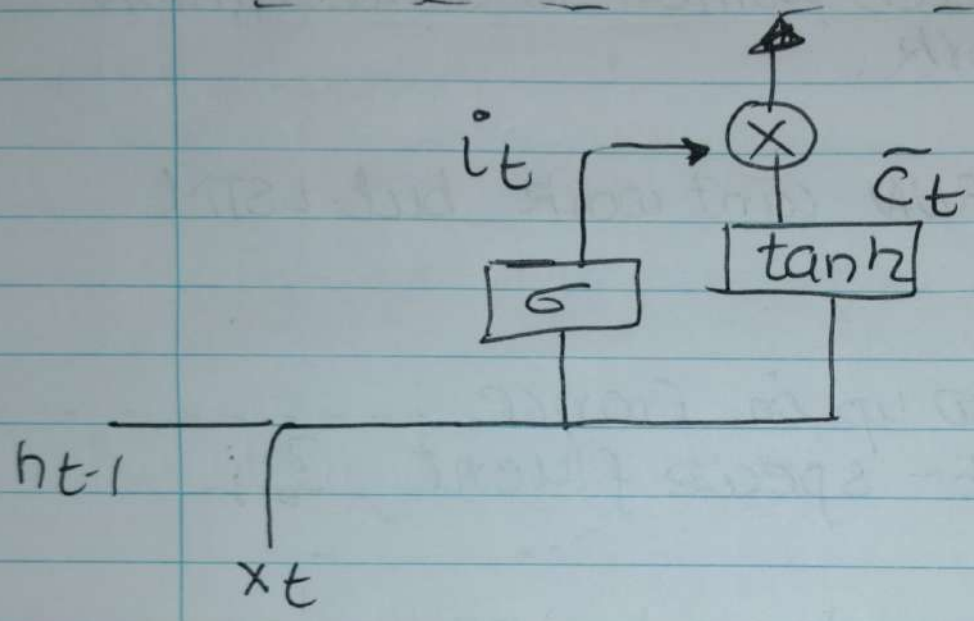
Forget Layer =>



$$f_t = \sigma(W_f^x x_t + W_f^h h_{t-1} + b)$$

Date: __ / __ / ____

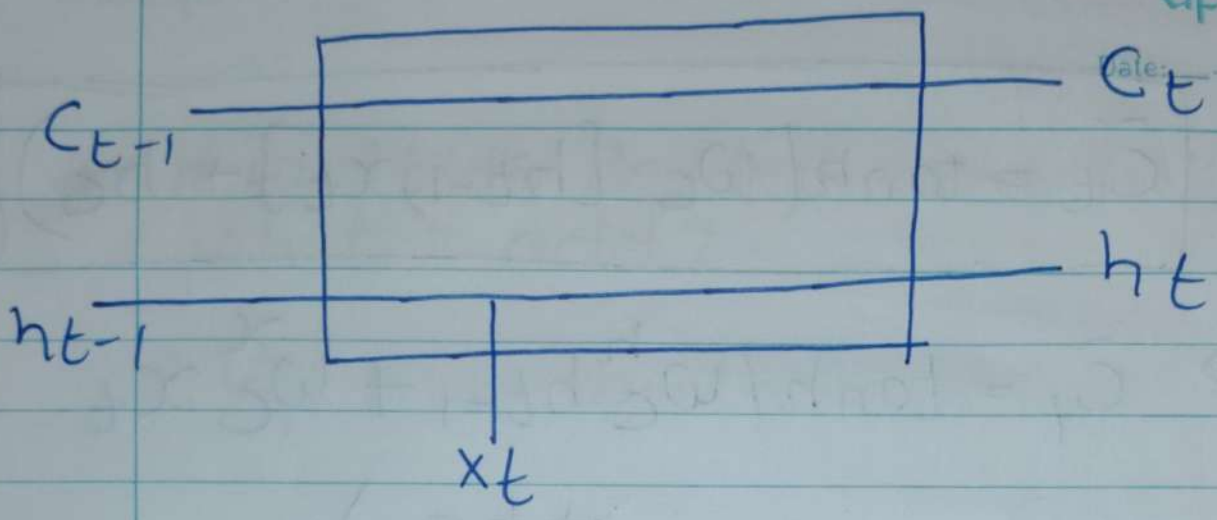
Input gate layer



→ i_t → Input gate layer decides which value we will update in memory cell & which part of i/p & previous hidden state should be used to update

Till Now, we have forget the things we want, we decided which things we will update

→ $\hat{C}_t$ ⇒ a vector of new candidate



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$\begin{bmatrix} 1\ 0\ 0\ 1 \ldots \\ 0\ 1\ 0 - - \\ 0\ 1\ 1\ 0 \ldots \\ \ldots\ldots \end{bmatrix} * \begin{bmatrix} \text{Previous} \\ \text{info.} \end{bmatrix}$$

- $i_t * \tilde{C}_t$

$$\begin{bmatrix} 0\ 1\ 1\ 0 \ldots \\ 1\ 0\ 1\ 0 \ldots \\ 1\ 0\ 0\ 1 \ldots \end{bmatrix} * \begin{bmatrix} \text{New} \\ \text{info.} \end{bmatrix}$$

New info. we can add.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$\tilde{C}_t = \tanh(W_C^h h_{t-1} + W_C^x x_t + b_C)$$

Forget Layer

$$f_t = \sigma(W_f^h h_{t-1} + W_f^x x_t + b_f)$$

can be written as

$$\text{ie } f_t = \sigma(W_f [h_{t-1}, x_t] + b_f)$$

Input Layer

$$i_t = \sigma(W_i^h h_{t-1} + W_i^x x_t + b_i)$$

can also

$$i_t = \sigma(W_i [h_{t-1}, x_t] + b_i)$$

★ Back Progation in DL (Gate)

apsara

Date: ___.___.___

★ How it is decided what to forget what to add?

↳ W_f & W_i are trained

Output Gate

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

final o/p

$$\boxed{h_t = o_t * \tanh(C_t)}$$

↳ Check in DL (Gate)

NLP

— × — × — × — × — × — × — ×

There are Two Models to Train Word Embeddings

Word2Vec

Glove

Word 2 Vec

apsara

Date: __/__/____

→ To train word Embedding via word2vec we use Neural Network

Dataset (Corpus)

The future King is the Prince
Daughter is the princess
Son is the prince
only a man can be King
only a woman can be queen
The - - - - - -
- - - - -

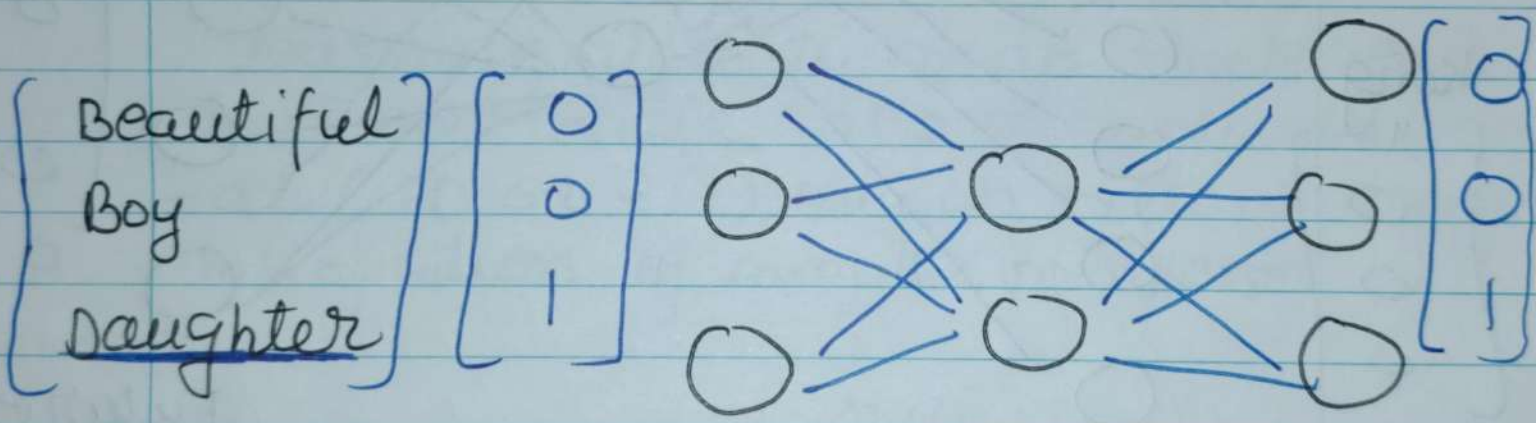
(I) Remove Stop Words

future King Prince
Daughter princess
Son Prince
man King
Boy Man

(II) Create Bigrams
↳ Two consecutive words occuring in a sentence

[future, King]
[King, Prince]
[Daughter, princesses]

(III) Convert words to one hot.



