Neural Network:

A system composed of many simple processing elements operating in parallel network whose function is determined by ~~neural~~ structure connection strength and the processing performed at computing element.

Artificial Neuron:

Info. processing system that has certain characteristics in common with biological Neurons.

Steps

1) The processing element receives many signals.
2) Signals can be modified by weights at receiving synapse.
3) The processing element sum the various weighted inputs.
4) Under appropriate condition, the neuron transmit the op signal.
5) Output from one particular neuron may go in another neuron.

JUNE '21

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

JULY '21

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

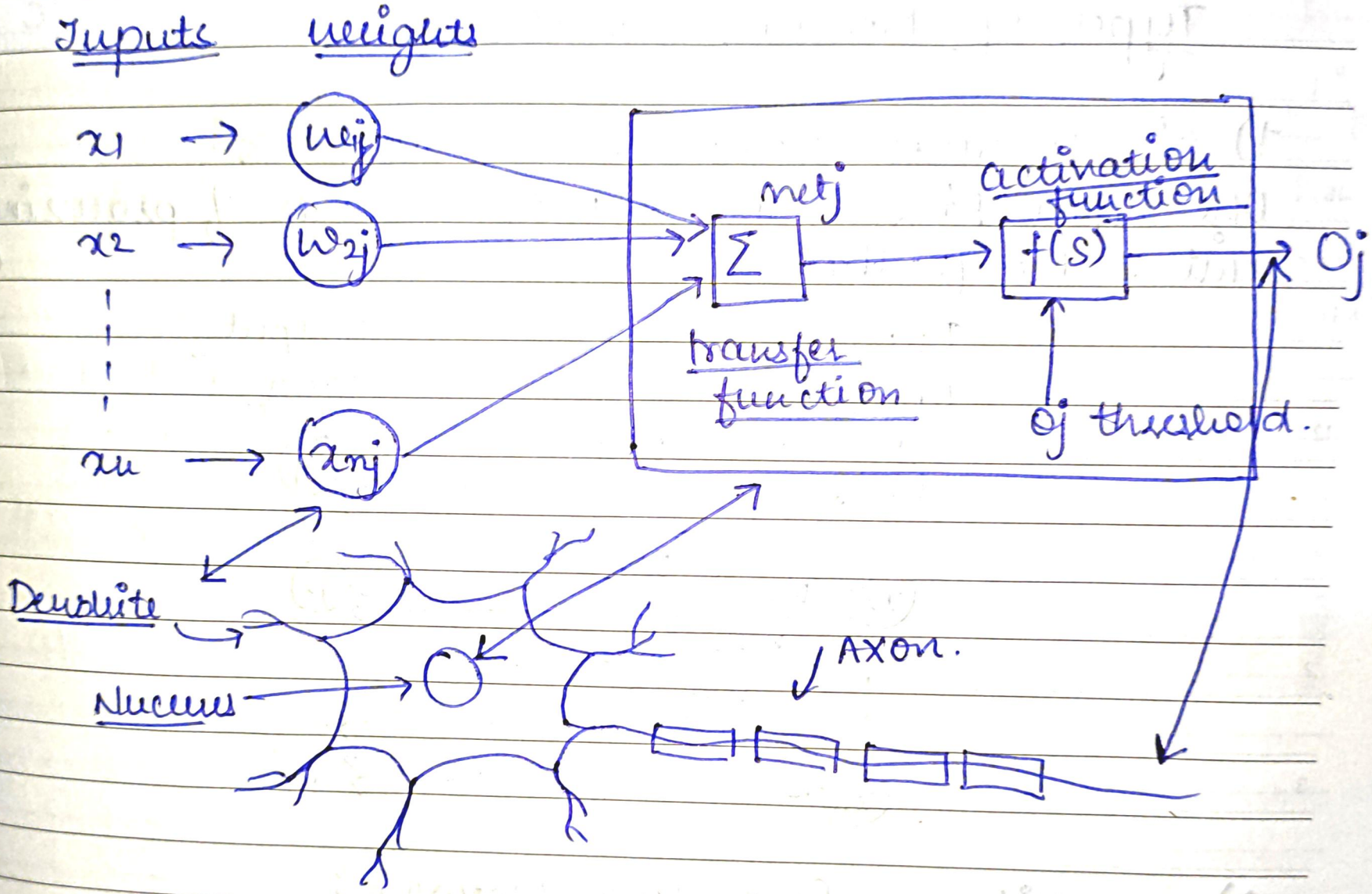
Artificial Neural Network (ANN)

Wk 20 | DAY 135-230

SATURDAY

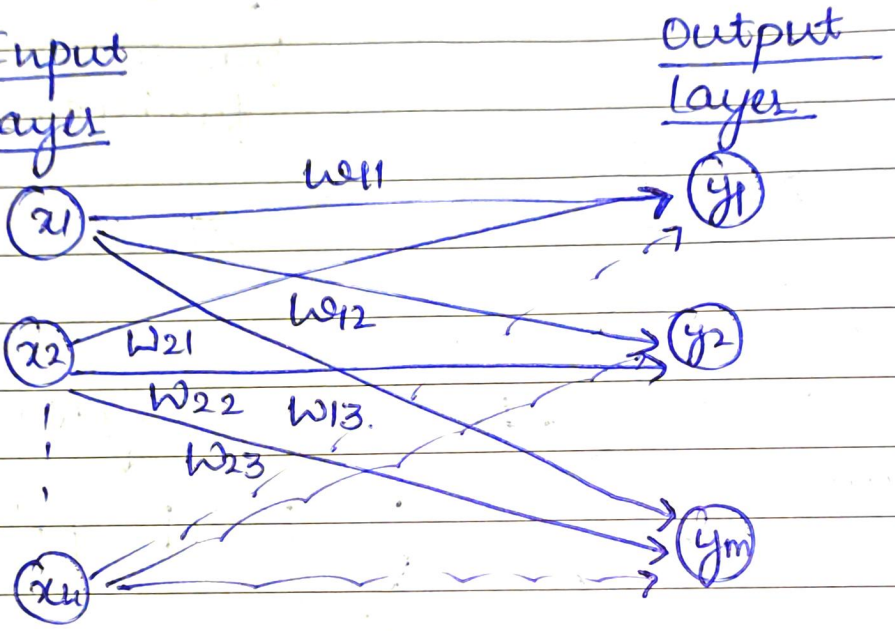
MAY '21

15

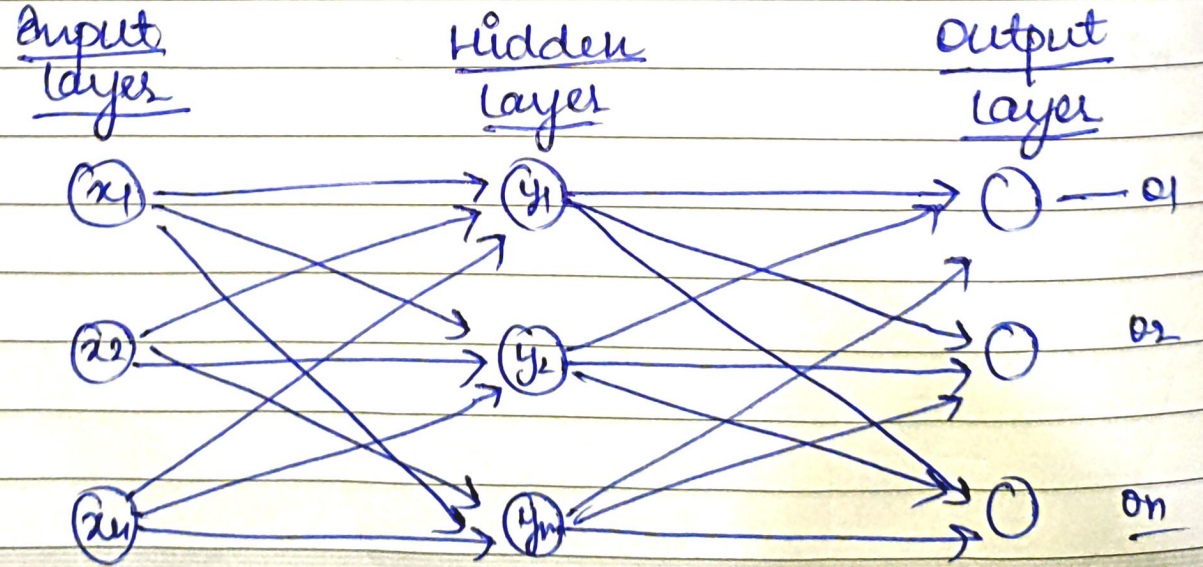


Types of Neuron connections

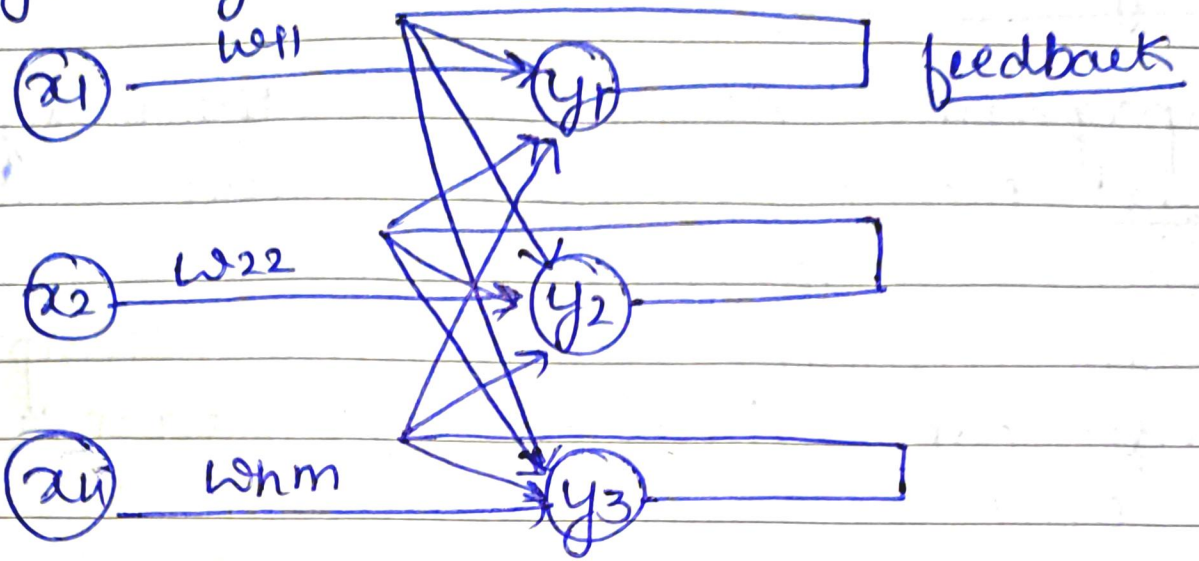
1) Single layer feed-forward Network:
One input layer, one output layer of processing unit, No feedback.



2) Multilayer Feed fwd Network:
One Ip layer, One op layer and one or more hidden layer of processing unit.



3) Single layer Recurrent Network :



4) Multilayer Recurrent Network :

Backpropogation Algorithm

↳ Backpropagation Algo. learns the weights for Multilayer Neural Network given with fixed set of units and interconnections.

↳ It employs gradient ducent to attempt to minimi the square error between the network output values and the target values of those output.

In backpropagation algorithm, we consider network with multiple outputs

The sum of errors of all the network o/p.

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

outputs → set of output units in ntwork

t_{kd} & o_{kd} → target output & network ouput associated with kth output unit.

$d \in D$ → training example.

Notations :

1) each training example is a pair of $(\vec{x}, \vec{t})$ (↓ Input values, ↓ target values)

2) η = learning rate

3) n_i = no. of Ip units n_h = no. of hidden units

n_o = no. of Op units

4) x_{ij} = ip from i to j

5) w_{ij} = wt from i to j

1) Create a feed forward Network with n_i, n_o and n_h

2) Initialize all ntwk wt to some small random number.

3) until termination condition is met, Do.
for each $(\vec{x}, \vec{t})$ in training example.
⇒ propagate network input fwd in network.

(i) Input $\vec{x}$, to network and compute O_k for every unit u in network.

⇒ propagate error backward through network

(ii) for each network output unit k, calculate error

$$\delta_k \leftarrow O_k(1-O_k)(t_k - O_k)$$

(iii) for each hidden unit h, calculate error

$$\delta_h \leftarrow O_h(1-O_h) \sum_{k \in \text{outputs}} w_{hk}\delta_k$$

(iv) update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

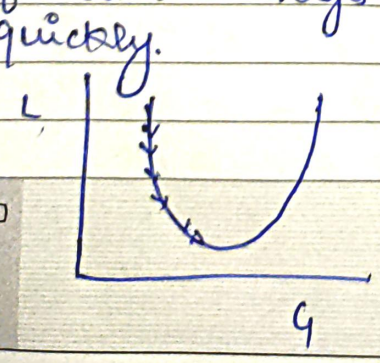
$$\Delta w_{ji} = \eta \delta_j x_{ij}$$

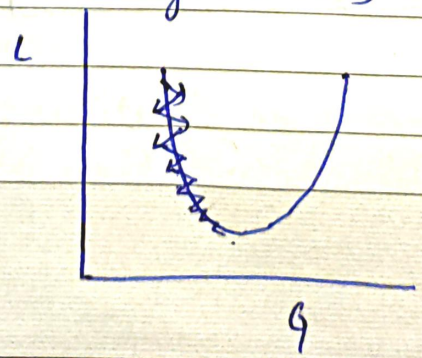
Gradient Descent

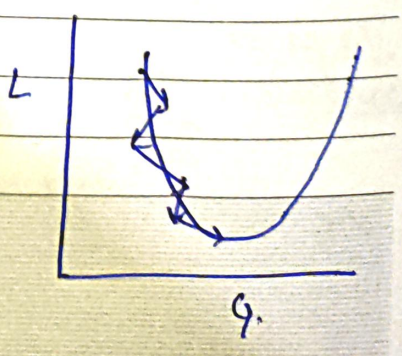
Gradient descent is an optimized algorithm used to find the value of parameters of a function that minimizes the cost function.

Depending upon the no. of training examples

Batch Gradient Descent	Stochastic Gradient Descent	Mini Batch Gradient Descent
Parameters are updated after computing the gradient errors wrt to entire training set.	Parameters are updated after computing the gradient of errors wrt to single Training example	Parameters are updated after computing the gradient of errors wrt to subset of training set.
$E = \sum_{i=0}^{n} (t_i - o_i)^2$	$E = (t - o)^2$	$E = \sum_{i=0}^{k} (t_i - o_i)^2$
smooth & loss function converges quickly.	slow (large dataset)	







Stocastic Gradient Descent

1) Take an exple
2) Feed it to NN
3) calculate its gradient
4) use gradient in step 3 to update the weights.
5) Repeat 1-4.

Mini Batch GD

1) Pick a mini batch
2) Feed it to NN
3) Calculate mean gradient
4) use the mean gradient to update weights
5) Repeat 1-4.

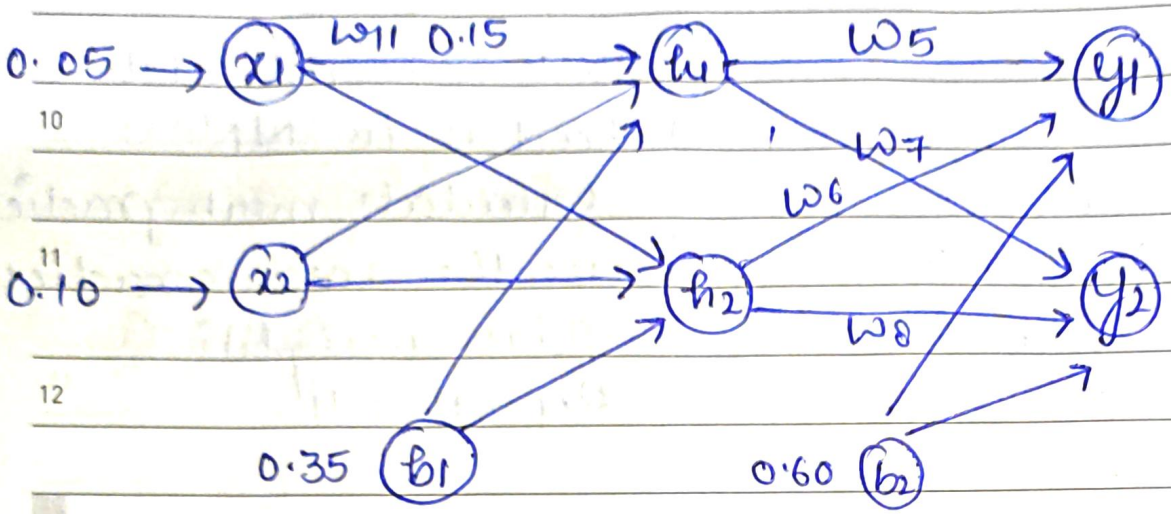
Difference between Batch & Stocastic

BGD.	SGD
1) Compute Gradient of Whole training set.	1) compute Gradient using Singee training example.
2) slow converge	2) faster converge.
3) Computationally exp.	3) Computationally less exp.
4) for small dataset.	4) for large dataset
5) Deterministic in nature	5) Stocastic in nature.
6) Optimal sol.	6) Not optimal.
7) Can't escape shallow local minima.	7) can easily escape shallow local minima.
8) No. Random shuffling of pts are req.	8) Shuffle dataset. (Random order preferred).