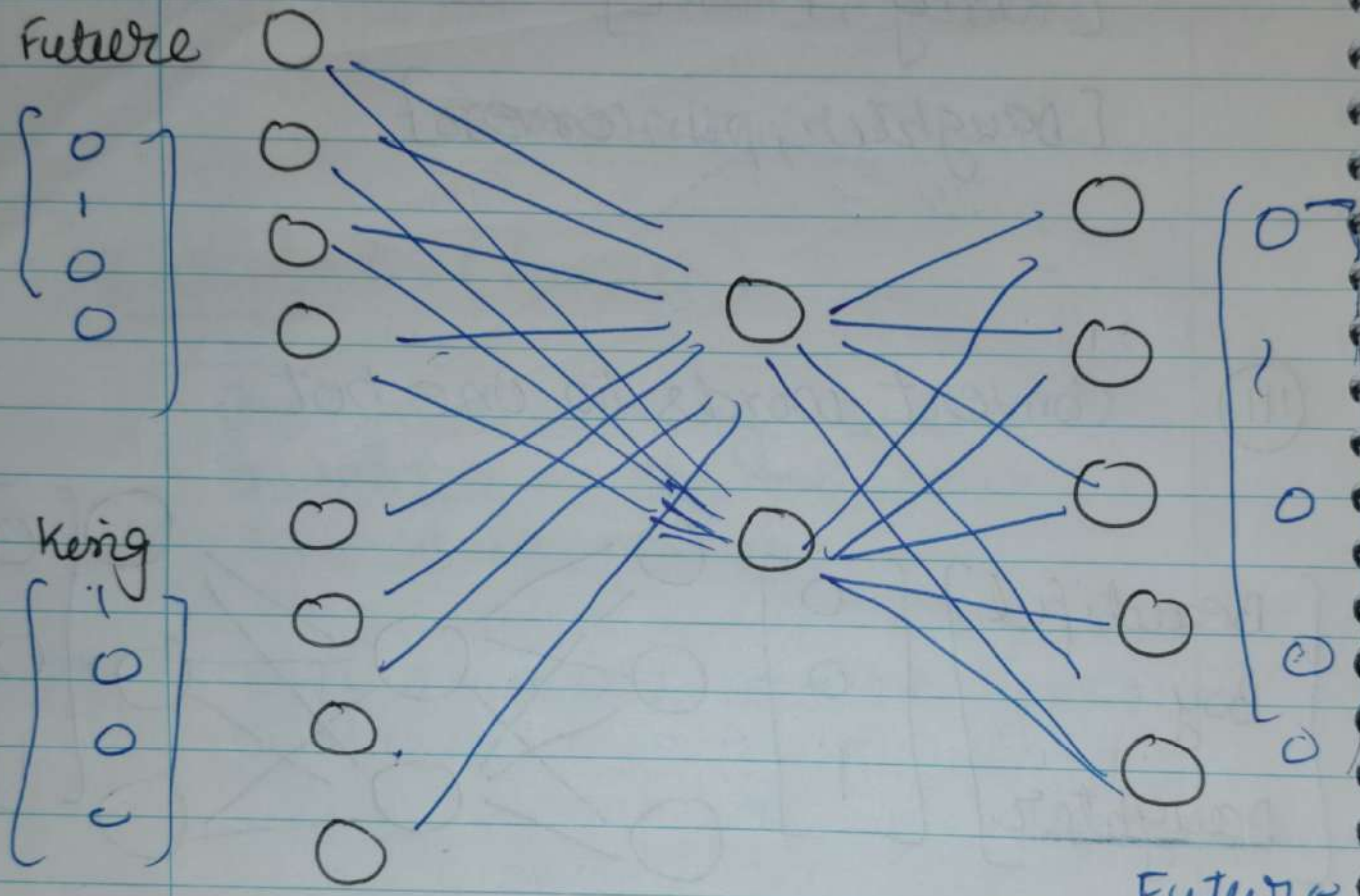
* Split for train test.

Beautiful
Boy
Princess

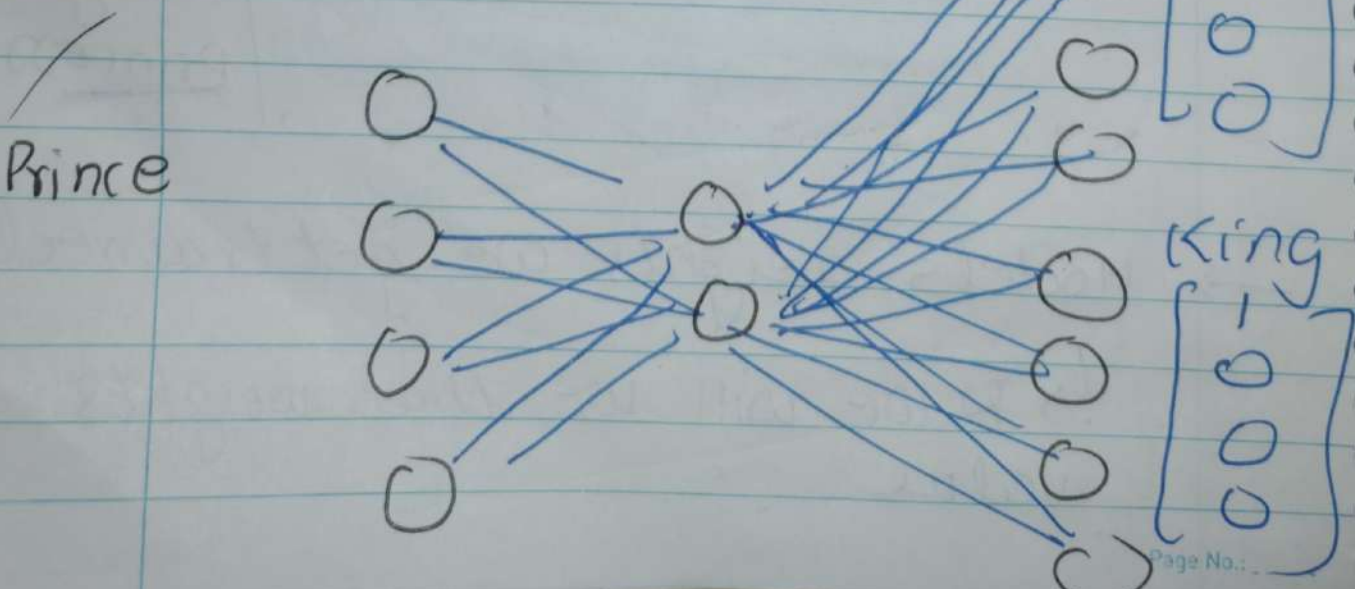
→ Models weights we get trained
↳ & we will use that weights value.

CBOW

when two or more words are used to predict single word



SKIPGRAM



SOM

→ Dimensionality Reduction method

apsara

Date: __/__/____

↳ Self organizing ann no supervision is required

Learn on their own through unsupervised competitive learning

SOM : NN that uses unsupervis-ed learning approach & trained its n/w through a competitive learning algo to map multidem ensional data into low-dimension-al (1D or 2D) which allow easy intrepation of complex problem

MultiDimension feature Map → (Map) lower Dimens-ional feature Map

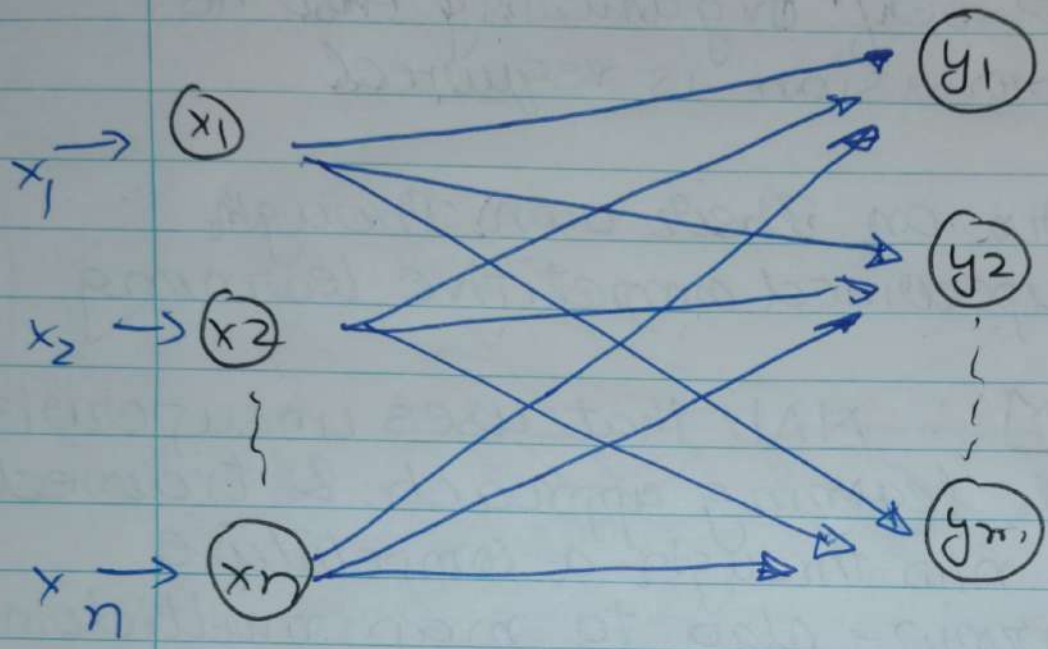
SOM layers => (i) i/p layer
(ii) o/p layer