SUNDAY 23

(Noisy)

Backpropagation Numerical



Target value
$t_1 = 0.01$
$t_2 = 0.99$

$x_1 = 0.05$ $\quad w_{11} = 0.15$ $\quad w_{21} = 0.20$

$x_2 = 0.10$ $\quad w_{12} = 0.25$ $\quad w_{22} = 0.30$

Forward Pass

$$h_1 = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$= 0.35 + 0.05 \times 0.10 + 0.20 \times 0.10$$
$$h_1 = \underline{0.3775}$$

$$h_{1f} = \frac{1}{1+e^{-h_1}} = \frac{1}{1+\frac{1}{e^{0.3775}}} = \underline{0.59327}$$

$$h_2 = b_1 + x_1 w_{12} + x_2 w_{22}$$
$$h_2 = 0.35 + 0.05 \times 0.25 + 0.10 \times 0.30$$
$$h_2 = \underline{0.3925}$$

$$h_{2f} = \frac{1}{1+\frac{1}{e^{0.3925}}} = \underline{0.596884}$$

$y_1 = b_2 + h_1 w_5 + h_2 \times w_6$

$y_1 = 1.1059$

$y_{1f} = \frac{1}{1+\frac{1}{e^{y_1}}} = \underline{0.751365}$

$y_2 = b_2 + h_1 w_7 + h_2 w_8$

$y_2 = 1.22492$

$y_{2f} = \frac{1}{1+\frac{1}{e^{y_2}}} = \underline{0.77298}$

$E_{total} = \sum \frac{1}{2} (t-o)^2$

$= \frac{1}{2}(0.01-0.751365)^2 + \frac{1}{2}(0.99-0.77298)$

$\underline{E_{total} = 0.298371111}$

Backward pass

$Error_w = \frac{\partial E_{total}}{\partial w}$

eg. for w_5

$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial y_{1f}} \times \frac{\partial y_{1f}}{\partial y_1} \times \frac{\partial y_1}{\partial w_5}$

$\downarrow$

$\frac{\partial\left(\frac{1}{2}(T_1-y_{1f})^2+\frac{1}{2}(T_2-y_{2f})^2\right)}{\partial y_{1f}}$

$2 \times \frac{1}{2}(0.01-0.751365) \times -1 + 0$

$= \underline{0.74136507}$

Gradient descent Algorithm (TE, η)

each TE is a pair of $(\vec{x}, \vec{t})$

$\vec{x}$ = vector of ip values

$\vec{t}$ = vector of target values.

1) Initialize each w_i to some random value.

2) until termination condition is met, do.

(i) Initialize each $\Delta w_i = 0$

(ii) for each $(\vec{x}, \vec{t})$ in TE

↳ Input $\vec{x}$ & compute o

↳ for each linear unit weight w_i

$$\Delta w_i \leftarrow \Delta w_i + \eta (t - o) x_i$$

(iii) for each linear unit weight, do

$$w_i \leftarrow w_i + \Delta w_i$$

Loss function

Loss function estimate how well particular algorithm models the provided data.

Regression loss

1) Mean square error
Avg. of squared diff b/w predicted & actual obs.

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

2) Mean absolute Error.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

3) Mean bias error.

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

4) Huber loss (smooth MAE)

$$loss = \begin{cases} \frac{1}{2}(x-y)^2 ; & |x-y| \le \delta \\ \delta|x-y| - \frac{1}{2}\delta^2 & \text{else} \end{cases}$$

5) Mean squared logarithmic error

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

Classification loss

1) Cross entropy loss.

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

2) Hinge loss

$$loss = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\sum \max(0, s_j - s_{y_i} + 1)$$

3) Kullback Leibler Divergence loss

$$D_{KL}(P||Q)$$

(discrete)

$$-\sum_x P(x) \log \frac{Q(x)}{P(x)} = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

(cont)

$$-\int P(x) \log \frac{Q(x)}{P(x)} dx = \int P(x) \log \frac{P(x)}{Q(x)} dx$$

$\frac{P(x)}{Q(x)}$ distribution

21' MAY

28 DAY 148-217 | Wk 22 FRIDAY

human brain filters info

Deep learning The subfield of artificial intelligence that focuses on creating large nn model that capable of making accurate data driven decisions.

Machine Learning	Deep learning
1) works on small amount of dataset for accuracy.	1) works on large amount of dataset.
2) Dependent on low-end Machine.	2) Heavily dependent on high-end machine
3) Divide the task into sub-task, solves them indivi- and finally combines them	3) Solves problem end to end.
4) Less time to train	4) More time to train
5) More time to test	5) Less time to test.

Applications
1. Speech Recognition
2. Image Recognition
3. Self Driving Cars & Robots.
4. Climate Prediction
5. Auto text gen., NLP.

Regularization

A technique which makes slight modifications to the learning algo such that model generalizes better.
This improves model's performance on the unseen data

Machine learning
(Penalizes the coeff)

Deep learning
(Penalizes weight matrices of Node)

Regularization Techniques

① L1 & L2 Regularization.

cost fxn = loss + RT

L_1 = loss + $\frac{\lambda}{2m} \times \Sigma \|w\|$

L_2 = loss + $\frac{\lambda}{2m} \Sigma \|w\|^2$

λ = regularization parameter.

② Dropout : At every iteration, Randomly select some nodes & remove them along with their incoming and outgoing connections.

#Ensemble learning Technique

③ Data Augmentation
Increase the size of training Data
eg. Img → flip, rotate, scale

④ Early stopping
- Cross-Validation Strategy
- Validation set
  - ↳ performance start degrading
    - ↳ stop training

SUNDAY 30
JUN

Parameters in BPN

1) No. of hidden layer nodes:

Guiding criterion → 1st & 3rd layer nodes are min.

2) Momentum coefficient

To reduce training time

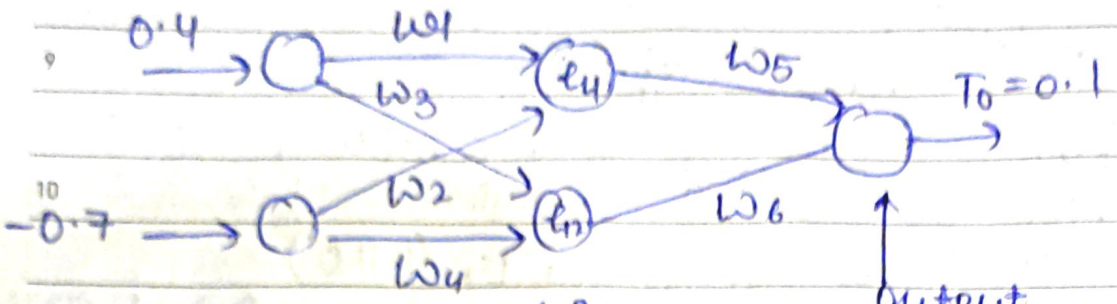
$$[\Delta W]^{n+1} = -\eta \frac{\partial E}{\partial W} + \alpha [\Delta W]^{n}$$

Overcomes local minima.

3) Sigmoidal Gain λ:

When weights become large & force neurons to operate in a region where sigmoidal fun^{n} falls flat, then adjust the λ. (decrease)

4) Local Minima: Change all weights randomly.



Step 1

$w_1 = 0.1$ $w_3 = 0.4$ $w_5 = 0.2$

$w_2 = 0.2$ $w_4 = 0.2$ $w_6 = -0.5$

Step 2 $h_1 = b_1 * 1 + w_1 i_1 + i_2 w_2$

$= 0 + 0.4 \times 0.1 + (0.2) \times (0.7)$

$\boxed{h_1 = 0.18}$

$h_2 = 0.4 \times 0.4 + -0.7 \times 0.5$

$\boxed{h_2 = 0.02}$

Step 3 $O_{H_1} = \frac{1}{1+e^{-h_1}} = 0.5448$

$O_{H_2} = \frac{1}{1+e^{-h_2}} = 0.5050$

Step 4 $\text{Net}\, O_1 = \Sigma w_i x_i$

$= 0.5448 \times 0.2 + 0.5050 \times (-0.5)$

$= -0.14354$

Step 5 $O_{o1} = \frac{1}{1+e^{-\text{net}\, o_1}} = 0.4642$

Step 6 $E_{total} = \frac{1}{2}(t_{o1} - o_{o1})^2$

$= \frac{1}{2}(0.1 - 0.4642)^2$

$= 0.0663$

Step 7:

$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial o_{o1}} \times \frac{\partial o_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$

$\frac{\partial E}{\partial o_{o1}} = -(t_{o1} - o_{o1}) = -(0.1 - 0.4642) = 0.3642$

$\frac{\partial o_{o1}}{\partial net_{o1}} = o_{o1}(1 - o_{o1}) = 0.4642(1 - 0.4642)$

$= 0.2487$

$\frac{\partial net_{o1}}{\partial w_5} = 0.5448$

$\frac{\partial E}{\partial w_5} = 0.3642 \times 0.2487 \times 0.5448$

$= 0.0493$

Step 8: $w_5^+ = w_5 - \eta \frac{\partial E}{\partial w_5} = 0.2 - 0.6 \times 0.0493$

$\boxed{w_5^+ = 0.17042}$

Step 9 $\frac{\partial E}{\partial w_6} = \frac{\partial E}{\partial o_{o1}} \times \frac{\partial o_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_6}$

$= 0.3642 \times 0.2487 \times 0.505$

$= 0.0457$

Step 10 $w_6^+ = w_6 - \eta \frac{\partial E}{\partial w_6} = -0.52742$

Step:11 $\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial o_{h1}} \times \frac{\partial o_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}$

$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial o_{o1}} \times \frac{\partial o_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial o_{h1}} \times \frac{\partial o_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}$

$= 0.3642 \times 0.2487 \times w_5 \times o_{h1}(1-o_{h1}) \times i_1$

$= 0.001796$

Step12 $w_1^+ = w_1 - \eta \frac{\partial E}{\partial w_1}$

$= 0.1 - 0.6 \times 0.001796$

$\boxed{w_1^+ = 0.8989}$

Step:13

$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial o_{o1}} \times \frac{\partial o_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial o_{h1}} \times \frac{\partial o_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_2}$

$= -0.0031448$

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

$\boxed{w_2^+ = -0.1981}$

Step:14 $\frac{\partial E}{\partial w_3} = -0.004528$

$w_3^+ = w_3 - \eta \frac{\partial E}{\partial w_3} = 0.4027.$

Step:15 $\frac{\partial E}{\partial w_4} = 0.00792$ $\boxed{w_4^+ = 0.1952}$

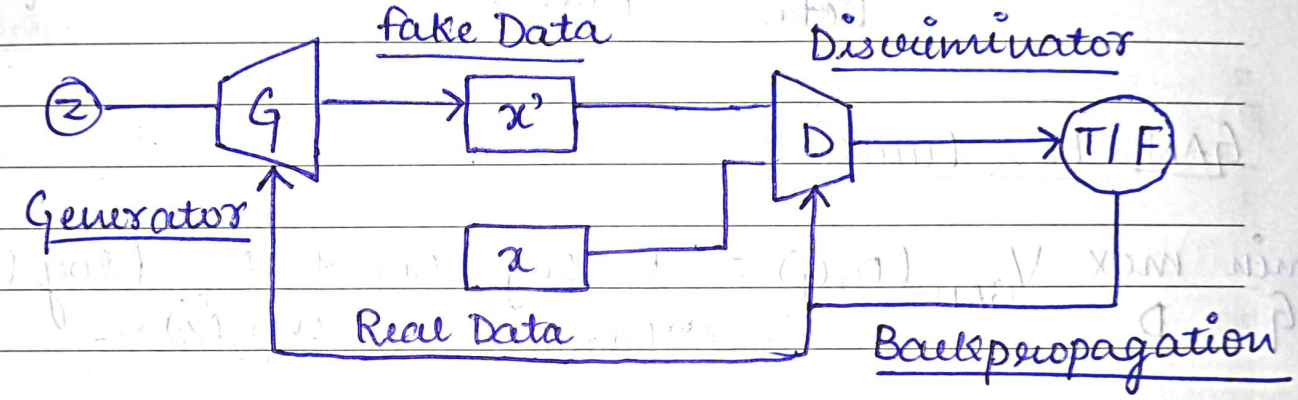
Generative Adversarial Network (GAN)

Generative → Generate (Create)

Adversarial → Cheater (Investigator)

Network → Neural Network

Components

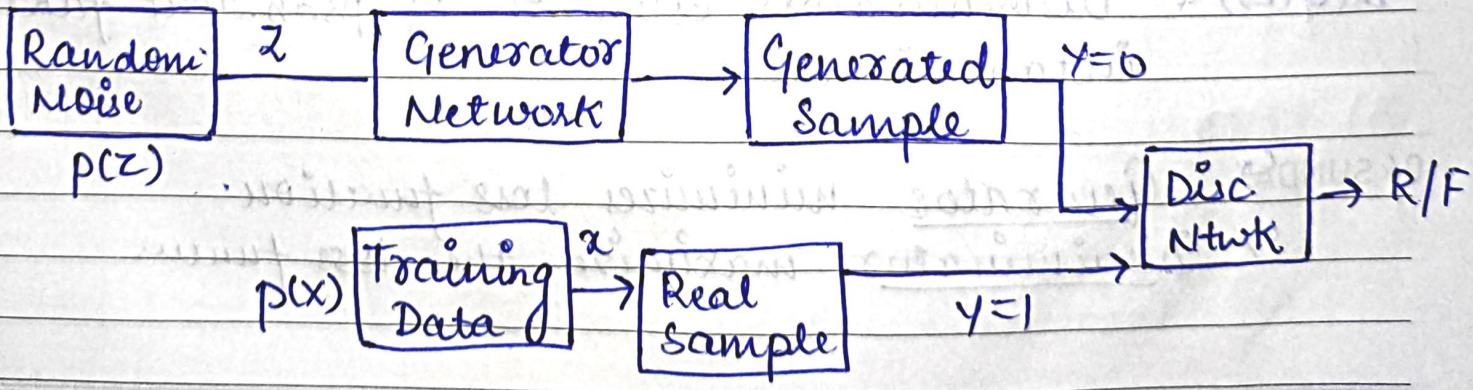


Generator:

- works on unsupervised learning approach
- It will generate fake data

Discriminator:

- works on supervised learning approach
- Discriminator does the classification



Training of GAN

① Define a problem

② Select Arch. of GAN

③ Train Disc. on Real Data.

④ Generate Fake IIP for Generator.

⑤ Train Disc. on Fake Data

⑥ Train Generator with the O/P of Disc

GAN Loss function

$$\min_G \max_D V_{GAN}(D,G) = \mathop{E}_{x \sim p_{data}(x)} \log D(x) + \mathop{E}_{z \sim p_z(z)} (\log(1 - D(G(z))))$$

Formula is derived from cross-entropy b/w real and generated distributed.

$D(x)$ = Discriminator's estimate of prob that real data instance is real.

Ex = expected value over all real data instances.

$G(z)$ = Generator's op when given noise z.

$D(G(z))$ = Discriminator's estimate of prob. that fake instance is real.

06 SUNDAY

Generator minimizes loss function

↳ Discriminator maximizes the loss funcn.

Advantages

1) Better Modelling of Data Distribution.
2) GANs can train any Kind of Generator Network.
3) No Need to use Markov chain repeated sample.

Disadvantages

1) Hard to train, unstable
2) Good sync req. btw Disc & Generator.
3) Mode collapse issues

21 JUNE

08 DAY 159-206 | Wk 24 TUESDAY

Semi-Supervised Learning

1) The Algo is trained upon a combination of labelled and unlabelled data.

Small labelled Data	Large amount of Unlabelled Data

2) Semi-supervised Algo assumptions:

a) Continuity assumptions: The points which are closer to each other are more likely to have the same output lebal.

b) Cluster Assumptions: Data can be divided into discrete clusters & points in same cluster are more likely to share an output label.

c) Manifold Assumptions: Data lies approximately on a manifold of much lower dimension than the input space.

3) Applications:

a) Speech Analysis
b) Internet Content classification
c) Protein seq. classification.

Principal Component Analysis

1) PCA is used to reduce dimensions of data without much loss of information.

2) PCA is an orthogonal linear transformation that transfer data to new coordinator system such that greatest variance by any projection of data comes to lie on first coordinate.
Second greatest variance lies on second coordinate & so on.

Algorithms:

Step:1 Get Data
Step:2 Subtract the means
Step:3 Calculate covariance Matrix.
Step:4 Calculate EV & Eigen values of Covariance Matrix
Step:5 Choosing components & forming a feature vector.
Step:6 Deriving the new Dataset.

Advantages

1) Lack of Redundancy
2) Reduced compexity in img grouping using PCA.
3) Reduction of noise since max. variance.

Disadvantages

1) Covariance matrix is difficult to evaluate.
2) Simplest invariance Couldnot be captured by PCA.

21' JUNE $S = \sum_{K=1}^{n} (x_K - m)(x_K - m)^T$

10 DAY 161-204 | Wk 24 THURSDAY

closely related to PCA & FA reg.

Linear Discriminant Analysis

LDA is used for data classification & dimensionality reduction

It maximizes the ratio b/w class variance to within the class variance in any particular dataset

Algorithm

Step:1 Compute within-class & between class scatter Matrix

Step:2 Compute eigen values & EV for scatter Matrix.

Step:3 Sort the Eigen values & selecting top k

Step:4 Create a new matrix
eigenvectors → k eigenvalues.

step:5 Obtain new features using dot product of data & matrix in step 4

Advantages

1) Simple, fast, Portable algo, better than some logistic reg.

Disadvantages

1) Not scalable.
2) Inefficient to update the Model.
3) Require Normal distribution assumption on features.

Application:

↳ face Recognition ↳ Medical
↳ Customer Identification.

PCA

1) Unsupervised learning Algorithm

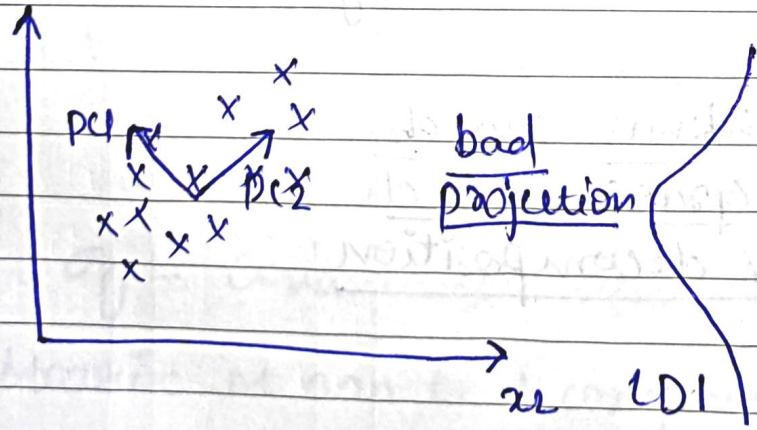
2) Feature classification

3) Ignores class labels & focuses on finding the PC that maximizes the variance in given data.

4)

5)

PCA: Component axes that maximize the variance.



bad projection

LDA.

1) Supervised learning Algorithm.

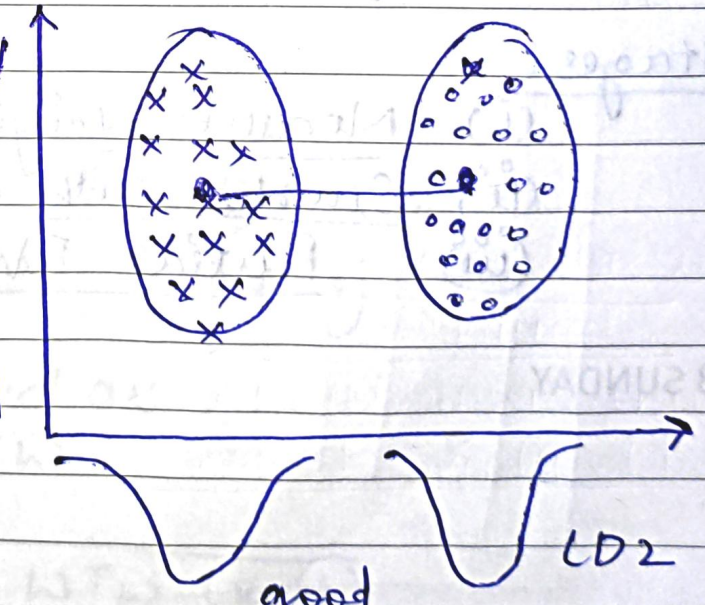
2) Data Classification.

3) It finds linear discriminants that represents those axes which maximizes separation b/w diff. classes

4) Perform better multi-class classification.

5)

LDA: maximizing the component axes for class-sep.



good projection

Manifold learning

↳ For Non-linear dimensionality Reduction.
↳ It is based on the idea that ~~idea that~~ the dimensionality of many datasets are only artificially high.

1. Isomap Algorithm
2. Locally linear embedding
3. Hessian Eigen Mapping
4. Spectral embedding
5. Local Tangent space Alignment
6. Multi-dimensional Scaling
7. t-distributed stochastic Neighbour embedding

Isometric Mapping Algorithm

Extension of Multi-dimensional Scaling.

Stages:

(i) Nearest Neighbour search.
(ii) Shortest path graph search
(iii) Partial EV decomposition.

13 SUNDAY

Metric learning

Euclidian distances are less meaningful in high dimensions. They fail to capture the non-linearity in data because:

a) they represent isotropic distance matrix
b) these distances don't capture class structure.

Mahalanobis Distance Matric :

In a 'd' dimensional space : $x, y \in \mathbb{R}^d$

$$D(x,y) = \sqrt{(x-y)^T M (x-y)}$$

M = Inverse of covariance matrix

Covariance Matrix in 'd' dimensional space :

$$M = \Sigma^{-1} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd} \end{bmatrix}^{-1} \quad dxd.$$

σ_{ij} = Covariance b/w variables i, j

Matrix M can be decomposed as

$$M = W^T W$$

Mahalanobis Distance

$$D(x,y) = \sqrt{(x-y)^T W^T W (x-y)}$$

$$D(x,y) = \|Wx - Wy\|_2$$

21' JUNE

15 DAY 166-199 | Wk 25 TUESDAY

MAY '21

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JUNE '21

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

unsupervised
Data specific
lossy

Autoencoders

↳ Autoencoders are specific type of feed forward Neural Networks where the input is same as the output.

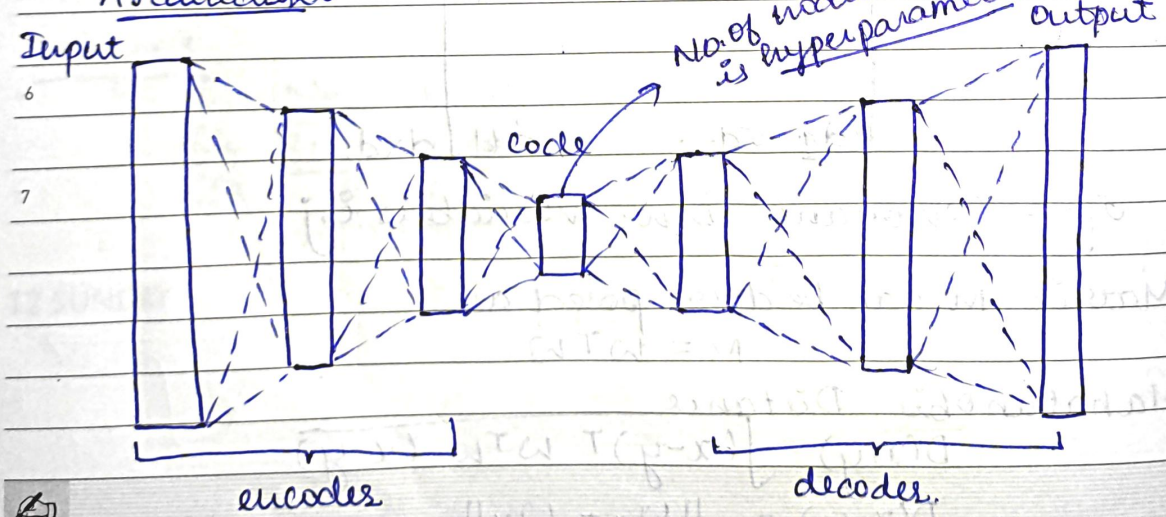
↳ They compress the input into lower dimension code and then reconstruct the output from this representation.

↳ Autoencoder Components.

a) Encoder compress the I/P & produces the code
b) Decoder then reconstruct the input using this code.

↳ Need 3 things : Encoding Method, Decoding Method, Loss function.

Architecture :



Hyperparameters:

1) Code size : No. of nodes in middle layer.
2) No. of layers:
3) No. of Nodes per layer
4) Loss function

Hyperparameter Optimization: To find the hyperparameters that delivers the best performance as measured on validation set.

	Model	Overview	Hyperparameters.
1)	C4.5	Decision tree	$C = \{0.05, 0.10, 0.20 \ldots\}$
2)	NNET	3 layer NN	size (4, ..28) decay = {0.10, 0.20}
3)	KNN	KNN	$C = \{2^*(0..7)+\}$
4)	RF	Random forest	mtry = {10, 50, 100, 200 ... 1000)
5)	SVM	SVM	$C = \{2^{-6}, \ldots 2^{10}\}$

Application Autoencoders:

(i) Image Generation
(ii) Image colourization
(iii) Dimensionality Reduction
(iv) Data compression
(v) Feature Extraction
(vi) Image Denoising

Types

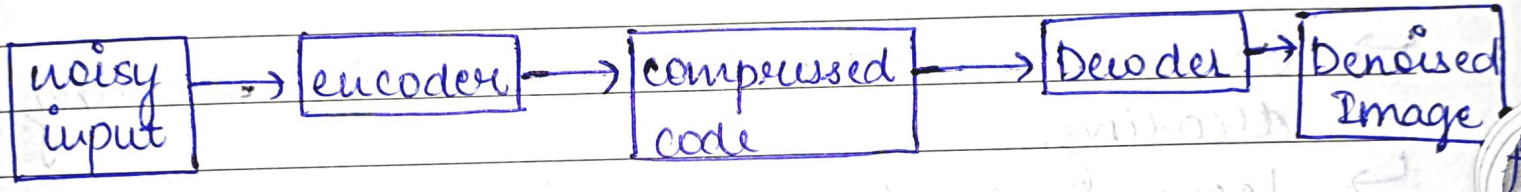
(i) Denoising
(ii) Sparse
(iii) Deep
(iv) Contractive
(v) Undercomplete.
(vi) Convolutional
(vii) Variational

Autoencoder (Types)

1) Denoising Autoencoder :

↳ These autoencoders take partially corrupted input while training to recover the original undistorted input

↳ Data corruption can be done by making some inputs zero.



2) Sparse Autoencoders : (Hidden nodes > input nodes)

Sparsity constraint is applied on Hidden layers to prevent output layer copy input layer.
It takes highest Activation value in HL & zero out rest of hidden nodes. at a time.

3) Deep Autoencoders : (Restricted Boltzman Machine)

Consist of 2 deep belief Net, one for encoding & other for decoding (Typically 4-5 layers).

4) Contractive Autoencoders :

↳ To have a robust learned representation which is less sensitive to small variation in data.

↳ This can be done by applying penalty term in loss function

#(Frobenius Norm of Jacobian Matrix)

5) Undercomplete Autoencoders:

↳ To capture most impt features present in data.
↳ Have smaller dimension for hidden layer as compared to hidden layer.
↳ penalize $g(f(x))$ for being diff. from ip x.

6) Convolutional Autoencoders:

↳ Convolution networks are used in encoding & decoding.
↳ learn to encode the ip in set of simple signals & try to reconstruct ip from them.

7) Variational Autoencoders:

↳ Make strong assumptions concerning the dist. of latent variables.

1) Denoising (DAE) $L = |\tilde{x} - g(f(x))|$
2) Sparse SAE $L = |x - g(f(x))| + \Omega(h)$
3) Deep
4) Contractive (CAE) $L = |x - g(f(x))| + \lambda \|J_f(x)\|_F^2$
5) Undercomplete (UAE) $L = |x - g(f(x))|$
6) Convolutional (ConvAE)
7) Variational (VAE)

Batch Normalization

↳ A technique for training DNN that standardizes the inputs to a layer for each mini-batch.

↳ BN makes the input to each layer to have zero mean and unit variance

↳ Stabilizing the learning process and reducing the no. of training epochs required to train deep Networks.

Normalization of Input:

h_i = layers $\quad \mu = \frac{1}{m} \Sigma h_i$ $\quad h_i = \frac{h_i - \mu}{\epsilon + \sigma}$

$$\sigma = \sqrt{\frac{1}{m} \Sigma (h_i - \mu)^2}$$

$$\boxed{h_i = \frac{(h_i - \mu)}{\sigma + \epsilon}} \qquad \epsilon = \text{smoothing term}$$

Rescaling of offset:

$$\boxed{h_i = \gamma h_{i(norm)} + \beta}$$

γ = rescaling $\quad \beta$ = shifting

Advantage

1) Reduce ~~Invariant to~~ internal covariant shift
2) Reduce dependencies of gradients on the scale of parameters and their initial values.
3) Allow use of saturating nonlinearities & higher learning rate.

Disadvantages:

1) Computational overhead during training
2) Difficult to estimate mean & SD.
3) Batch size ≠ 1

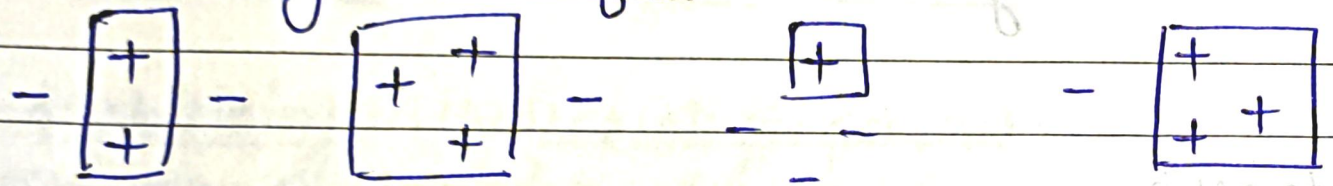
① Reduce Internal covariance shift
② Reduces dependencies of gradient on the scale of parameters
③ Allow use of saturating Non-linearities & Higher learning rate.

① Computational overhead
② Difficult to estimate mean & SD
③ Batch size ≠ 1.

VC Dimension (Vapnik & Chervonenkis)

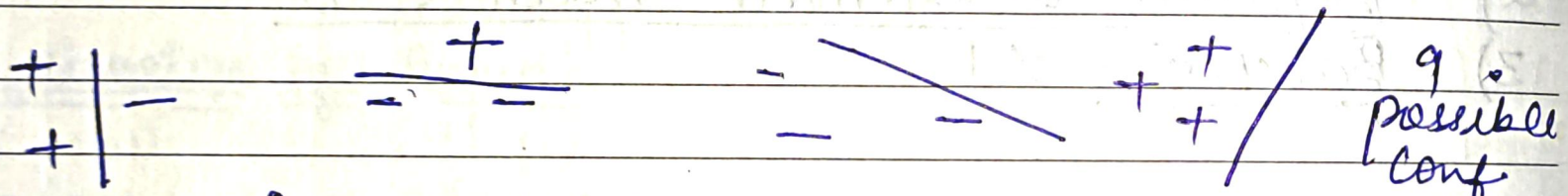
VC Dimension of a Hypothesis set H, denoted by dvc is the largest value of N for which H can shatter all N training examples.

Rectangle classifier



4×4 16 possible conf.

VC dimension is 4



9 possible conf

VC dimension is 3.

PCA

	X	Y	$x-\bar{x}$	$(x-\bar{x})^2$	$y-\bar{y}$	$(y-\bar{y})^2$	$(x-\bar{x})(y-\bar{y})$
	2.5	2.4	0.69	0.4761	0.49	0.2401	0.3381
	0.5	0.7	-1.31	1.7161	-1.21	1.4641	1.5851
	2.2	2.9	0.39	0.1521	0.99	0.9801	0.3861
	1.9	2.2	0.09	0.0081	0.29	0.0841	0.0261
	3.1	3.0	1.29	1.6641	1.09	1.1881	1.4061
	2.3	2.7	0.49	0.2401	0.79	0.6241	0.3871
	2.0	1.6	0.19	0.0361	-0.31	0.0961	-0.0589
	1.0	1.1	-0.81	0.6561	-0.81	0.6561	0.6561
	1.5	1.6	-0.31	0.0961	-0.31	0.0961	0.0961
	1.1	0.9	-0.71	0.5041	-1.01	1.0201	0.7171
$\bar{x}$	1.81	1.91		$\frac{5.549}{9}$		$\frac{6.449}{9}$	$\frac{5.339}{9}$
				0.6165		0.7165	0.6154

$$C_{2\times 2}\begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix} = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

$$cov(x,y) = \sum_{i=1}^{n} \frac{(x_i-\bar{x})(x_{yi}-\bar{y})}{n-1}$$

Eigen values

$$|C-\lambda I| = 0 \qquad \begin{vmatrix} 0.6165-\lambda & 0.6154 \\ 0.6154 & 0.7165 \end{vmatrix} = 0$$

$$\lambda^2 - 1.33\lambda + 0.0630 = 0 \qquad \lambda_1 = 1.28393$$

$$\lambda_2 = 0.049068$$

	x	y	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x})^2$	$(y-\bar{y})^2$	$(x-\bar{x})(y-\bar{y})$
9	2	1	-2.5	-4	6.25	16	10.0
	3	5	-1.5	0	1.25	0	0
10	4	3	-0.5	-2	0.25	4	1
	5	6	0.5	1	0.25	1	0.5
11	6	7	1.5	2	1.25	4	3
	7	8	2.5	3	6.25	9	7.5
12	27	30			17.5	34	22

$\mu_1 = \frac{27}{6} = 4.5$

$\mu_2 = \frac{30}{6} = 5$

Covariance Matrix $= \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$

Characteristic equ: $|M - \lambda| = 0$

$$\begin{vmatrix} 2.92-\lambda & 3.67 \\ 3.67 & 5.67-\lambda \end{vmatrix} = 0$$

$\lambda^2 - 8.59 + 3.09 = 0$

$\lambda_1 = 8.22$

$\lambda_2 = 0.38$ (very small) left out.