If there n i/ps => m o/ps

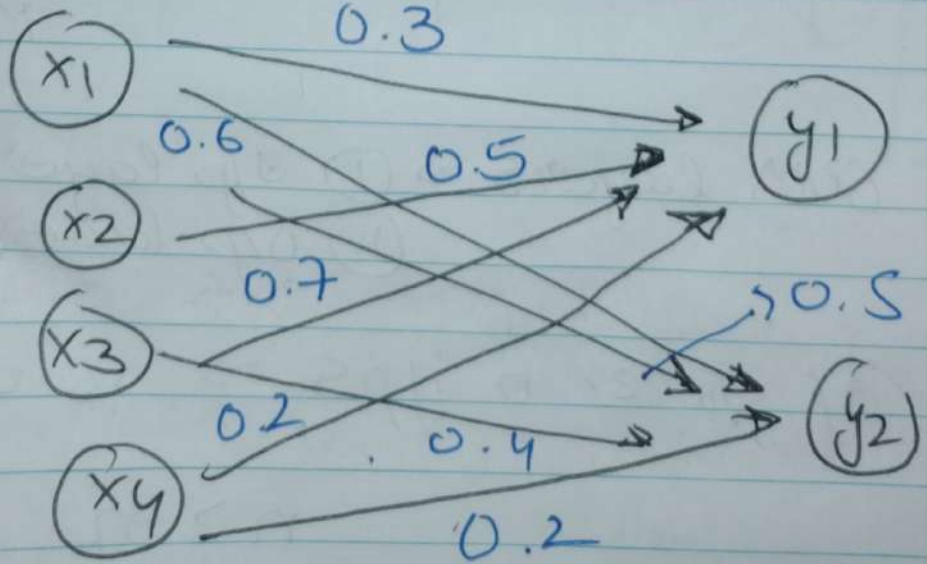
$n > m$

Page No.: _____

Architecture SOM (Fully Connected)



Q Consider the n/w shown in fig which consider four training samples each vector of length 4 & two o/p units



→ Train SOM by determining the class membership of i/p data

x_1: (1, 0, 1, 0) x_2: (1, 0, 0, 0)

x_3: (1, 1, 1, 1) x_4: (0, 1, 1, 0)

Ans

→ Steps

↳ i) Calculate Euclidean Distance one which have less won

ii) Update that particular weights which won

$$w_j(t+1) = w_j(t) + \eta(t)(x_s - w_j(t))$$

- Intial wt. Matrix

$$\begin{bmatrix} \text{unit } 1 \\ \text{unit } 2 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.5 & 0.7 & 0.2 \\ 0.6 & 0.8 & 0.4 & 0.3 \end{bmatrix}$$

1st Iteration

Training sample : $x_1 : (1, 0, 1, 0)$

→ Compute Eucledian dis b/w x_1 & unit 1 weights

$$d^2 = (0.3-1)^2 + (0-0.5)^2 + (1-0.7)^2 + (0-0.2)^2$$

$$= 0.87$$

→ Ily for x_1 & unit 2 wts.

$$d^2 = (0.6-1)^2 + (0.7-0)^2 + (0.4-1)^2 + (0.3)^2$$

$$d^2 = 1.1$$

★ Unit 1 wins.

→ Update Unit 1 weights

$$w(t+1) = w(t) + \eta(t)(x_s - w(t))$$

$$w(t+1) = [0.3\ 0.5\ 0.7\ 0.2]$$

$$+ 0.6[(1,0,1,0) - (0.3\ 0.5\ 0.7\ 0.2)]$$

$$w(t+1) = [0.72\ 0.2\ 0.88\ 0.08]$$

$$\begin{bmatrix} \text{unit 1} \\ \text{unit 2} \end{bmatrix} = \begin{bmatrix} 0.72 & 0.2 & 0.88 & 0.08 \\ 0.6 & 0.7 & 0.4 & 0.3 \end{bmatrix}$$

Iteration 2

$$x_2 = [1, 0, 0, 0]$$

→ From Unit 1

$$d^2 = (1-0.72)^2 + (0-0.2)^2 + (0.88)^2 + (0.08)^2$$

$$d^2 = 0.0784 + 0.04 + 0.774 + 0.0064$$

$$= 0.8992$$

Ily x_2: from Unit 2

$$d^2 = (0.6-1)^2 + 0.7^2 + 0.4^2 + 0.3^2$$

$$d^2 = 0.9$$

Unit 1 wins

→ Update wt. of Unit 1

$$w_{t+1} = w_t - \eta(x - w_t)$$

$$w_{t+1} = [0.72\ 0.2\ 0.88\ 0.08] - 0.6[[1,0,0,0] - [0.72\ 0.2\ 0.88\ 0.08]]$$

$$w_{t+1} = [0.72\ 0.2\ 0.88\ 0.08] - 0.6[0.28\ -0.2\ -0.88\ -0.8]$$

$$\omega_{t+1} = [0.72 \quad 0.2 \quad 0.88 \quad 0.8]$$

$$t\,[-0.168 \quad 0.12 \quad 0.528 \quad 0.48]$$

$$\omega_{t+1} = [-0.96 \quad 0.32 \quad 1.408 \quad 1.28]$$

Ully Calculate 3
Uly 4 iteration

BOLTZMAN MACHINES

↳ Used to Solve Two Problems

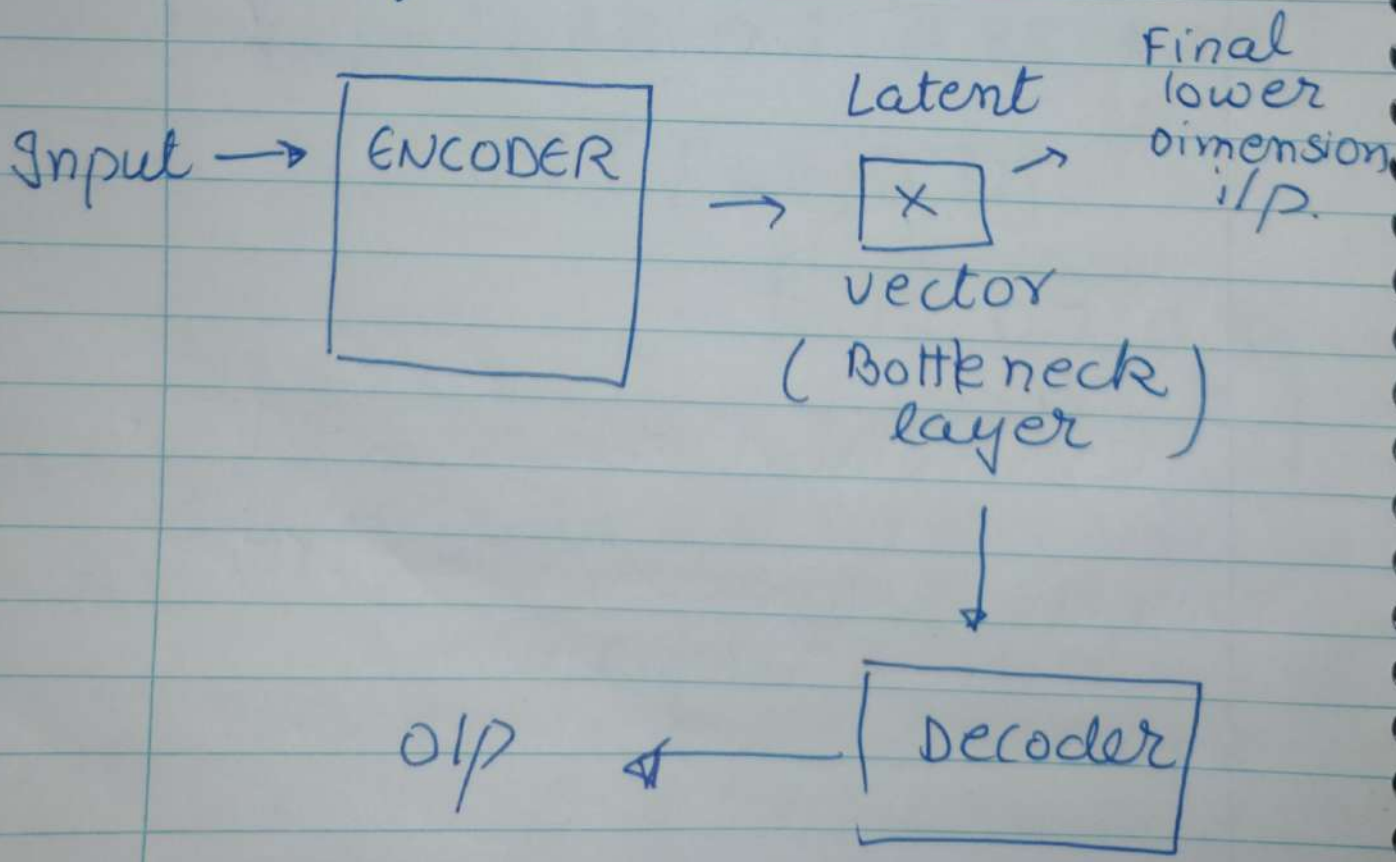
(I) Search Problem
(II) Learning Problem

- Restricted Boltzmann Machines

DON'T DETECT → THEY RECONSTRUCT

← AUTO ENCODER

↳ is an unsupervised learning technique



- Uses of AutoEncoder : Anomly Detection

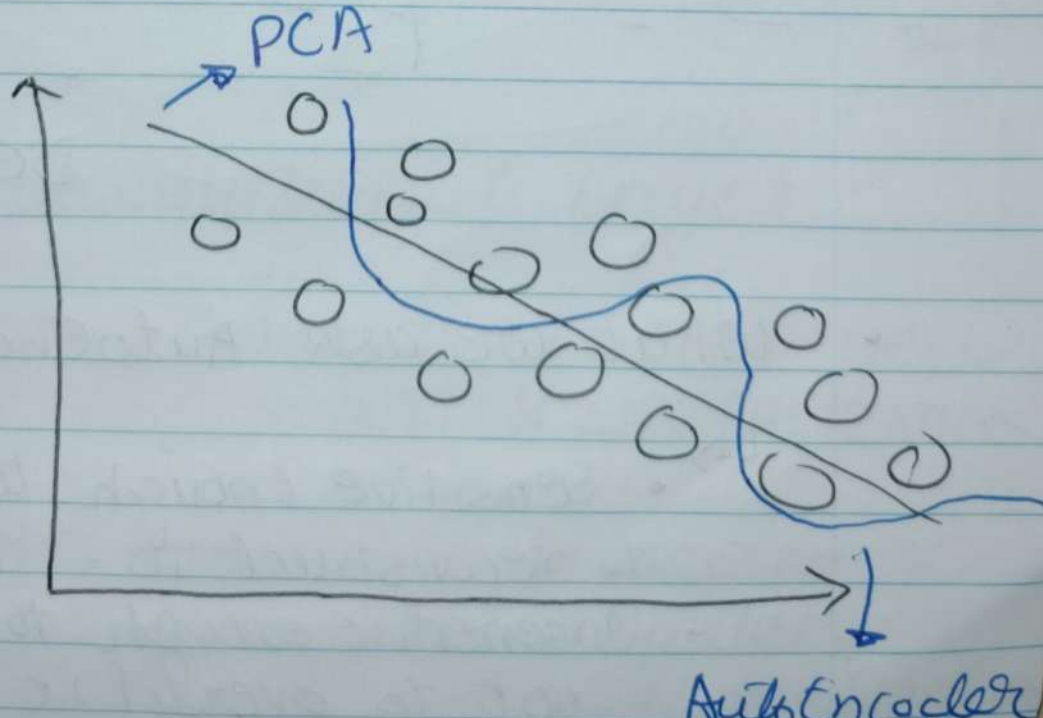
- Training Objective : To minimize Reconstruction error : error b/w o/p & i/p.

Q Difference in PCA & AutoEncoder?

→ Encoder can learn non-linear relationships

→ PCA learns linear relationships

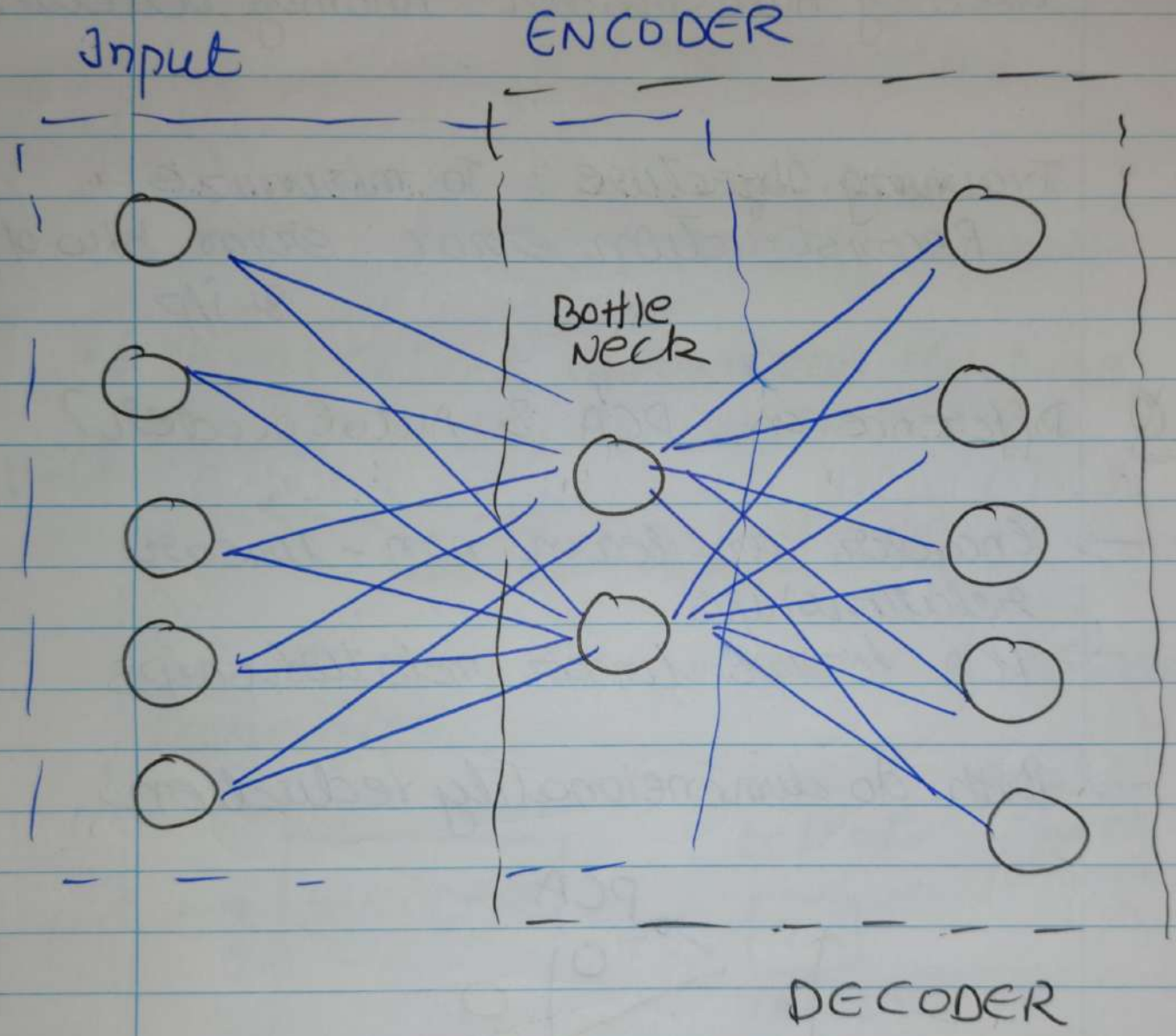
→ Both do dimensionality reduction



apsara

Date: __/__/____

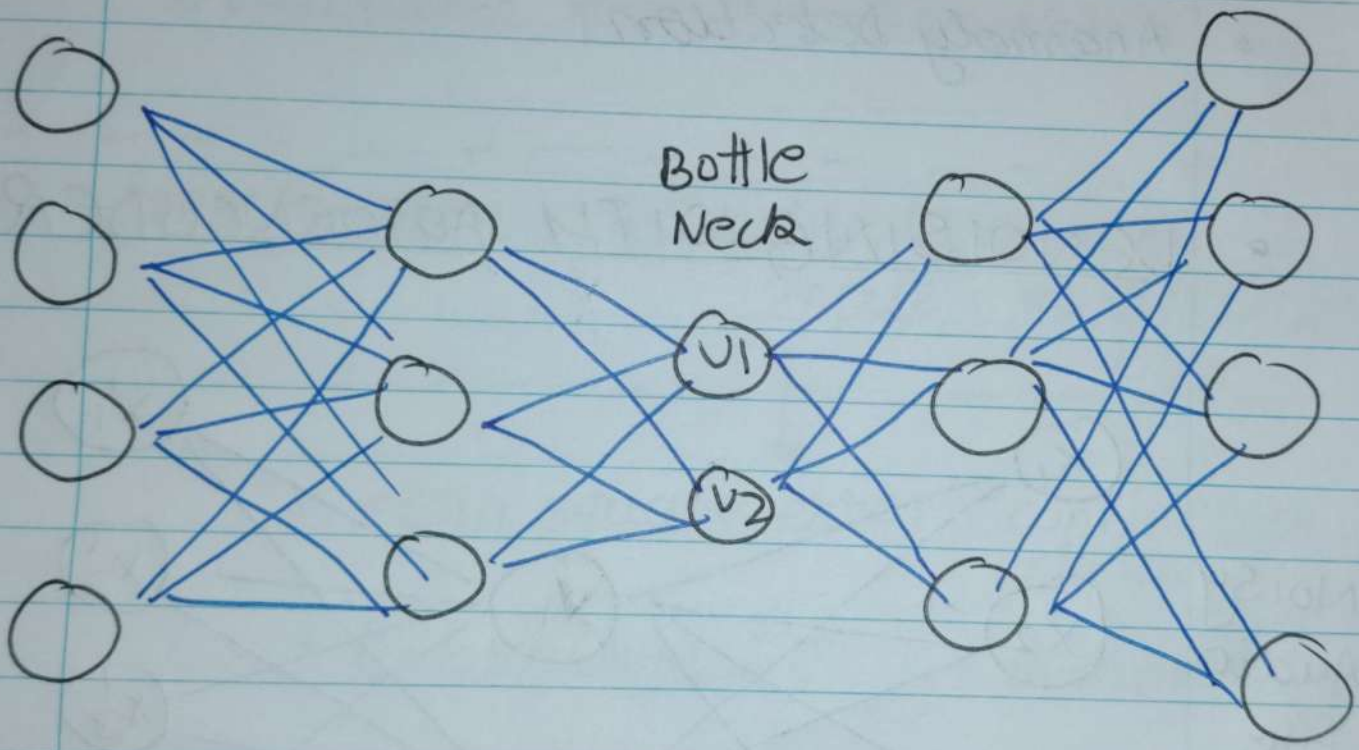
ARCHITECTURE



- What we ask AutoEncoder
  - ↳ • Sensitive enough to i/p data to reconstruct it.
    - Insensitive enough to i/p data not to overfit it