Eigen Vector corresponding to greatest eigen value is PC

$\therefore MX = \lambda X$

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 8.22 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$2.92x_1 + 3.67x_2 = 8.22x_1$$
$$3.67x_1 + 5.67x_2 = 8.22x_2$$

$$5.3x_1 = 3.67x_2 \quad -①$$
$$3.67x_1 = 2.55x_2 \quad -②$$

from ①&② $x_1 = 0.69x_2$

from ②) $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

Principal component $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

⇒ Plot graph

Ques Let feature vector $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$

feature vector gets transformed to $= (v_1)^T (FV - MV)$

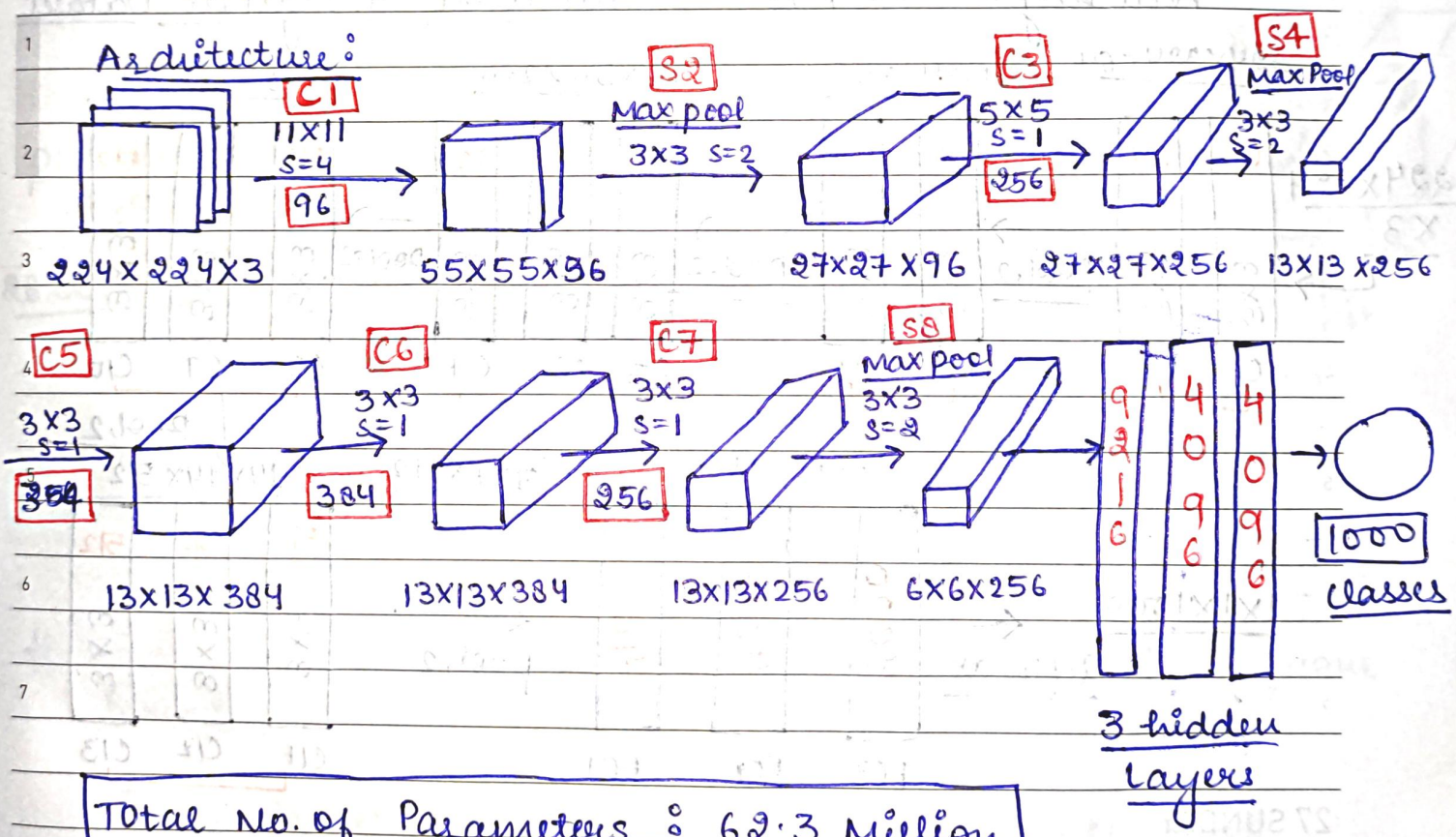
$$= \begin{bmatrix} 2.55 & 3.67 \end{bmatrix} \left[\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 4.5 \\ 5 \end{bmatrix} \right]$$

$$= -21.055$$

AlexNet

↳ first CNN which used GPU to boost performance
↳ 5 convolutional layers
3 Max Pooling Layers
2 Normalization Layers. ReLU
2 fully connected Layers
1 Softmax layers.

Architecture:



Total No. of Parameters : 62.3 Million

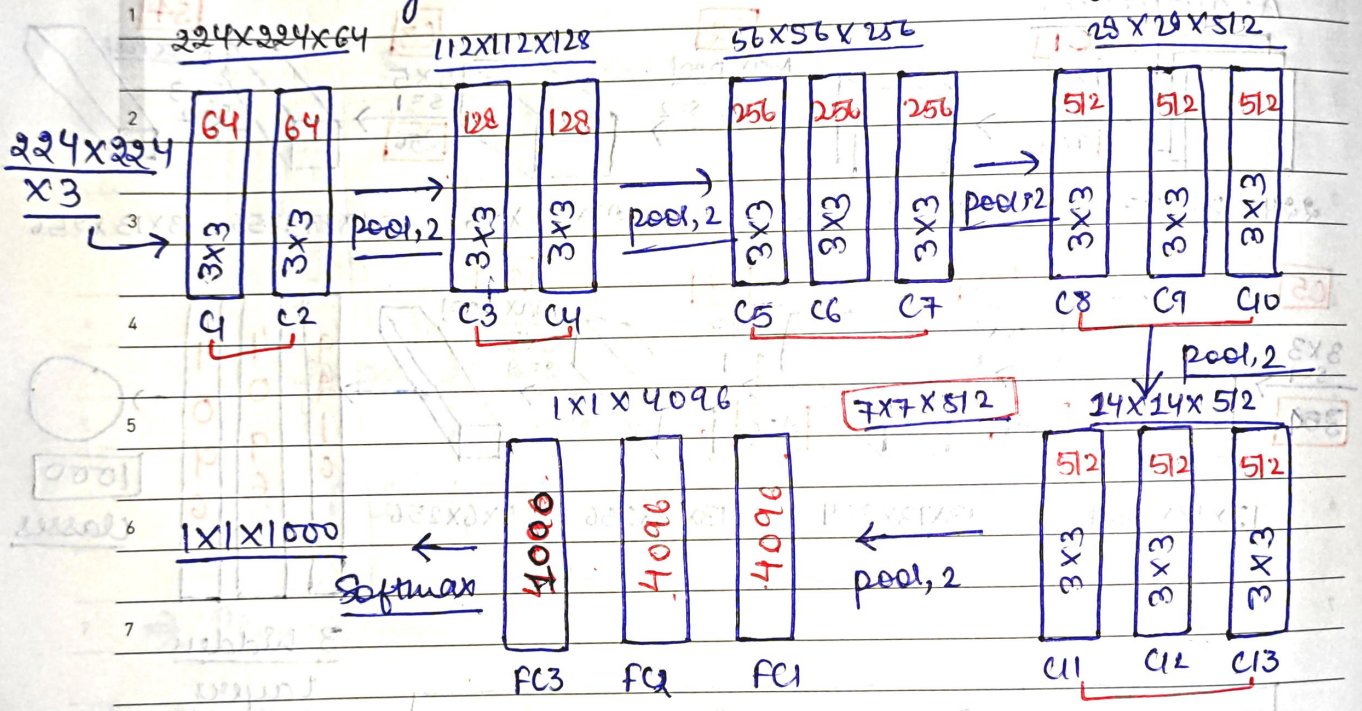
Visual Geometry Group Architecture

↳ VGG16 has total of 16 layers, only convolution and pooling layers are used.

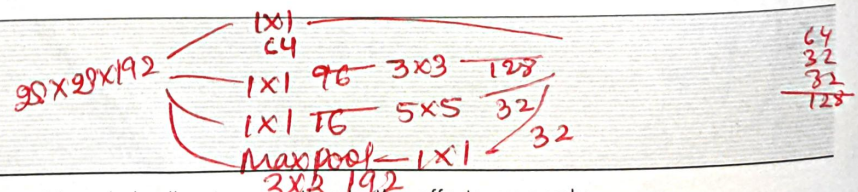
↳ Kernel : 3X3 Stride = 1

Max pooling : 2X2 Stride = 2

↳ Accuracy : 92.7% trained on ImageNet Dataset

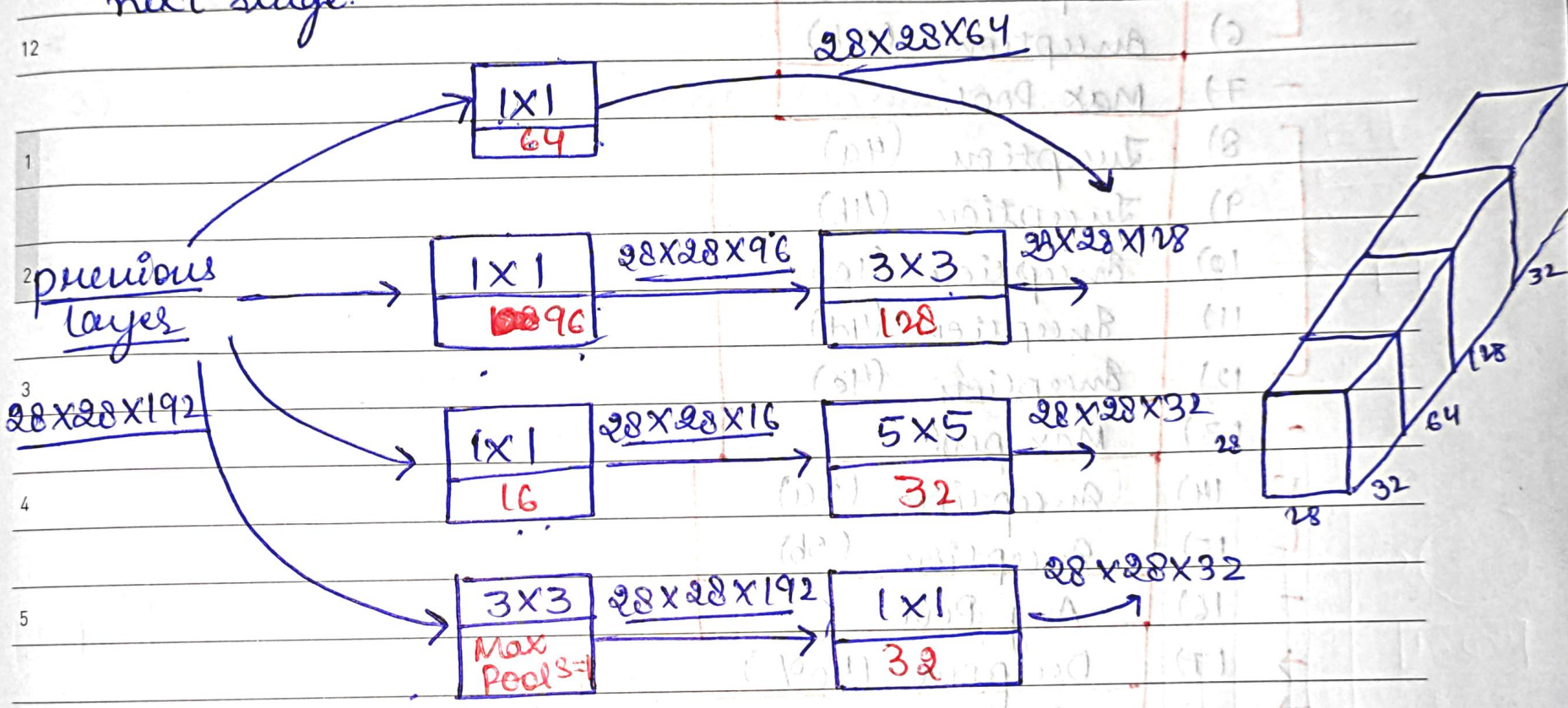


27 SUNDAY



Inception Architecture

Inception layer is the combination of all those layers (1X1 conv layers, 3X3 conv layer, 5X5 conv layer) with their output filter banks concatenated into a single output vector forming the ip of next stage.



Same padding helps in getting contribution from boundary pixels.

GoogleNet devised Inception Module that Approximate a Sparse CNN with a Nomal Dense Construction.

GoogLeNet

Inception Architecture : 27 Layer Deep.

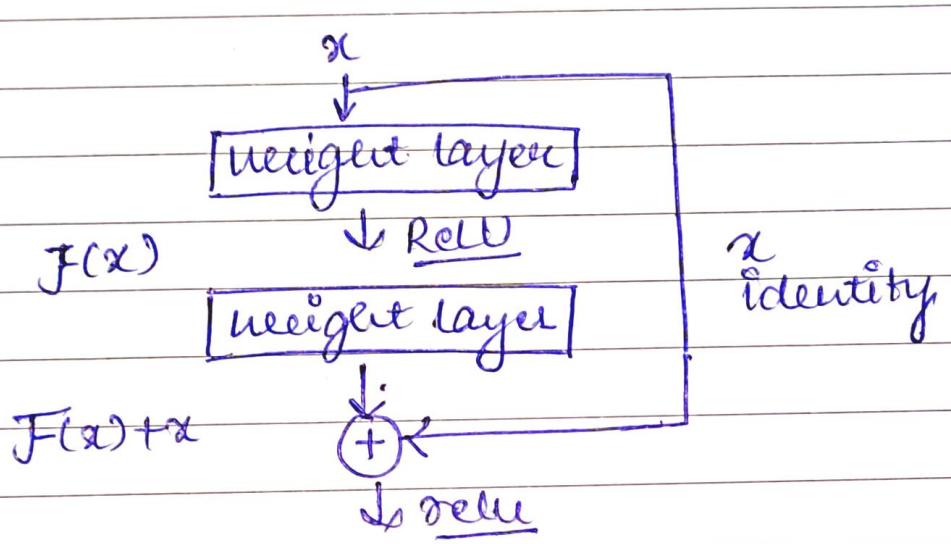
1) Convolution
2) Max Pool
3) Convolution
4) Max Pool
5) Inception (3a)
6) Inception (3b)
7) Max Pool.
8) Inception (4a)
9) Inception (4b)
10) Inception (4c)
11) Inception (4d)
12) Inception (4e)
13) Max pool
14) Inception (5a)
15) Inception (5b)
16) Avg Pool
17) Dropout (40%)
18) Linear
19) Softmax.

ResNet (Residual)

1) In order to solve the problem of vanishing gradient this Network uses the technique of skip connections

2) Skip training from few layers & connect them directly to output.

3) Network fit Residual Mapping



21' JULY Geometric Invariance

02 DAY 183-182 | Wk 27 FRIDAY

JUNE '21							
M	T	W	T	F	S	S	
	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30					

JULY '21							
M	T	W	T	F	S	S	
			1	2	3	4	
5	6	7	8	9	10	11	
12	13	14	15	16	17	18	
19	20	21	22	23	24	25	
26	27	28	29	30	31		

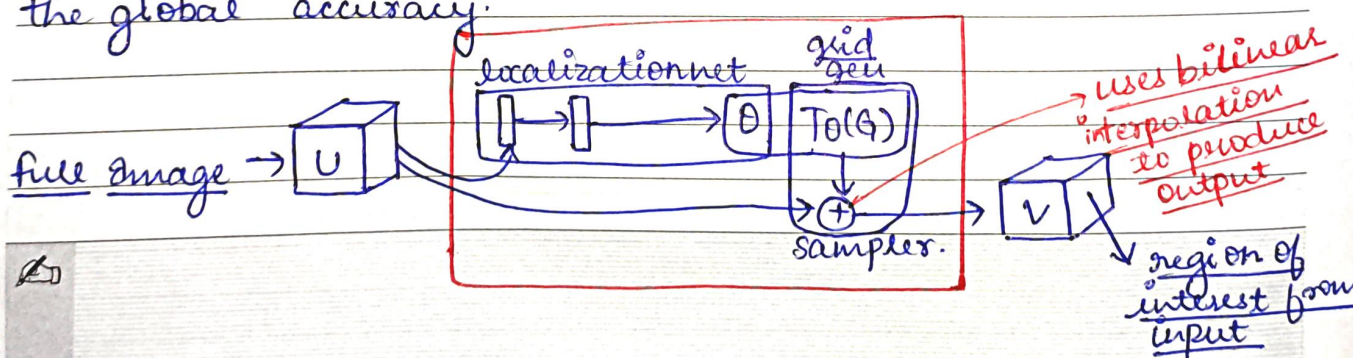
Spatial Transformer Networks

STN allow a neural Network to learn how to perform spatial transformation on the input image inorder to enhance the geometric invariance of Model.

eg. it can crop a region of intersection, scale, correct the orientation of image.

Components:

Localization Network	Grid Generater	Sampler
Regular CNN which regress transformation parameters Transformation is never learned explicitly from dataset, instead ntwk learns automatically the Spatial trans. that enhances the global accuracy.	Generate a grid of coordinates in input images corresponding to each pixel from output image.	It uses the parameters of transformation and applies it to the input img



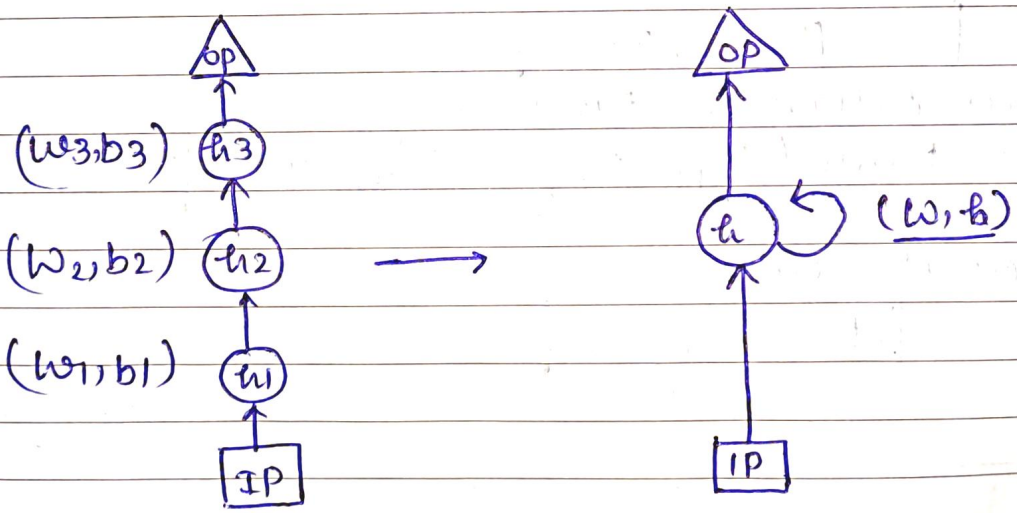
Recurrent Neural Network

↳ RNN are a type of NN where the output from previous step are fed as input to current step.

↳ RNN has hidden state (Memory) which remembers all the info about what has been calculated.

eg. Let DN having one IP, 3 HL, one OP layer having weight & bias as (w_i, b_i), $i \to \{1,2,3\}$

RNN converts independent activations into dependent activations by providing same weight & bias to all layers, this reducing the complexity of increasing parameters, & memorizing each previous output by giving it as IP to next HL



SUNDAY 04

A) Calculate current state: $h_t = f(h_{t-1}, x_t)$

$h_t = f(h_{t-1}, x_t)$

$h_t \to$ current state
$h_{t-1} \to$ previous state
$x_t \to$ IP state.