→ $E(x, \hat{x})$ + Regularization

→ HOW AUTOENCODER ACTUALLY LOOKS



Deep Convolutional Layer

- ENCODER : CONVOLUTION + LEAKY RELU + BATCH NORMALIZATION
- DECODER : Convolution transpose + Leaky Relu + Batch Normalization

Date: __/__/____

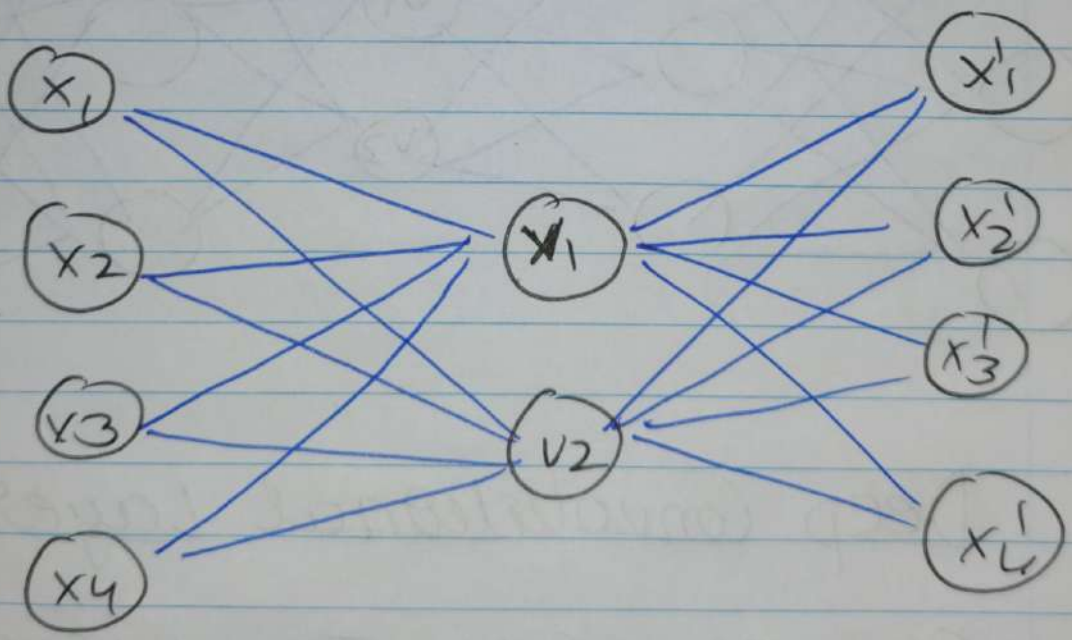
AutoEncoder Applications

Generation

Denoising => Images, Data

Anamoly Detection

DENOISING WITH AUTOENCODER



You create pairs noisy & clean data

Noisy given as i/p to the autoen-coder

- clean version is used to minimize error (reconstruction)
- Model's loss fxn compares reconstructed audio to the clean audio & calculate reconstruction error.

- loss function : reconstruction loss : input - reconstructed o/p.
  - mean square error : continuous data
  - Binary cross entropy : Binary data

Q★ AutoEncoder is not a Generative Model but Variational AutoEncoder is a generative model?

→ AutoEncoder lacks the ability to generate new, diverse sample.

apsara

Date: __/__/____

→ Variational AutoEncoder can generate new sample.

AutoEncoder	Variational AutoEncoder
i) Latent Representation → Deterministic	i) Latent Repres-entation => Probabilistic
ii) Not Capable of New Data Generation	ii) ~~Not~~ Capable of New Data Generation
iii) Function: Reconstruction	iii) Function: Generation + Reconstruction
iv) Loss fxn: mse or bce	iv) Loss fxn: mse/ bce + KL Divergence