$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$

B) Applying activation function.

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} X_t)$$

W_{hh} → wt at recurrent neuron

W_{xh} → wt at input neuron

C) Calculate output

$$Y_t = W_{hy} h_t$$

Y_t → output

W_{hy} → wt at op neuron.

1) Calculate current state:

$$h_t = f(h_{t-1}, x_t)$$

2) Apply Activation function

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

3) calculate output

$$Y_t = W_{hy} h_t$$

Advantages of RNN

1) Remembers info → useful for series prediction → LSTM
2) RNN used with conv. layers to extend powerful pixel Neighborhood.

Disadvantages of RNN

1) Gradient vanishing and exploding problem
2) Training RNN is difficult.
3) long sequence if using tanh / ReLU as AF.

Applications → Text Generation → Machine Translation
→ Speech Recognition → Sentiment Analysis → Anomaly Detection
→ Visual search → Face Recognition → Stockprice forecasting

LSTM Network

↳ Special Kind of RNN, capable of learning long-term dependencies.

↳ LSTM make small modifications to the info by Multiplication and Addition.

↳ LSTM flow information in the form of cell states

LSTM deal with both LTM and STM for making calculations simple and effective.

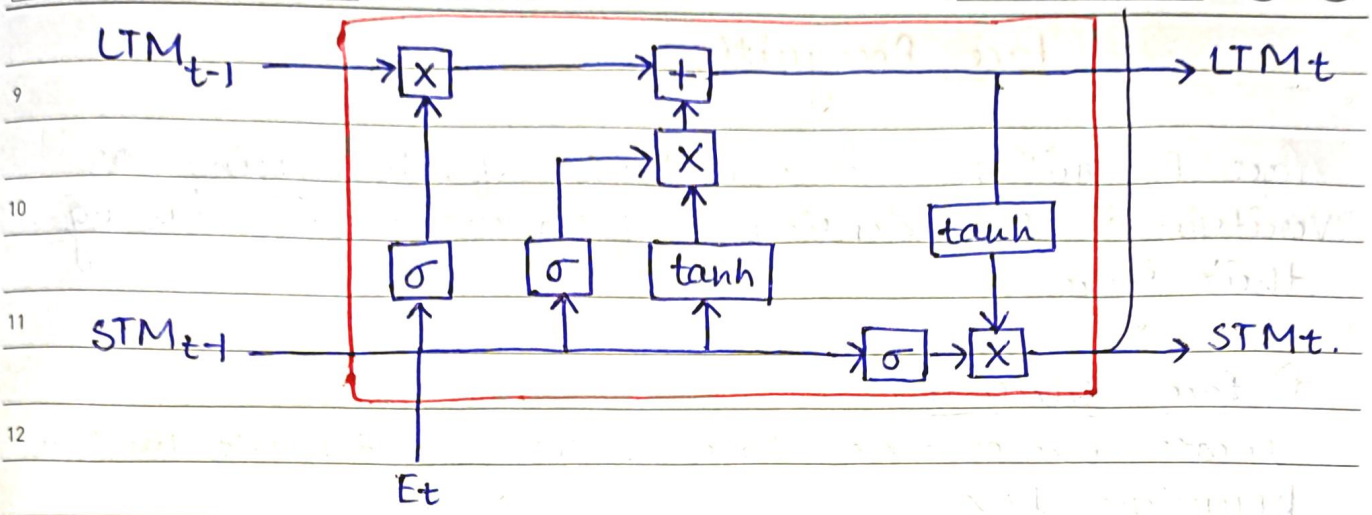
1) Forget Gate: LTM goes to FG and it forgets info that is not useful.

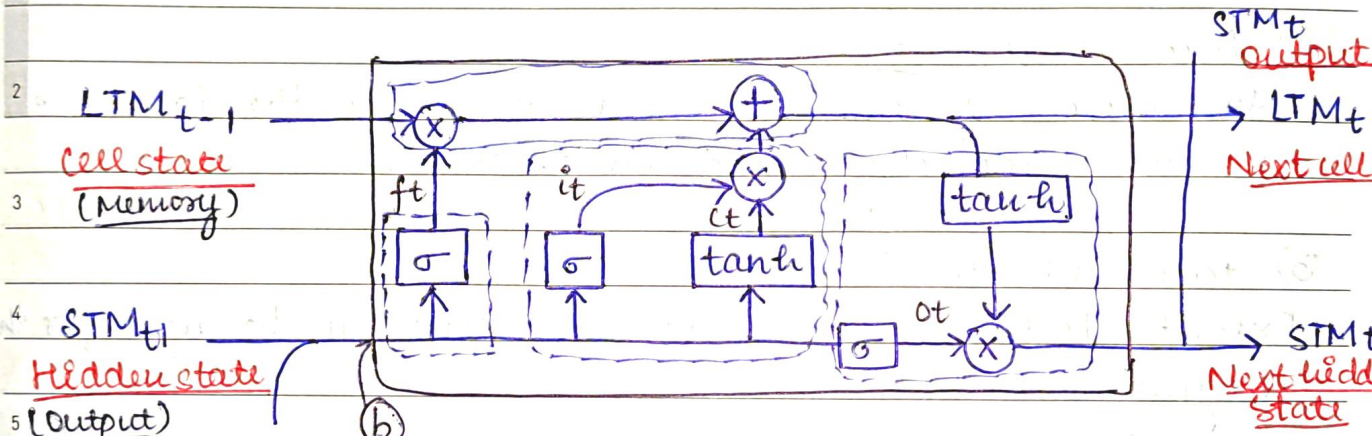
2) Learn Gate: Current input and STM combined so that necessary info. we recently learned from STM can be applied to current IP.

3) Remember Gate: LTM info. that we haven't forget and STM + Event in RG which works as updated LTM.

4) Use Gate: This gate also uses LTM, STM & event to predict O/P of current event which works as an update STM

3-σ 2-t
3-x 2-tanh





Applications:

1) Time series prediction
2) Robot Control
3) Speech Recognition
4) Handwriting Recognition
5) Music Composition
6) Drug Design
7) Object cosegmentation
8) Market prediction
9) Semantic parsing
10) Protein Homology detect

Face Recognition

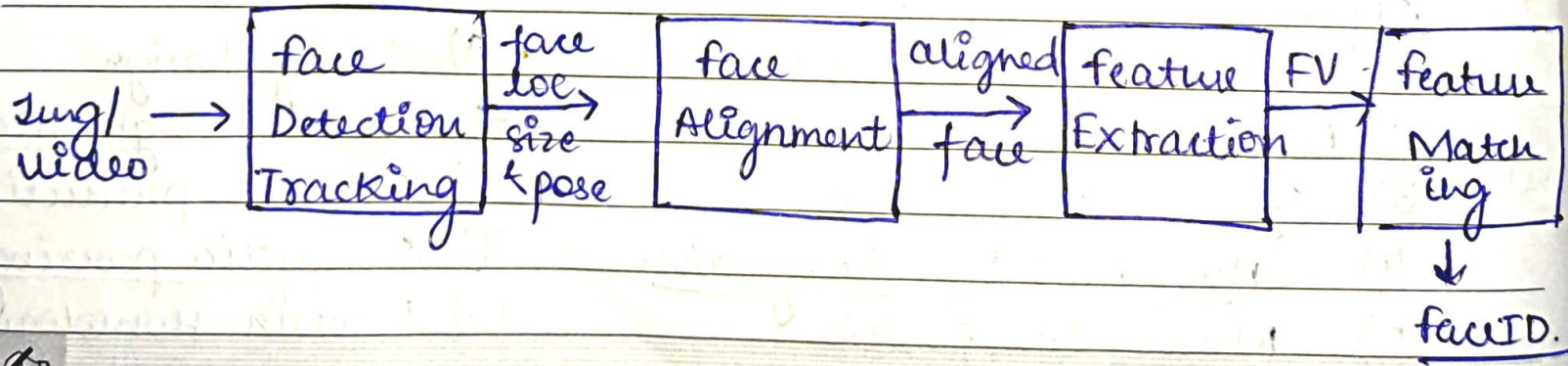
Face Recognition is a method of identifying or verifying the identity of an individual using their face.

1) Face detection:
Locate one or more faces in image and mark with a bounding box.

2) Face alignment:
Normalize face to be consistent with the database such as geometry and photometrics.

3) Face extraction:
Extract features from the face that can be used for the recognition task.

4) Face Recognition:
Perform matching of face against one or more known faces in a prepared database.



Audio WaveNet

1) WaveNet is deep Neural Network for generating raw audio. It was created by researchers at London-based AI firm DeepMind

2) The technique is able to generate relatively realistic sounding human like noises by directly modelling waveforms using a NN Method trained with recordings of real speech.

3) Tests with US English and Mandarin ~~reporti~~ reportedly showed that system outperforms Google's best existing (TTS) system

4) WaveNet is an audio generative model based on the Pixel CNN.
On this generative model, each audio is conditioned on previous audio sample.
The conditional prob. is modelled by a stack of convolutional layers.

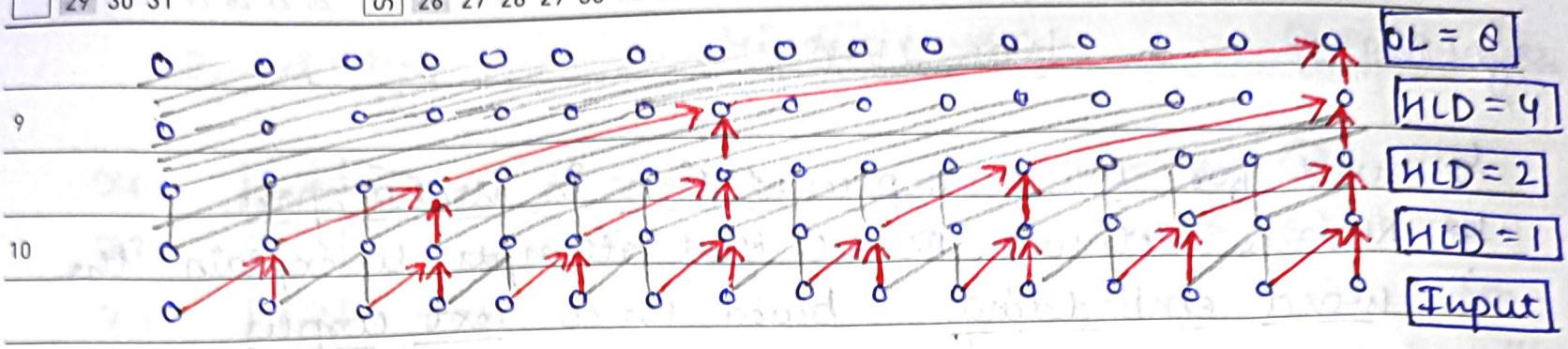
5) This network doesnot have pooling layers and output of model has same time dimensionality as IP.

PCNN dilations

④ Output - Ouput Same dimension

① Deep Mind ③ AGM
② Pixel CNN

OL - output dilation
HLD - Hidden Layer Dilation



6) fully convolutional NN, where convolutional layers have various dilations factor that allows its receptive field to grow exponentially with depths and covers thousands of time steps.

7) Joint probability of waveform $x = \{x_1, \dots x_T\}$ is factorized as product of conditional probabilites

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_1, \dots x_{t-1})$$

each audio sample x_t is therefore conditioned on samples at all previous timesteps.

21' JULY word embedding based on text corpus

14 DAY 195-170 | Wk 29 WEDNESDAY

JUNE '21

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

JULY '21

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Word2Vec

↳ used for word representation in vector space

↳ Neural Network Model that attempts to explain the word embeddings based on a text corpus.

↳ 2 layer Neural network that processes text.
Input: text corpus.
Output: set of vectors.

↳ Output of W2V NN is vocabulary in which each item has vector attached, which can be fed into DLN to detect relationship b/w words.

↳ To implement word embedding using Word2Vec:
(i) Continous Bag of words (CBOW)
(ii) Skip-Gram.

1) CBOW Model: Takes the context of each word as input and tries to predict the target word in its neighbourhood.

Objective function $-\log\left(P\left(\frac{w_o}{w_i}\right)\right)$

w_o = output words

w_i = context words.

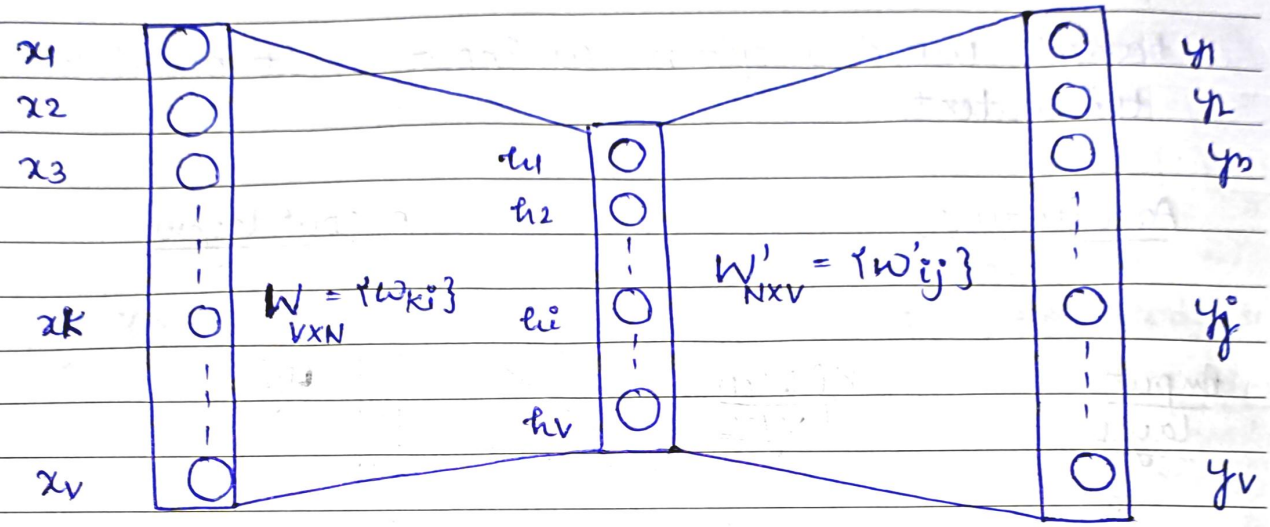
$$p(w_o|w_i) = \frac{\exp(v'^{T}_{w_o} v_{w_i})}{\sum_{w=1}^{W} \exp(v'^{T}_{w} v_{w_i})}$$

Disadv
① Slower to train
② takes avg. of context

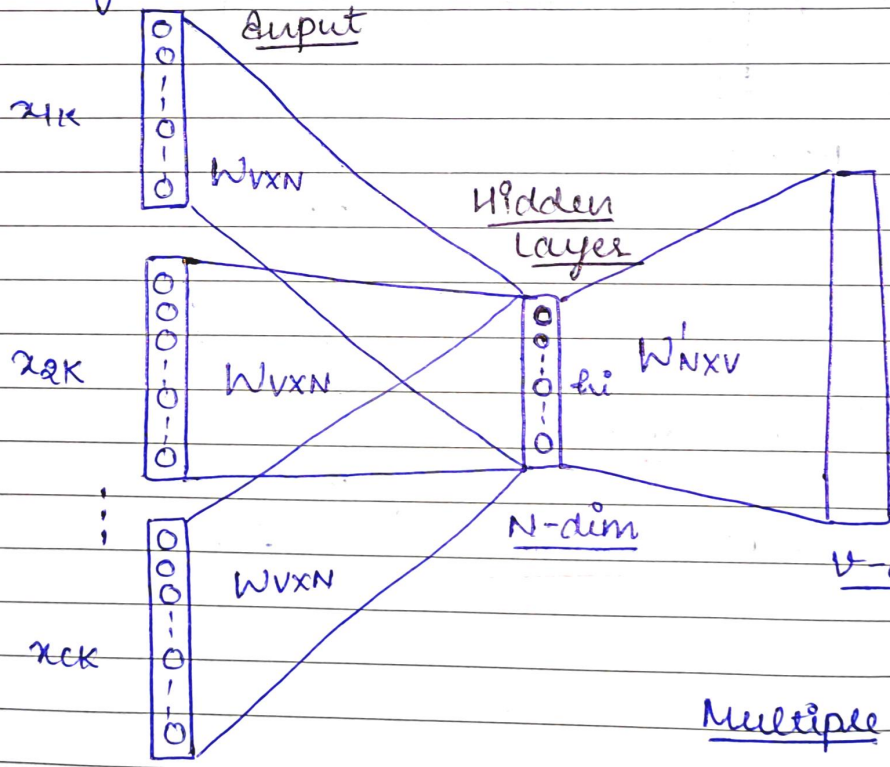
Advantages
① Faster
② Better for Frequent words representation
③ Low on Memory