Tips for AutoEncoder

↳ i) Keep no. of neurons less in Hidden layer

Page No.: _____

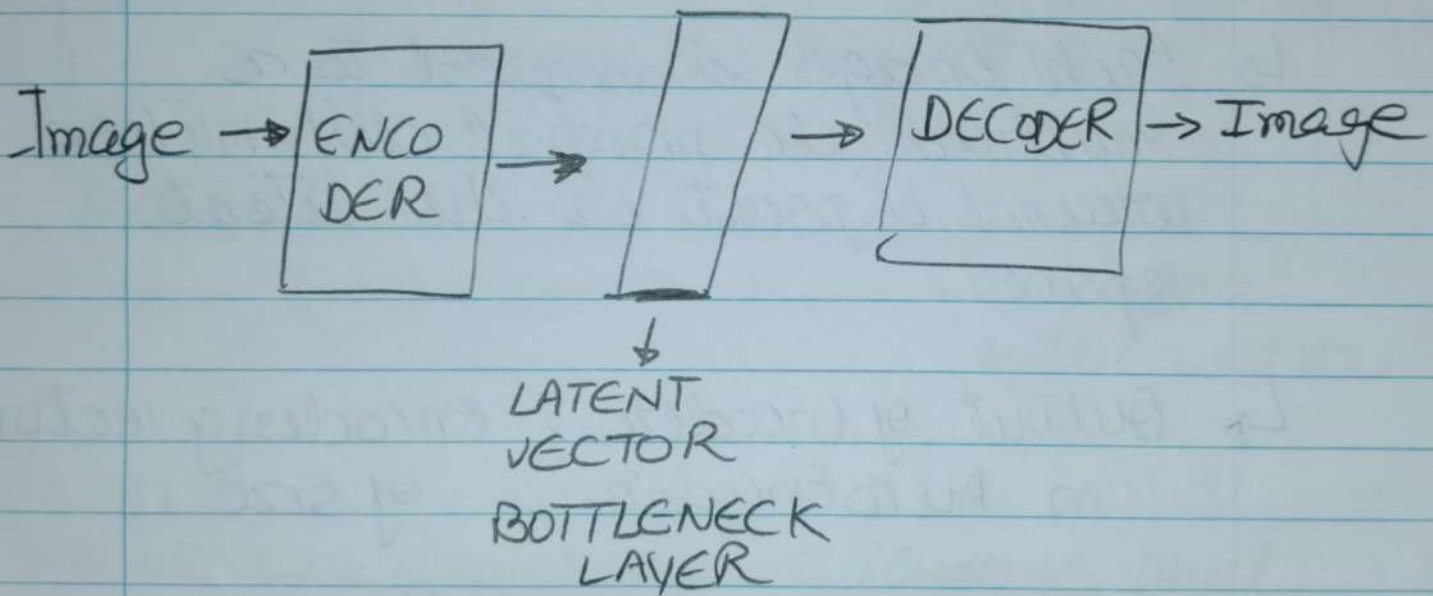
Variational ENCODER

Rapsara

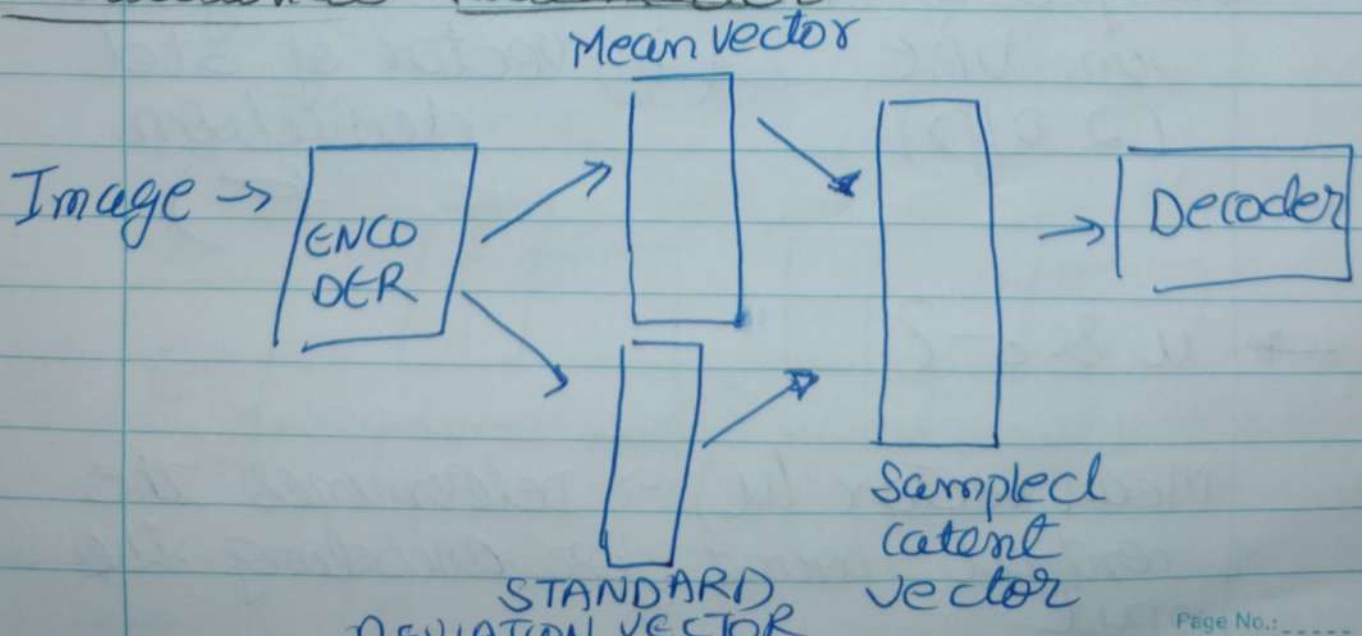
Date: ___.___.____

★ If you are working with images ⇒ You use convolutional layer.

AUTOENCODER



Variational AutoEncoder



Problem in AutoEncoder

↳ Each instance is mapped directly to one point in latent space

Variational AutoEncoder ⇒ Solver

↳ Each image is mapped to a multivariate normal distribution around a point in the latent space.

↳ Output of Encoder : Encoding vector of size n

↳ Output of Encoder in VAE (2 o/p)
(i) Vector of mean (μ)
(ii) Vector of Std deviation (σ)

→ μ & σ ?

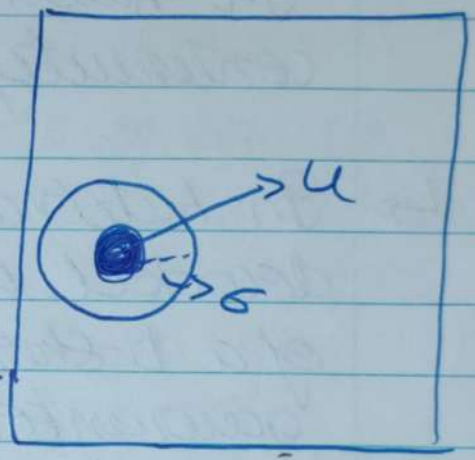
Mean vector (μ) -> Determines the central point for encoding the input.

apsara

Date: __ . __ . ____

σ : How much the encoding can deviate from central point

AutoEncoder

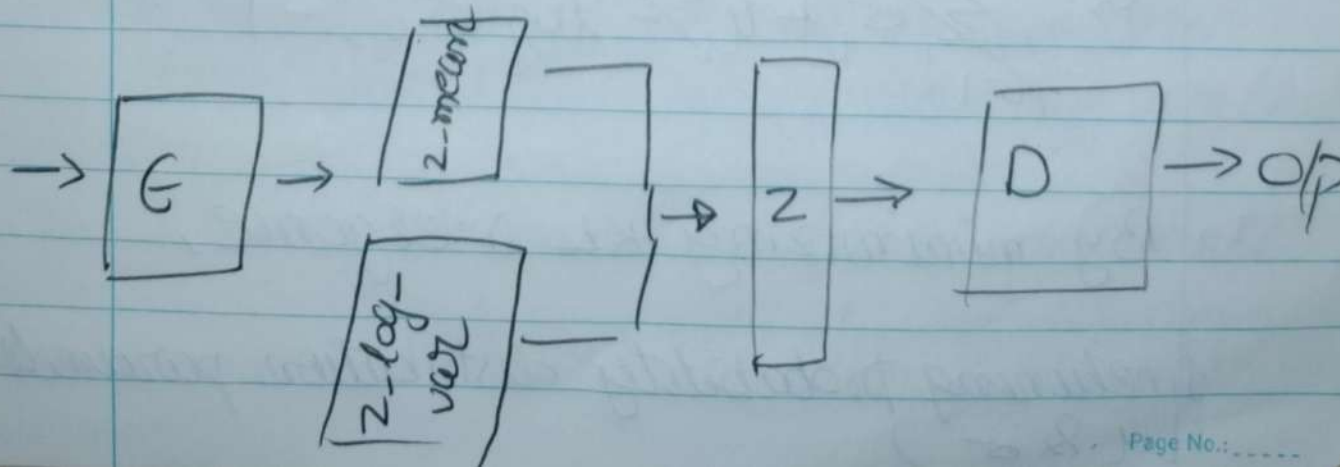


Variational AutoEncoder

→ we can generate z (sample point)

↳ z = z_mean + z_sigma * epsilon

z_sigma = $e^{(z\_log\_var * 0.5)}$



Page No.: _____

Q How this changes in Encoder in VAE helps?

↳ In Autoencoder, latent space lacks continuity

↳ In Autoencoder, if the point (-5, 3) decoded into a recognizable image of a t-shirt, then there was not guarantee that (-5.1, 3.1) would produce something similar

→ But in VAE decoder ensure nearby yield similar decoded images

K-L Divergence

↳ $\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$

↳ By minimizing KL Divergence, finetuning probability distribution parameter (μ & σ)

Uses Variational AutoEncoder

↳ i) Generating Purely Synthetic music
ii) Generating fake human faces.

→ DL Gate => Applications of Encoder
↳ Imp to learn concept

REPARAMETERIZATION TRICK

↳ In VAE, if we try to take sample from a probability distribution to feed it to decoder it is a random process
& the process of generating sample from distribution is non-differentiable
↳ So, we can't compute gradients
↳ so, can't do BackProp

Q Now, Why this process of random picking sample is non-differentiable?

↳ ∵ it don't involve depend on neural n'w's learned weights.

Q ONE MORE QUESTION => HOW TO USE VAE FOR NEW IMAGE GENERATION?

↳ So, for Generation of new images from VAE, only involve decoder part no role of encoder (X)

★ Reparameterization trick is used to make sampling process differentiable.

Q How Reparameterization trick make sampling process differentiable

By Expressing Random sample as a deterministic fxn of model's parameter and fixed noise term

$z = \mu + \sigma(\epsilon)$ → sample point from standard normal distribution

↳ from our distribution

μ, σ => o/ps of the encoder

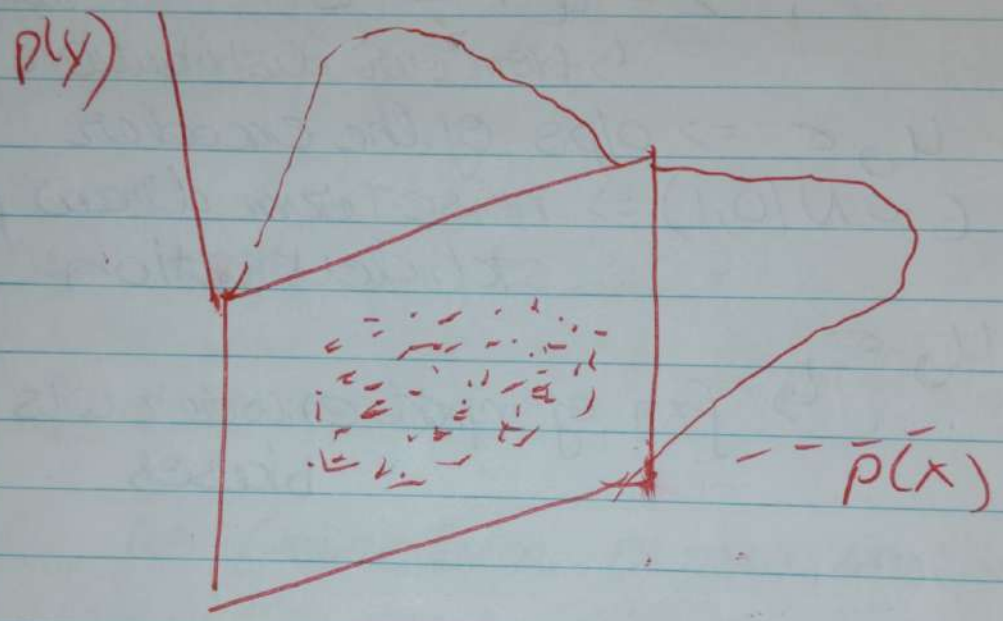
$\epsilon \sim N(0,1)$ => noise term drawn from std. destribution

μ, σ → fxn of input encoder wts. & biases

HOW TO GO FROM VANILLA AUTO ENCODER TO VARIATIONAL AUTO ENCODER

↳ • Modify Encoder Component
↳ • Modify loss fxn

Visualise Multivariate Normal Distribution



$$z = u + \sigma \cdot \epsilon$$

$$\sigma = e^{\frac{\log(\sigma^2)}{2}}$$

- Modify Loss Function

$$\text{LOSS} = \text{RMSE} + \text{KL}$$

↓

Root mse

KL ⇒ difference b/w normal distribution (mean, log variance vector) from standard normal distribution

$$D_{KL} = \frac{1}{2} \sum (1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

★ What does this KL loss term do?