JULY '21
THURSDAY 15
DAY 196-169

Input layer | Hidden layer | Output layer



Single context-word



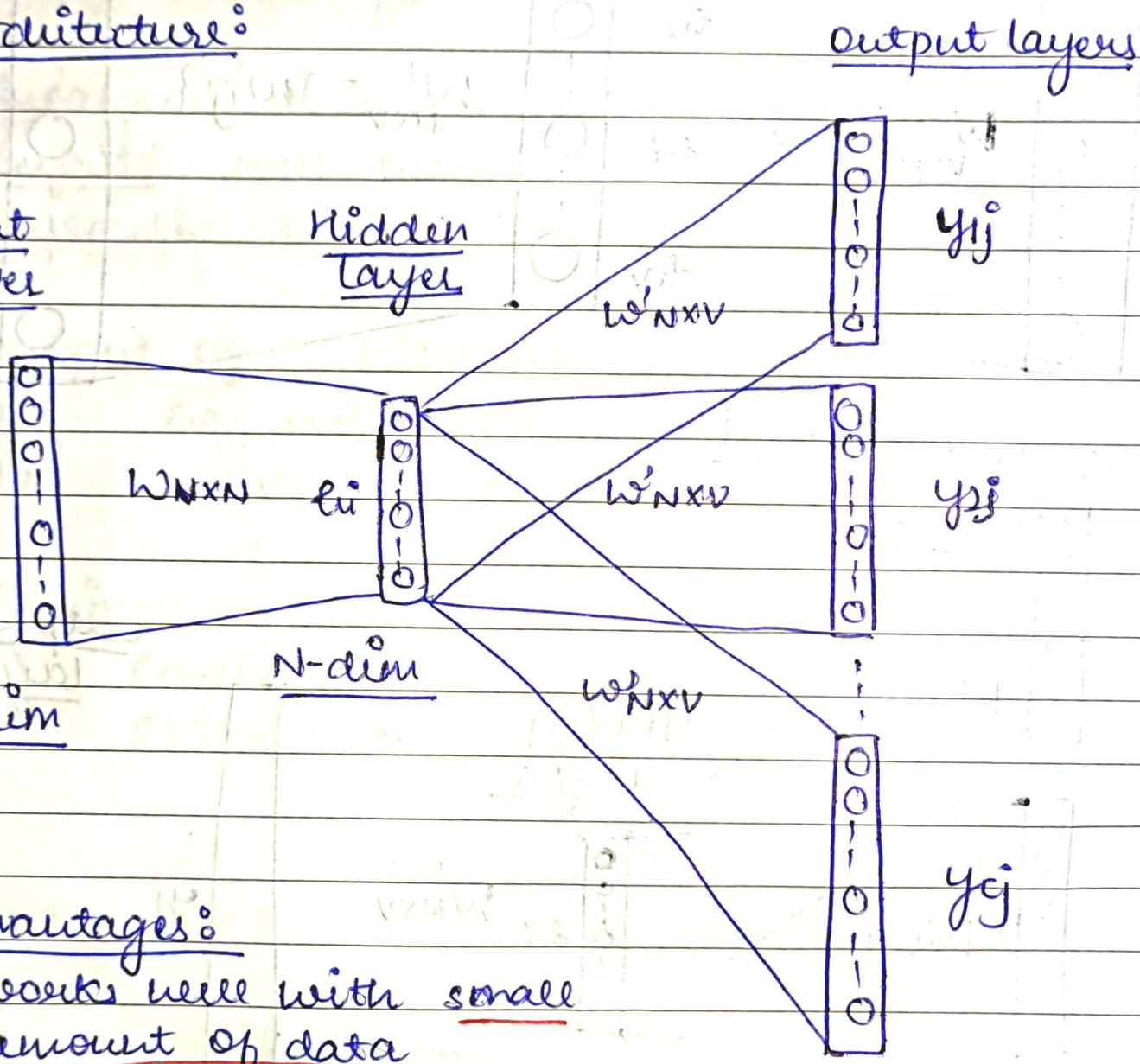
Multiple context words

C x V-dim

2) Skip-Gram Model

target word is given as input, it predicts the context.

Architecture:



Advantages:

1) works well with small amount of data
2) Represent rare words well.

Image Captioning

The process of generating textual desp. from an image based on objects and actions in image.

Divide the task into 2 Modules:

Image Based Model

Extract the features and nuances out of an Image.

↓

Encoder

↓

Convolutional NN

Language Based Model

Translate features and objects given by IBM to a natural sentence)

↓

Decoder

↓

Recurrent NN

ImageNet

↳ Imagenet is a large dataset of images which are originally labelled with synsets of WordNet lexicon tree.

↳ For computer vision Research (object detection, image classification & object localization).

↳ 14,197,122 images organized into 21,841 subcategories. These sub-categories can be considered as subtree of 27 High-level categories.

14,197,122 images → 21,841 categories → 27 high level categories

Bioinformatics

Bioinformatics is the field of study that deals with computational & Mathematical approaches to analyse & extract knowledge from Biological data.

Application of ML in Bioinformatics.
↳ Diseases prediction
↳ Diagnosis Analysis
↳ Survival Analysis

Bioinformatics Tools:

1) Homology and similarity Tools:
Computational Analysis → protein & DNA seq → similarity → common Ancestry

2) Protein function Analysis:
Protein (size, structure physiochemical properties) → Identification / seperation → mass spectroscopy / electrophoresis

3) Structural Analysis:
Seismic Response → vulnerability of Building → Topologgies

4) Sequence Analysis:
RNA, DNA, peptide seq → analytics → search / Align

Advantages

1) provides systematic info about genomics proteomics and metabolomics

2) helps in finding out evolutionary relationship between 2 species.

3) Gene Mapping & Sequencing

4) To reconstruct genes from Expressed sequence tags.

5) Storing, organizing & indexing huge database.

Disadvantages

1) Require sophisticate lab - require lots of funds

2) Need uninterrupted power supply.

3) Regular check of computer viruses.

4) High Maintence

LSTM Network

1) Special kind of RNN, capable of learning long-term dependencies.
2) makes small modifications to the info by Multiplication and Addition
3) LSTM flow information in form of cell states
4) Deal with both STM and LTM for making calculations simple and effective.

(A) Forget Gate: Decides which info needs attention and which can be ignored.

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

f_t = forget gate at t
W_f = wt matrix between forget gate & input Gate
h_{t-1} = previous hidden state
x_t = input
b_f = connection bias

(B) Input Gate: perform 2 operations to update cell states

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

i_t = input gate at t
b_i = bias vector at I
W_i = weight Matrix (input gate & OP gate)
W_C = weight Matrix (cell state info & Network op)
$\tilde{C}_t$ = value generated by tanh

(C) **Cell state**: To decide new cell state

[forget gate & info | Input gate & info | previous cell state]

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

C_t = cell state information
f_t = forget gate at t
i_t = input gate at t
C_{t-1} = previous cell state info.

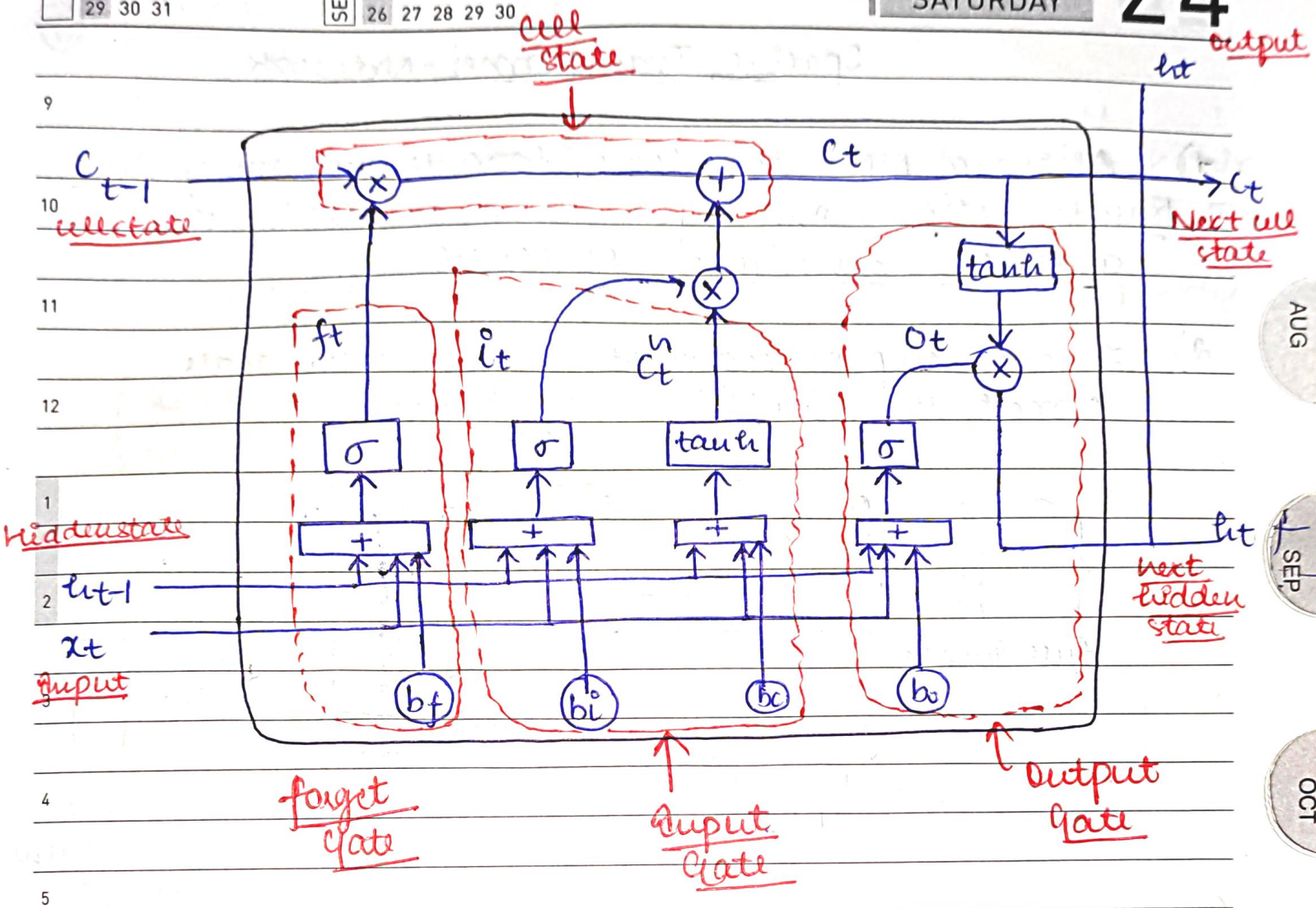
(D) **Output gate**: Determine value of next hidden state (contain info on previous inputs)

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

$$h_t = O_t * \tanh(C_t)$$

[current state & previous hidden state | new cell state.]

O_t = output gate at t
W_o = weight matrix - output gate
b_o = bias vector
h_t = LSTM output



Advantages:

1) Solves the problem of vanishing gradient.

Disadvantages:

1) Takes longer to train
2) Require More Memory
3) Easy to overfit
4) Sensitive to Random weight initialization
5) Difficult to apply dropout Algorithm

Applications:

↳ Language Translation
↳ Speech & Handwriting Recognition
↳ Music Generating
↳ Image processing
↳ Language Modelling

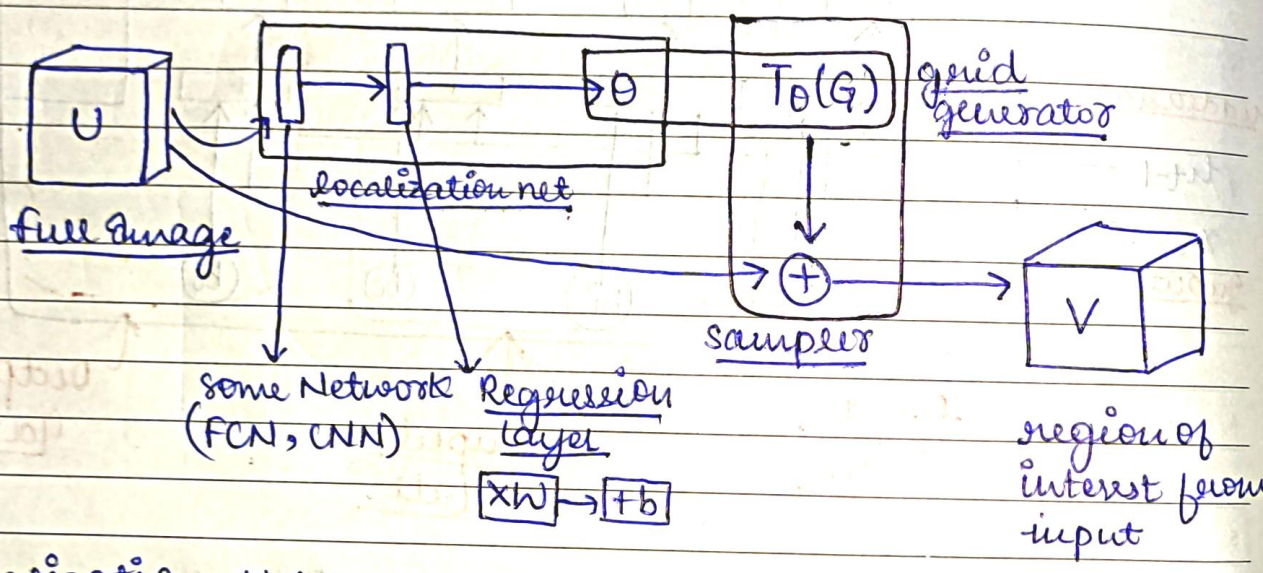
SUNDAY 25

AUG SEP OCT NOV DEC

Spatial Transformer Networks

1) Allow a NN to learn how to perform spatial transformation on input image to enhance geometric invariance of Model.

2) It can crop of region of intrsection, scale correct the orientation of image.



(A) Localization Net

1) Regular Neural Network that regress transformation parameters.

2) Transformation is never learned expicitly, instead Network learns automatically the spatial transform that enhances global accuracy.

transformation parameters

AUGUST '21

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

SEPTEMBER '21

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

(B) **Grid Generator:**

Transformation parameters predicted by LN are used in form of Affine Transformation Matrix for each image in batch.

(Affine transformation preserves points, st lines & planes)

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = T_\theta(G) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

source — target

θ_{11}, θ_{12}: angle by which img has to be rotated

θ_{13}, θ_{23}: translation along width & height

Thus, we obtain sampling grid

(C) **Sampler:** Iterate over entries of sampling grid and extract corresponding pixel value from IP map using Bilinear Interpolation.

1. find 4 neighbouring points
2. for each neighbouring point calculate weight
3. take weighted avg to produce output.

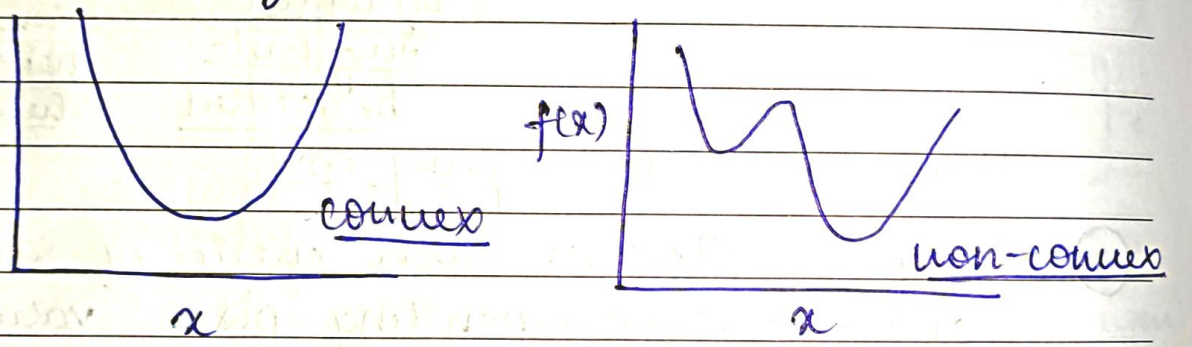
$$w_{ij} = U_{yx} \cdot dy \cdot dx + U_{y\bar{x}} \cdot dy\,(1-dx)$$
$$+ U_{\bar{y}x} \cdot (1-dy)\,dx + U_{\bar{y}\bar{x}}(1-dy)(1-dx)$$

Non convex optimization

1) Potentially many local Minima
2) saddle points

(A point on the surface of graph of a function where slopes in orthogonal directions are all zero, but which is not the local extremum of a function).

3) Very flat region (plateaus)
4) Widely varying Curvature



Gradient Descent optimization Algo.

(i) Momentum
(ii) Nestov Accelerated Gradient
(iii) Adagrad
(iv) Adadelta
(v) RMS Prop
(vi) Adam
(vii) Adam Extension

Momentum

SGD has trouble navigating Ravines, momentum helps accelerate.

Adds a fraction γ of updated vector of past step v_{t-1} to current vector v_t

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta_t} J(\theta_t)$$

$$\theta = \theta - v_t$$

↳ Reduce updates for dimension whose gradient changes directions.

↳ Increase updates for dimension whose gradients points in the same direction.

AUG

SEP

OCT

Adagrad

Adagrad adapts the learning rate to the parameters

SGD update $\theta_{t+1} = \theta_t - \eta g_t$

$g_t = \nabla_{\theta_t} J(\theta_t)$

Adagrad divides learning rate by square root of sum of squares of historic gradients.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

$G_t \in R^{d \times d}$ diagonal matrix where each diagonal ~~matrix~~ element i, is the sum of sq. of gradients wrt θ_i

$$\theta_{t+1} = \theta_t - \eta g_t = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$

Square root of sum of squares of historic gradients

RMS prop

↳ Divides learning rate by running average of squared gradients

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

γ = decay parameter, set 0.9

η = learning rate, 0.001

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

AdaM (Adaptive Momentum)

Stores running average of past squared gradients v_t like adadelta & RMSprop.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

m_t = first moment (mean) of gradients
v_t = second moment (uncentered variance) of gradients
β_1, β_2 = decay rates.

$$\hat{m_t} = \frac{m_t}{1-\beta_1^t}$$

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

Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t} + \epsilon}} \hat{m_t}$$

$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ $\quad \hat{m_t} = \frac{m_t}{1-\beta_1^t}$ $\quad \hat{v_t} = \frac{v_t}{1-\beta_2^t}$

$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ $\quad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t} + \epsilon}} \hat{m_t}$

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

$$g_t = \nabla_{\theta_t} J(\theta_t)$$

$$\Delta\theta_t = -\eta g_t$$

$$\boxed{\theta_t = \theta_{t-1} + \Delta\theta_t}$$

Momentum

$$\Delta\theta_t = -\gamma v_{t-1} - \eta g_t$$

Adagrad

$$\Delta\theta_t = \frac{-\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

$$G_t = \sum_{i=0}^{t} \text{diag}(\nabla_{\theta_i} J(\theta_i))^2$$

Adadelta

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

RMSprop

$$\Delta\theta_t = \frac{-\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Adam

$$\Delta\theta_t = \frac{-\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Anniversaries

Birthdays

$$G_t = \sum_{i=0}^{t} \text{diag}(g_i)^2$$

Weight Initialization

unlike convex optimization, it matters where you start.