→ Penalizes observation where mean & log variance vectors differe significantly from parameters of standard Normal Distribution

★ KL Divergence Loss: Promotes Symmetry Term around origin

- Decrease chances of large gaps b/w clusters of points

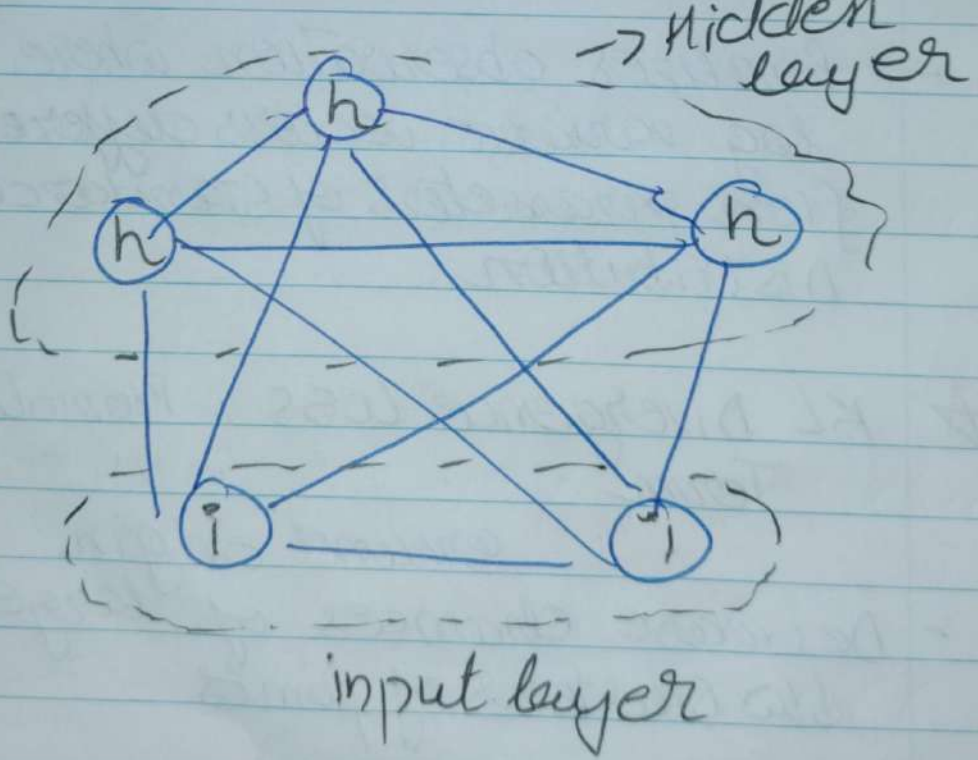
BOLTZMANN MACHINE

Date: __/__/____

↳ Unsupervised

↳ N/w of symmetrically connected neuron like units that make stochastic decision about whether to be on or off

- Fully connected
- No data flow direction



- No o/p layer
- input units are also interconnected

apsara

Date: __.__.____

- Work by probabilistic updating their states based on energy function
- Used for Dimensionality Reduction & feature learning
- NW learns to represent complex prob. Distribution
- During training, optimize weight by minimizing loss function ⇒ Energy of the system

RESTRICTED BOLTZMANN

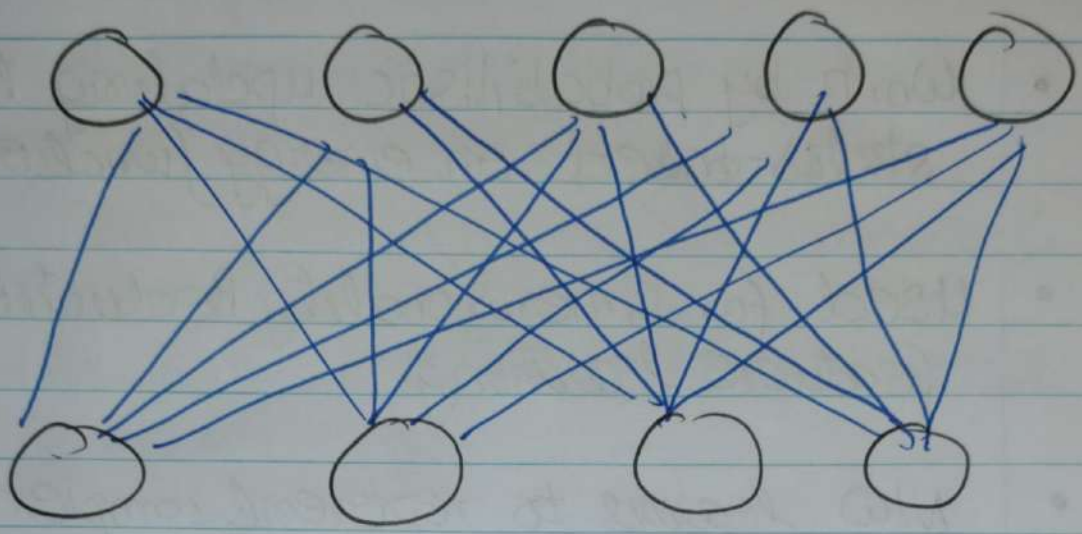
↳ i/p - i/p & i/p - hidden connect -ion was removed. ↳ hidden - hidden

Q Why shifted to Restricted Boltzmann?

i) ↳ Reduced complexity of model ⇒ R.B

ii) Slow convergence & high computional overhead ⇒ B

Hidden

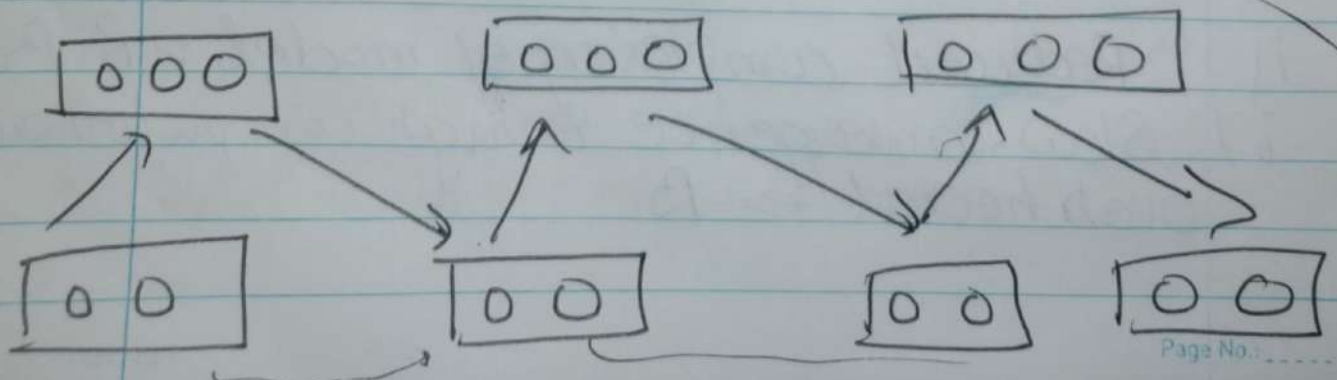


i/p (visible) node

RB → Generative Stochastic ANN that can learn probability distribution over its set of inputs

★ Can be used in Movie Recommendation System

GIBBS SAMPLING & CONTRASTIVE DIVERGENCE



Wow Just Amazing

2014 (Introduced)