- Initialize with zero → get stuck at big saddle pt.
- Initialize with constant value → difficult to break symmetry.
- Initialize with large values → off on the great plateaus, small gradient, slow converge.

> Initialize the set of networks with small Random values.

eg. zero mean Gaussian noise with constant variance

Natural Language processing

1) Sentence segmentation
Breaking piece of ~~sentence~~ text in to various sentenses

2) Word tokenization:
Breaking sentence into individual words called tokens.

3) Predicting part of speech for each token:
predicting whether the word is noun, verb, adverb adjective, pronoun, preposition etc.

4) Lemmatization:
feeding the Model with root word.

5) Identifying stop words:
Most frequently occuring words are stopwords.

6) Dependency passing:
finding out relation between words in a sentence.

7) finding noun phrases:
grouping words that represent same idea

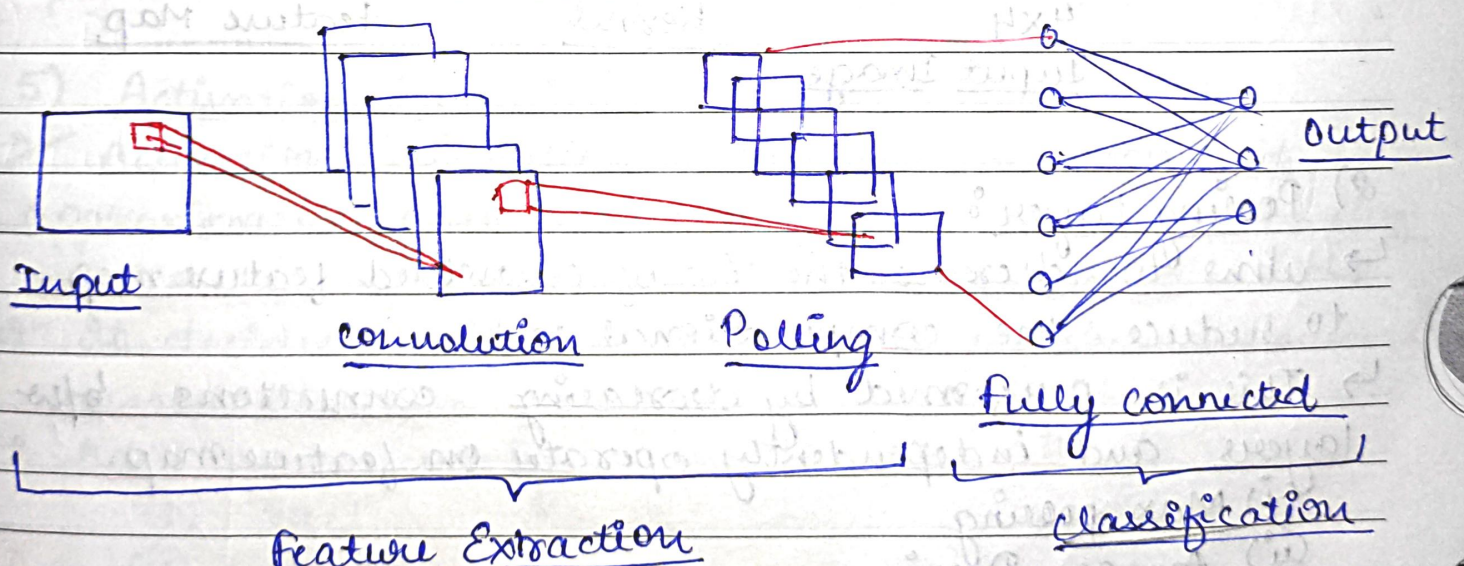
8) Name Entity Recognition:
NER maps words with the real world places.

Convolutional Neural Network

CNN are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analysing visual images.

CNN Architecture:

1) A convolution tool that separates and identifies the various features of image for analysis in a process called Feature Extraction.
2) A fully connected layer that utilizes the O/P from Convolution process and predicts the class of image based on features extracted in prev. stages.



$\left[\frac{i - k + 2p}{s}\right] + 1$ $\left[\frac{i - k + 2p}{s}\right] + 1$ $4 - 2$

21' APRIL

06 DAY 096-269 | Wk 15 TUESDAY

MARCH '21

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

APRIL '21

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

1) Convolution Layer :

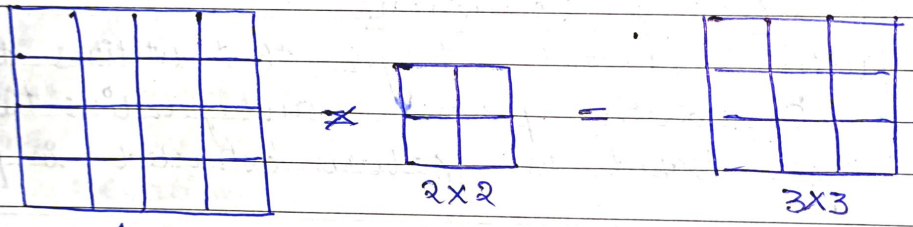
↳ used to extract various features from I/P image.

↳ Mathematical operation of convolution is applied b/w the input image and filter of particular size MXM called kernel filter.

↳ By sliding kernel over input image, dot product is taken b/w the filter and parts of image wrt to size of filter.

↳ The output is feature Map which gives info about ing such as corners, edges etc.

eg.



4X4 Input Image — Kernel — Feature Map

2) Pooling Layer :

↳ aims to decrease the size of convolved feature map to reduce the computational cost.

↳ This is performed by decreasing connections b/w layers and it independently operates on feature map.

(i) Max pooling

(ii) Average Pooling

eg Max pooling :

29	2	6	47
9	6	13	2
0	7	12	4
0	21	33	3

4X4

filter (2X2) = stride(2,2)

29	47
21	33

3) Fully Connected layer:

↳ Consist of the wts. and biases along with the neurons and is used to connects the neurons b/w 2 different layers

↳ These layers are usually placed before output layers and forms the last few layers of CNN Arch

↳ It work on the flattened image fed to it from previous layers.

4) Dropout layer:

↳ When all the features are connected to FC layer, it can cause overfitting of training dataset.

↳ To overcome this, Dropout layer is used wherein few neurons are dropped from NN during training process resulting in reduced size of Model.

5) Activation layer:

↳ Activation functions are used to learn and approximate any kind of continuous and complex relationship b/w the variables of Network.

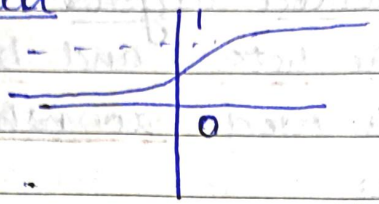
↳ It decides which info. of model should fire in forward direction.

↳ Adds Non-linearity to Network.

(i) ReLU
(ii) Softmax
(iii) tanh
(iv) Signoid function.

(i) sigmoid function

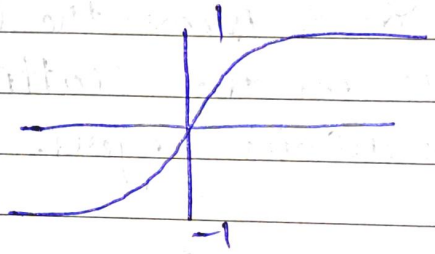
$$f(x) = \frac{1}{1+e^{-x}}$$



(0,1)

(ii) Tanh function

$$f(x) = \frac{2}{(1+e^{-2x})^{-1}}$$

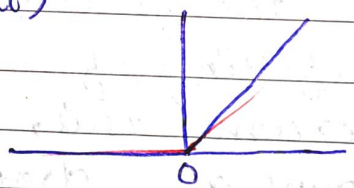


(-1,1)

(iii) ReLU (Rectified linear unit)

used in hidden layer only

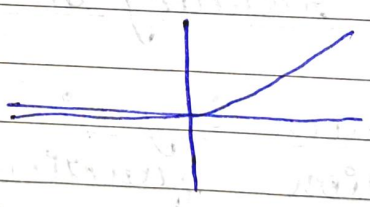
$f(x) = \max(0,x)$



$x<0 \Rightarrow 0$

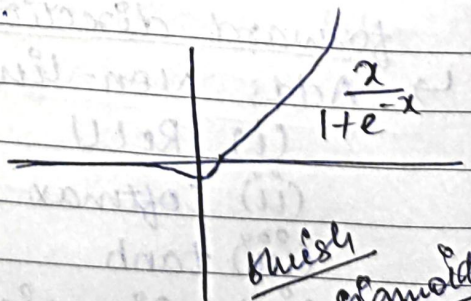
(iv) Leaky ReLU

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



(v) softmax function : combination of multiple sigmoids.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1 \dots K.$$



Swish fn = x sigmoid(x)

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad j = 1 \dots K$$

Delta Rule:

↳ Delta Rule is a gradient descent learning rule for updating weights of I/P to Artificial Neurons in a single layer Neural Network.

↳ Special case for More general Backpropagation Algorithm.

$$\Delta w_i = -n \sum_{d \in D} (t_d - O_d) x_{id}$$

D = set of training examples
η = learning Rate
t_d = target output for training example d
O_d = Output of linear unit for training example d.

Delta Rule is Gradient Descent learning rule for updating weights of Input to Artificial Neural in a single layer Neural Network.
It is a special case for More generalized Backpropagation Algorithm

$$\Delta w_i = -\eta \sum_{d \in D} (t_d - O_d) x_{id}$$

D = set of training Data
η = learning Rate
O_d = target output for training example d
t_d = Output of linear unit of training example d

Delta Rule is derived by attempting to minimize the error in O/P of neural Network using gradient descent

(1) $$E[\vec{w}] = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

The direction of steepest can be found by computing derivative of E wrt each component of vector $\vec{w}$
The vector derivative is called gradient of E wrt $\vec{w}$

(2) $$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

The gradient descent specifies the direction of steepest increase of E, the training rule for gradient descent is

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$$

(3) where, $\Delta \vec{w} = -\eta \nabla E(\vec{w})$

$\vec{w_i} \leftarrow \vec{w_i} + \Delta \vec{w_i}$

$\Rightarrow \Delta \vec{w_i} = -\eta \frac{\partial E}{\partial w_i}$ } component form ——— (1)

(4) Calculating gradient at each step. The vector of $\frac{\partial E}{\partial w_i}$ derivatives that form gradient can be obtained by differentiating E

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d 2 (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$= \frac{1}{2} \sum_d 2 (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x_d})$$

$$\boxed{\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-x_{i,d})} \quad \text{——— (2)}$$

from (1) & (2)

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\boxed{\Delta w_i = -\eta \sum_d (t_d - o_d) x_{id}}$$

21' APRIL

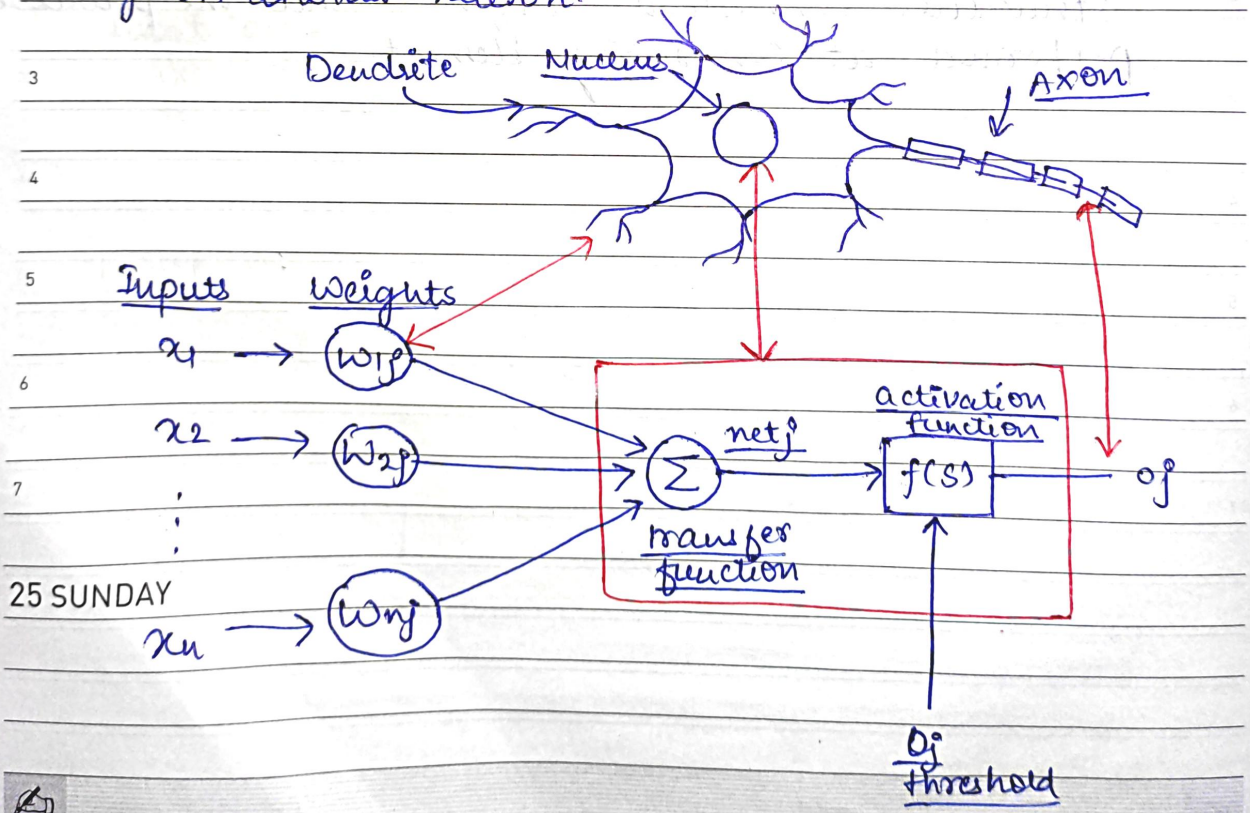
24 DAY 114-251 | Wk 17 SATURDAY

MARCH '21	M	T	W	T	F	S	S
	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31				

APRIL '21	M	T	W	T	F	S	S
				1	2	3	4
	5	6	7	8	9	10	11
	12	13	14	15	16	17	18
	19	20	21	22	23	24	25
	26	27	28	29	30		

Artificial Neuron is information processing system that has certain characteristics in common with biological Nets.

↳ processing elements receives many signals.
↳ Signals can be modified by a weight at receiving synapse.
↳ The processing element sum the weighted input
↳ under appropriate condition, the neuron transmits the output signal.
↳ The output from one particular neuron may go in another neuron.



25 SUNDAY

There are two ways of exerting one's Strength: one is pushing down, the other is pulling up

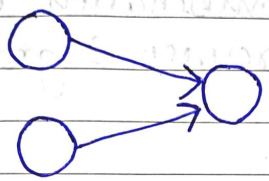
MAY '21

S	M	T	W	T	F	S
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

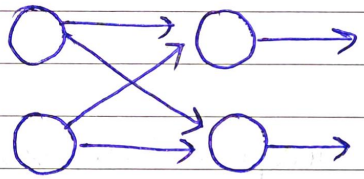
JUNE '21

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

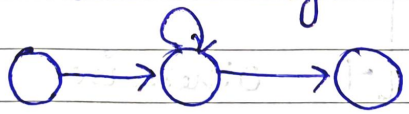
Single layer: One input layer, one output layer of processing unit No feedback connection.
eg. perceptron.



Multi layer: One input layer, one output layer, and one or more hidden layers of processing unit.
No feedback connection.
eg. Multilayer perceptron.

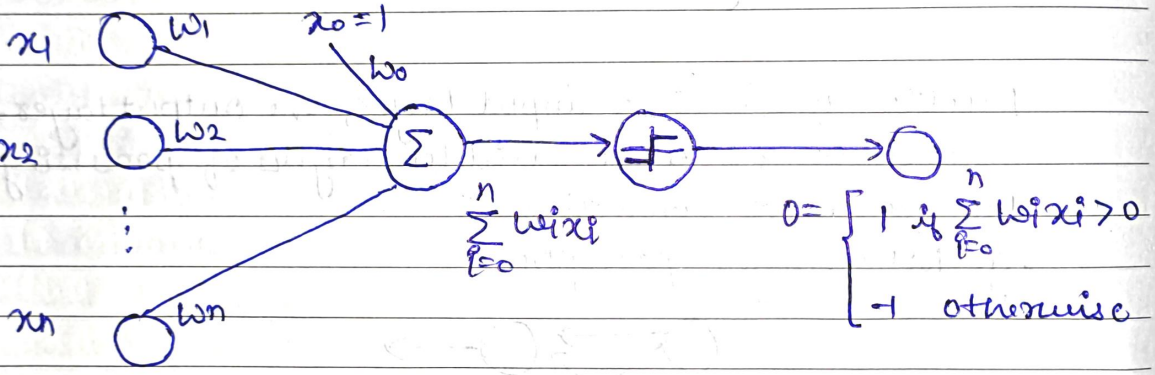


Recurrent: A network with atleast one feedback connection. It may or may not have hidden layers. units. RNN have memory.
eg. RNN



Perceptron : Single layer Neural Network.

Perceptron takes a vector of real valued inputs and calculates a linear combination of these I/P then output 1 if the result is greater than some threshold otherwise -1.



Given input x, Output $O(x_1, x_n)$ generated by perceptron.

$$O(x_1, \ldots x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots w_n x_n > 0 \\ -1 & \text{Otherwise} \end{cases}$$

$$O(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

w_i = real valued constant, that determines the contribution of input x_i to perceptron output.

w_0 is threshold that weighted combination of inputs must surpass in order for perceptron to O/P 1.

Learning a perceptron involves choosing weights $w_0, w_1, \ldots w_n$

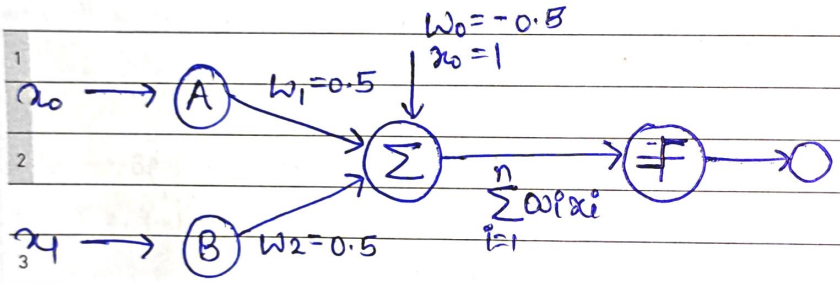
∴ The space H of candidate hypotheses considered in perceptron learning is set of all real-valued weight vectors.

$$H = \{\vec{w} \mid \vec{w} \in R^{n+1}\}$$

Singe Perceptron can represent Boolean function

AND

A	B	A ∧ B
0	0	0
0	1	0
0	0	0
1	1	1



$$O(x_1, x_2) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 > 0 \\ -1 & \text{otherwise} \end{cases}$$

(i) If $A(x_0) = 0$ & $B(x_1) = 0$

$-0.8 + (0.5)(0) + (0.5)(0) = 0$ Output = 0

(ii) If $x_0 = 0$ $x_1 = 1$ $\Rightarrow -0.8 + (0.5) * 1 + (0.5) * 0 = -0.3 < 0$

Output = 0

(iii) If $x_0 = 1$ $x_1 = 0$ $\Rightarrow -0.8 + (0.5) 1 + (0.5) * 0 = -0.3 < 0$

Output = 0

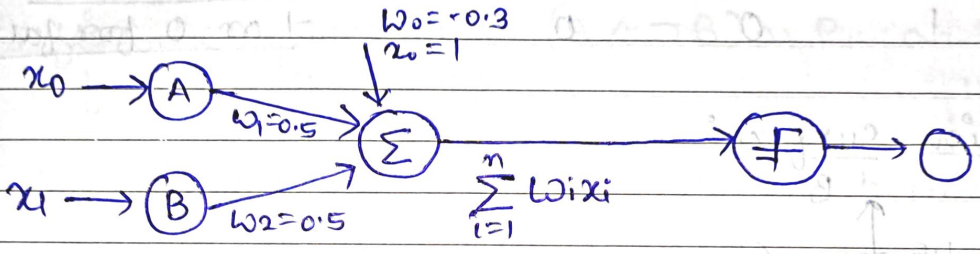
(iv) If $x_0 = x_1 = 1$ $\Rightarrow -0.8 + 0.5 \times 1 + 0.5 \times 1 = 0.2 > 0$

Output = 1

OR

x_0	x_1	$x_0 \vee x_1$
0	0	0
0	1	1
1	0	1
1	1	1

Let $w_0 = -0.3$
$w_1 = 0.5$
$w_2 = 0.5$



(i) $A=0$ & $B=0$ $\Rightarrow -0.3 + (0.5 \times 0) + (0.5 \times 0)$
$-0.3 < 0$ Output = 0

(ii) $A=0$ & $B=1$ $\Rightarrow -0.3 + (0.5 \times 0) + (0.5 \times 1)$
$= 0.2 > 0$ Output = 1

(iii) $A=1$ & $B=0$ $\Rightarrow -0.3 + (0.5 \times 1) + (0.5 \times 0)$
$= 0.2 > 0$ Output = 1

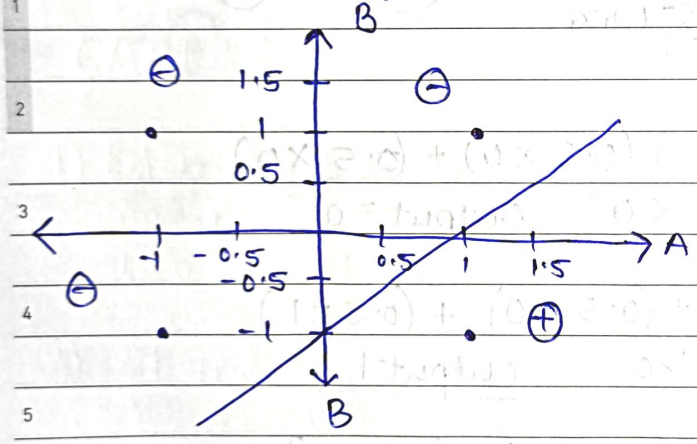
(iv) $A=1$ & $B=1$ $\Rightarrow -0.3 + (0.5 \times 1) + (0.5 \times 1)$
$= 0.7 > 0$ Output = 1

Ques $A \wedge \neg B$

A	B	¬B	A ∧ ¬B
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

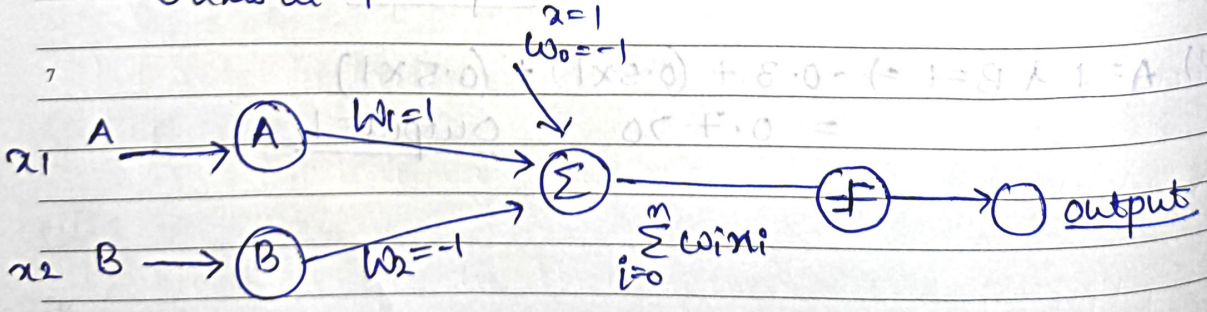
The values of A and B are 1 (true) or -1 or 0 for false

Decision Surface:



Let Weights are
$w_0 = -1$
$w_1 = 1$
$w_2 = -1$

The line crosses A axis at 1
B axis at -1



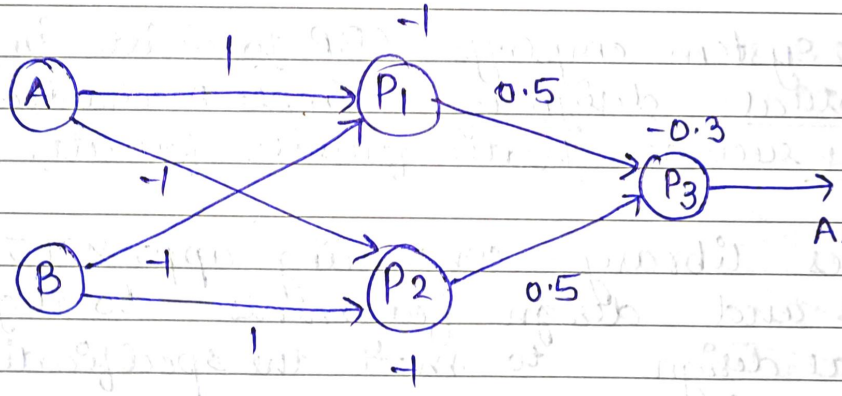
(ii) A XOR B cannot be calculated by single perceptron so we build a 2 layer network of perceptron.

Expressing $A \oplus B$ in terms of logical connections

$$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$$

① Define perceptron for $(A \wedge \neg B)$ as P_1 and perceptron for $(\neg A \wedge B)$ as P_2

② Composing the output of P_1 and P_2 into P_3 that implements $O(P_1) \vee O(P_2)$.



Backpropagation Algorithm

1) The Backpropagation Algo. learns the weights for Multilayer Network, given a network with a fixed set of units and interconnections. It employs gradient descent to attempt to minimize the sq errors b/w the network O/P values & the target values for those outputs.

2) On Backpropagation Algorithm, we consider network with multiple O/P units rather than single units. The sum of errors of all the network O/P units

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{Outputs}} (t_{kd} - o_{kd})^2$$

Outputs = set of O/P units in the network
t_{kd} and o_{kd} = the target and O/P values associated with Kth output unit.
d = training example.

2 phases of computation.

Forward Pass:
1) Run the NN and compute error for each neuron of the output layer.

Backward pass:
start at the O/P layer and pass errors backward through the network, layer by layer, by recursively computing local gradient descent.

21' MARCH

24

DAY 083-282 | Wk 13

WEDNESDAY

FEBRUARY '21							
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

MARCH '21							
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

BACKPROPAGATION (training_examples, η, n_{in}, n_{out}, n_{hidden})

Each training example is pair of form $(\vec{x}, \vec{t})$,
$\vec{x}$ vector of network input values, $\vec{t}$ vector of target network output values.

η = learning rate n_{hidden} = no. of hidden layer units.

n_{in} = no. of network inputs, n_{out} = no. of output units

x_{ji} = input from unit i into unit j

w_{ji} = weight from unit i to j.

1) ↳ Create feed forward Network with n_i, n_{hidden}, n_{out}.
2) ↳ Initialize all Network weights to small random no.
3) ↳ Until termination condition is met, Do
4) ↳ for each $(\vec{x}, \vec{t})$ in training examples, Do
5) Propagate I/P forward through network
6) 1) Input $\vec{x}$, to network and compute O_u of every unit u in the network.
7) Propagate errors backward through network.
8) 2) for each network output unit k calculate error
9) $$\delta_k \leftarrow O_k (1 - O_k)(t_k - O_k)$$
10) 3) for each hidden unit h, calculate error
11) $$\delta_h \leftarrow O_h (1 - O_h) \sum_{k \in outputs} w_{h,k} \delta_k$$ $w_{hk}\delta_k$
12) 4) Update each network weight w_{ji}
$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$
Where, $$\Delta w_{ji} = \eta \delta_j x_{i,j}$$

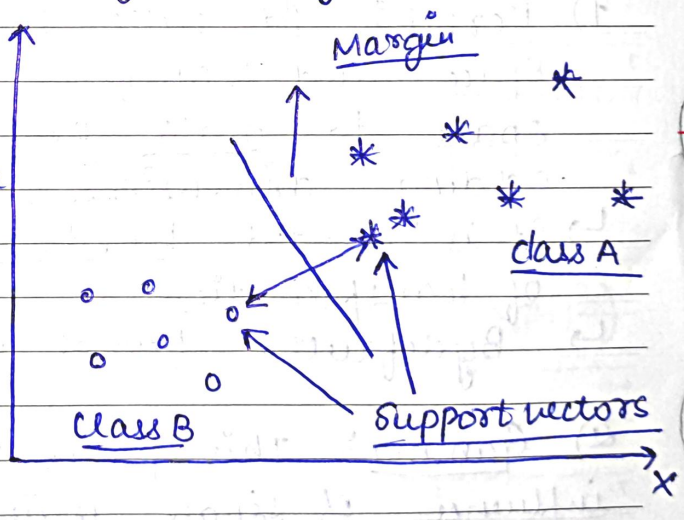
Support Vector Machine

↳ SVM is powerful and flexible supervised ML Algorithm used for classification and regression analysis.

↳ An SVM Model is basically representation of different classes in Hyperplane in Multidimensional Space.

↳ The hyperplane will be generated in iterative manner by SVM so that the error can be minimized.

↳ The Goal of SVM is to divide the dataset into classes to find a maximum marginal Hyperplane (MMH)



↳ Support vectors:
Datapoints that are closest to the hyperplane
Separating line will be defined using these points

↳ Hyperplane:
A decision plane or space which is divided b/w the set of objects of diff. classes.

↳ Margin: The gap b/w 2 lines on the closest datapts. of different classes. It can be calculated as the ⊥ dist. from the line to support vectors.

Linear SVM:

It is used for linearly separable data, which means if a dataset can be classified into 2 classes using single straight line then such data is linearly separable data & classifier is linear SVM.

Non-linear SVM

It is used for Non-linearly separable data, i.e. if a datataset cannot be classified into by using straight line then such data is Non-linear data & classifier is Non-linear SVM.

Parameters used in Support vector classifier:

1) **Kernel:** It takes lower dimensional input space and transform it into higher dimensional space to make problem separable by adding dimensions.
   - ↳ It is selected based on type of data and type of transformation.
   - ↳ By default, Kernel is RBF Kernel.

2) **Gamma:** This parameter determines how far the influence of single training example reaches during transformation, which in turn affects how tightly decision boundaries endup surrounding points in input space.
   - ↳ 'γ' increases, model gets Overfits, 'γ' decreases model gets underfit.

3) C parameter: This parameter controls the amount of regularization applied on data.
↳ Large value of C = Low regularization = cause overfitting.
↳ Lower value of C = High Regularization = cause underfitting (may lead to lower accuracy).

Types of Kernels:

1) Linear Kernel: $K(x, x_i) = \text{sum}(x * x_i)$.

2) Polynomial Kernel:
$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$$
d = degree of polynomial
popular in Image processing.

3) Gaussian Kernel: $K(x, y) = \exp\left(\frac{\|x-y\|^2}{2\sigma^2}\right)$

4) Gaussian RBF: $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$
$1 > \gamma > 0$ good value 0.1

5) Laplace RBF: $K(x, y) = \exp\left(-\frac{\|x-y\|}{\sigma}\right)$

NN
6) Hyperbolic Tangent Kernel: $K(x_i, x_j) = \tanh(K x_i \cdot x_j + c)$
$K > 0$ (for some) & $c < 0$
7) Sigmoid Kernel: $K(x, y) = \tanh(\alpha x^T y + c)$

8) ANOVA radial basis Kernel:
$$K(x, y) = \sum_{k=1}^{n} \exp\left(-\sigma (x^k - y^k)^2\right)^d$$
regression

Advantages

1) SVM is more effective in high dimensional space.

2) SVM can be used for linearly separable and non linearly separable data

3) SVM is relatively memory efficient.

4) Guaranteed optimality
Owing to nature of convex Optimization, the soln will always be Global Min.

5) SVM can carryout feature mapping using simple dot product using kernel Trick.

Disadvantages

1) Not suitable for large datasets.

2) SVM doesn't perform well when data set has noise target classes are overlapping

3) SVM doesnot give best performance for handling text structures.

4) The choice of kernel is perhaps the biggest limitation of SVM.

5) In case where no. of features for each datapt exceeds the no. of training data samples SVM will under perform.

14 SUNDAY

(5) (1) More efficient in HDS
(2) Memory efficient
(3) Guaranteed optimality
(4) feature mapping using simple dot product.

Properties of SVM

1) Flexibility in choosing similarity function.

2) Sparseness of solution when dealing with large datasets. only support vectors are used to specify the separating hyperplanes.

3) Ability to handle large feature space: Complexity doesnot depends on dimensionality of feature space.

4) Overfitting can be controlled by soft margin approach.

5) A simple Convex optimization problem which is guaranteed to converge to single global soln.

6) Feature selection.

REGRESSION

Regression Analysis or Regression is a statistical method and a predictive Modelling Technique which investigates the relation between one dependent and a series of other independent variables.

eg. Relationship b/w rash driving & no. of road accidents by a driver is best studied through regression

Advantages:

1) It indicates the significant relationship b/w dependent and independent variables

2) It indicates the strength of impact of Multiple Independent variables on dependent variables.

21' JANUARY

22 DAY 022-343 | Wk 04
FRIDAY

DECEMBER '20

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JANUARY '21

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Linear Regression A supervised ML algorithm where the predicted output is continuous and has a constant slope.

Simple Regression
$y = mx + b$

Multivariable Regression
$f(x, y, z) = w_1 x + w_2 y + w_3 z$

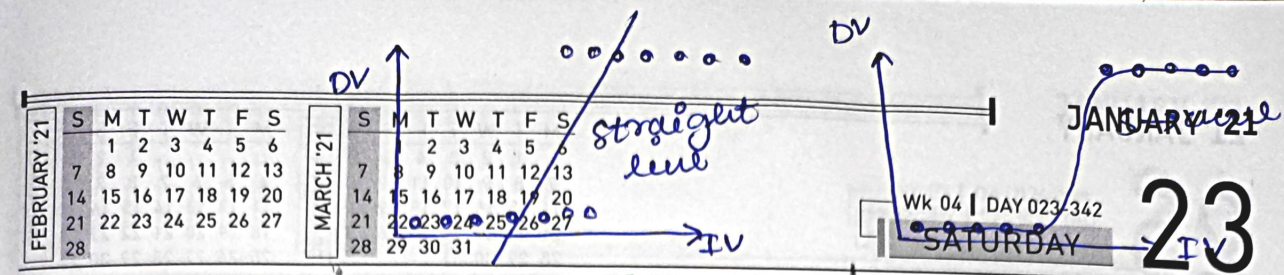
Logistic Regression: A supervised learning classification algorithm used to predict the probability of target variable.

Binary (Binomial) Regression
O/P : 1/0

Multinomial Regression
O/P 'Type A'
'Type B'
'Type C'

Ordinal Regression
Types having Quantitative significance
O/P 'Poor' / 'Good'

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1 x + b_2 x_2 + \_\_\_ + b_n x_n$$



Parameter	Linear Regression	Logistic Regression
1) Model	Supervised Regression Model	Supervised classification Model.
2) Used	To predict values with in a constant range	To predict the probability of a target variable.
3) Activation Function	No activation function is used	Activation function is used to convert linear regression to logistic regression
4) Threshold value.	Threshold value is added	No threshold is needed
5) Purpose	To find best-fitted line	To fit the line values to sigmoid curve.
6) Loss function Method	Mean square Error Method	Maximum likelihood Estimation.
7) Output	Continuous Values	Categorical values.
8) Collinearity	collinearity b/w the Independent variables	No collinearity b/w independent variables
9) MUATPLOC	Gaussian / Normal distribution of the dependent var.	Binomial Distribution of dependent var.

SUNDAY 24