apsara

Date: __/__/____

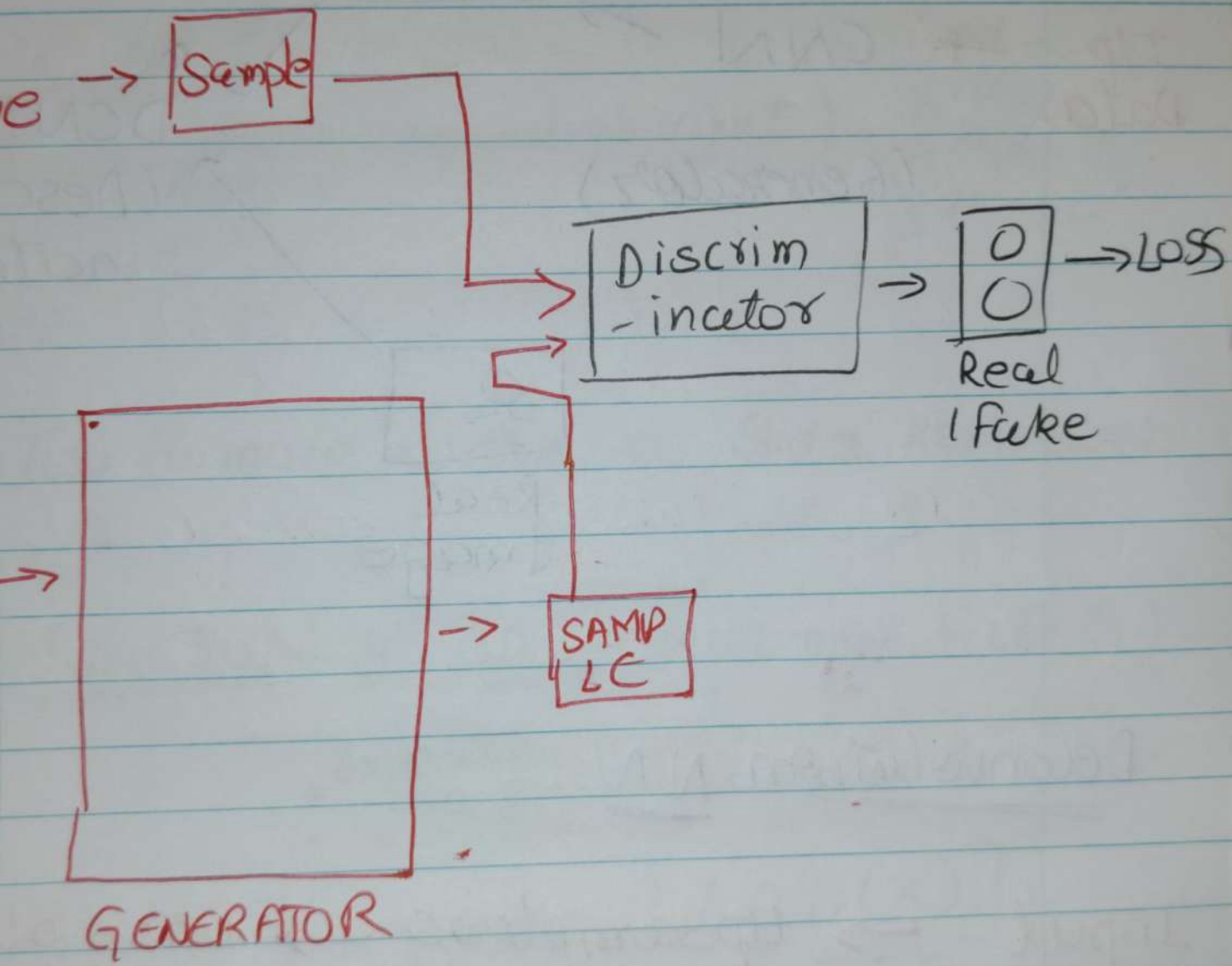
GAN

↳ Generative Adversial N/w

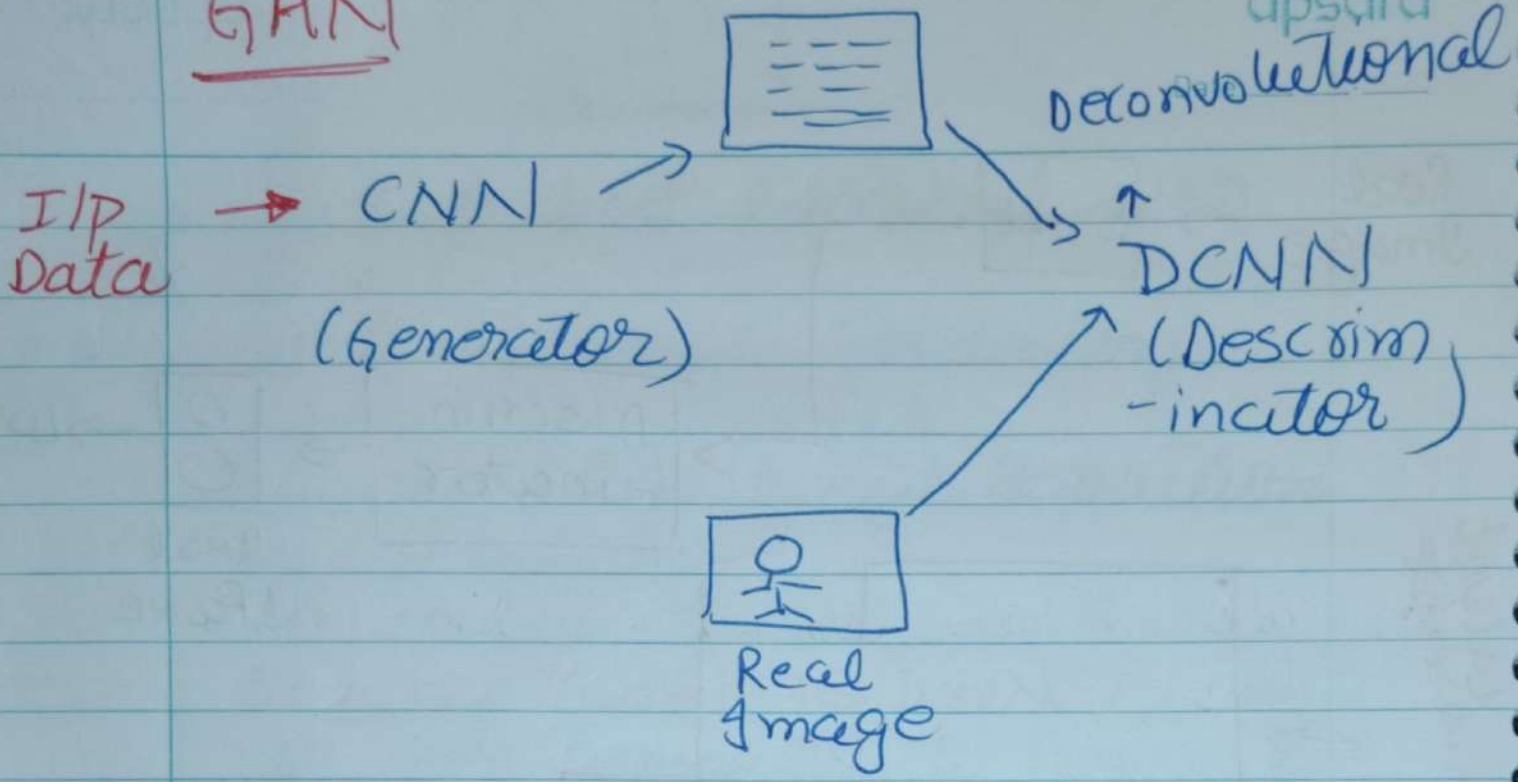
- Text to Image Generation
- Water Mark Remover
- Low Resolution to High Resolution by Lip-Syncing

★ AFTER MODEL IS TRAINED, we use only Generator to Generate Images

- Generator : Convo Layer + Batch Norm + ReLU.
- DISCRIMINATOR : Takes Two i/p
  i) Generator o/p.
  ii) Real image



GAN



Deconvolution NN

Input → Upsampling → Deconvolutio-nal layer

(Flatten Layer)

(Opposite of Down Sampling Max Pooling)

(Feature Generation Image Reconstru-ction)

↓

Activation Function

Batch Normalization (optional)

← O/P

apsara

Date: __/__/____

→ 0/1 (for many images we put as input)

1st => 0
2nd => 0
3rd => 0
4th => 1
5th => 0
:

- Two formula written in Slides Remember it.

LOSS FUNCTION OF GAN: $\min_G \max_D V(D,G)$

$$V(D,G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

↑ Expectation of Discriminator when we give real images

↳ Expectation of Discriminator when we give fake images

G = Generator
D = Discriminator
Pdata(x) = distribution of real data (→ Probability)
P(z) = Distribution of generator
D(x) = Discriminator N/w

$G(z)$ ⇒ Generator N/w

It can also be written as

$$L_D = -[\log(D(x)) + \log(1 - D(G(z)))]$$

o/p of Discri-minator
- $D(x)$ ⇒ o/p of real data by Discriminator
- $G(z)$ ⇒ fake data generated by generator
- $D(G(z))$ ⇒ Descriminator o/p for fake data

Real Data → Discriminator → $D(x)$

Fake Data → ,, → $D(G(z))$

o/p ⇒ 0 or 1

↳ we want to train our model in such a way that it should

(In Discriminator)
- $D(G(z))$ ⇒ o/p for fake data ⇒ 0
- $D(x)$ ⇒ o/p for Real data ⇒ 1

→ It will train the model perfectly by making contest

DC-GAN → Deep Convolutional GAN

↝ It uses ~~convolutional~~ DC

↳ ~~Simple~~ GAN: Use convolutional layer (deconvolution layer) to generate images

↳ Simple GAN: Use fully connected Dense Layer

GAN	DCNN
(i) Generic Arch. with fully connected layers	(i) Arch. specially designed for image generation
(ii) used for generating various type of Data	(ii) Generating Images

(iii) Can take any form of i/p data including vector, matric	(vi) Image Data
(v) Fully Connect -ed layer	(iv) Stabilize Traini -ng by Batch Normalization, avoiding fully connected layer

Other Types of GAN

ii) Conditional GAN.

iii) Super Resolution GAN: Low Resolution → High Resolution

↳ DNN + Adversial NN

iv) Info GAN: