18/04/23

MODULE - 1

*) EIGEN DECOMPOSITION - EXAMPLE

1) $A = \begin{bmatrix} 4 & 3 \\ 2 & -1 \end{bmatrix}$

∴ Characteristic Equation :- $|A - \lambda I|$

$\Rightarrow$ Let $|A - \lambda I| = 0$

$\Rightarrow \left| \begin{bmatrix} 4 & 3 \\ 2 & -1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$

$\Rightarrow \left| \begin{bmatrix} 4-\lambda & 3 \\ 2 & -1-\lambda \end{bmatrix} \right| = 0$

$\Rightarrow (4-\lambda)(-1-\lambda) - 6 = 0$

$\Rightarrow -(4-\lambda)(1+\lambda) = 6$

$\Rightarrow -(4 + 4\lambda - \lambda - \lambda^2) = 6$

$\Rightarrow -(4 + 3\lambda - \lambda^2) = 6$

$\Rightarrow \lambda^2 - 3\lambda - 4 = 6$

$\Rightarrow \lambda^2 - 3\lambda - 10 = 0$

$\Rightarrow \lambda^2 - 5\lambda + 2\lambda - 10 = 0 \Rightarrow \lambda(\lambda - 5) + 2(\lambda - 5) = 0$

$\therefore \boxed{\lambda = 5, -2}$

$\therefore \lambda = -2$

$$\Longrightarrow \begin{bmatrix} \lambda-(-2) & 3 \\ 2 & -1-2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} 4-(-2) & 3 \\ 2 & -1-(-2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} 6 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\Longrightarrow 6x_1 + 3x_2 = 0 \quad \Rightarrow -2x_1 = x_2 \quad -(1)$

$2x_1 + x_2 = 0 \Rightarrow -2x_1 = x_2 \quad -(2)$

$$\therefore V_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ -2x_1 \end{bmatrix} = \underline{\begin{bmatrix} -1 \\ 2 \end{bmatrix}}$$

$\therefore \lambda = 5$

$$\Longrightarrow V_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underline{\begin{bmatrix} 3 \\ 1 \end{bmatrix}}$$

$$V = (V_1 \quad V_2) = \begin{bmatrix} 3 & -1 \\ 1 & 2 \end{bmatrix}$$

$$\therefore V^{-1} = \frac{1}{7}\begin{bmatrix} 2 & 1 \\ -1 & 3 \end{bmatrix}$$

Ans Eigen decomposition equation:- $A = VDV^{-1}$

$$= V\,\text{diag}(\lambda)\,V^{-1}$$

$$\therefore A = \begin{bmatrix} 3 & -1 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 5 & 0 \\ 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} \frac{2}{7} & 1/7 \\ -1/7 & 3/7 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 3 \\ 2 & -1 \end{bmatrix}$$

Ex 2) $A = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix}$

$\therefore$ Characteristic equation: $|A - \lambda I| = 0$

$$\Longrightarrow \left| \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$$

$$\Longrightarrow \left| \begin{pmatrix} 2-\lambda & 3-0 \\ 2-0 & 1-\lambda \end{pmatrix} \right| = 0$$

$$\Longrightarrow (2-\lambda)(1-\lambda) - 6 = 0$$

$$\Longrightarrow (2 - 2\lambda - \lambda + \lambda^2) = 6$$

$$\Longrightarrow \lambda^2 - 3\lambda + 2 = 6 \Longrightarrow \lambda^2 - 3\lambda - 4 = 0$$

$\Rightarrow (\lambda^2 - 4\lambda + \lambda - 4) = 0$

$\Rightarrow (\lambda(\lambda - 4) + 1(\lambda - 4)) = 0$

$\Rightarrow (\lambda - 4)(\lambda + 1) = 0$

$\Rightarrow \boxed{\lambda = 4, -1}$

$\therefore \lambda_1 = 4$

$\Rightarrow \begin{bmatrix} 2-\lambda & 3 \\ 2 & 1-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$

$\Rightarrow \begin{bmatrix} 2-4 & 3 \\ 2 & 1-4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$

$\Rightarrow \begin{bmatrix} -2 & 3 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$

$\Rightarrow -2x_1 + 3x_2 = 0 \quad \therefore \frac{x_1}{x_2} = \frac{3}{2}$

$2x_1 - 3x_2 = 0$

$\therefore v_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$\therefore \lambda_2 = -1$

$$\Rightarrow \begin{bmatrix} 2-\lambda & 3 \\ 2 & 1-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} 2-(-1) & 3 \\ 2 & 1-(-1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} 3 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\Rightarrow 3x_1 + 3x_2 = 0 \qquad 3x_1 = -3x_2$$
$$2x_1 + 2x_2 = 0 \qquad \therefore \frac{x_1}{x_2} = \frac{-3}{3} = \frac{-1}{1}$$

$$v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\therefore V = \begin{bmatrix} v_1 & v_2 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ 2 & 1 \end{bmatrix}$$

$$\therefore V^{-1} = \frac{1}{|V|}[V] = \frac{1}{5}\begin{bmatrix} 3 & -1 \\ 2 & 1 \end{bmatrix}$$

Ans $A = VDV^{-1}$

$$= \begin{bmatrix} 3 & -1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 3/5 & -1/5 \\ 2/5 & 1/5 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix}$$

Fun 4
PAGE NO.:
DATE:

*) Gradient - descent method

- For functions with multiple inputs, we must make use of the concept of partial derivatives.
- The partial derivative $\frac{\partial}{\partial x_i} f(x)$ measures how f changes as only the variable x_i increases at point x.
- The gradient generalizes the notion of derivative to the case where the derivative is with respect to a vector, denoted by $\nabla_x f(x)$.
- In some words, the directional derivative of the function $f(x+\alpha u)$ w.r.t α, evaluated at $\alpha = 0$

$$\boxed{\frac{\partial}{\partial \alpha} f(x+\alpha u) = u^T \nabla_x f(x)}$$

- To minimize f, we will like to find the direction in which f decreases the fastest.
- We can also minimize f using the directional derivative.

- We can do this by:-

$$\min_{u,\, u^T u = 1} u^T \nabla_x f(x) = \min_{u,\, u^T u = 1} \|u\|_2 \, \|\nabla_x f(x)\|_2 \cos\theta$$

where:- θ is the angle b/w u and the gradient

- This is minimized when u points in the opp. direction as the gradient.
- This is also known as the method of steepest descent or gradient descent.
- Steepest descent proposes a new point:-

$$x' = x - \epsilon \nabla_x f(x)$$

where ϵ is the learning rate

- Although, thus gradient descent is limited to optimization in continuous spaces, the general concept of making small moves towards better configurations to discrete spaces.

*) Overfitting Vs Underfitting

- Overfitting and Underfitting are common challenges in deep learning, where a neural network may not generalize well to unseen data.
- Let's understand these concepts in more detail:-

(i) Overfitting: occurs when a neural network learns to perform exceptionally well on the training data but fails to generalize to new, unseen data. In other words, the model memorizes the training data instead of learning the underlying patterns. This can be mitigated by:-

(a) Regularization - adds penalty terms to the model's loss function.

(b) Data augmentation - artificially create diversity of training data & reduce overfit

(c) Early stopping - involves monitoring the model performance on the validation set during training

(ii) Underfitting: occurs when a neural network is too simple and fails to capture the underlying patterns in the training data. The model may have high bias & low variance, resulting in poor performance on both the training & new data.

Underfitting can be addressed by:-

(a) Increase model complexity:- Use deeper neural network to help model learn more complex patterns from data. Insufficient training data may lead to underfitting.

(b) Adjusting hyperparameters:- Hyperparameters such as learning rate etc affect the model's performance; experimenting with these hyperparameters may help in reduce underfitting.

- Regularization techniques, data augmentation, early stopping, adjusting hyperparameters are common strategies to mitigate overfitting & underfitting in deep learning.

*) Optimization in deep learning

- This refers to the process of finding the best set of parameters or weights for a neural network model in order to minimize the error using during training.
- Optimization is critical in deep learning as it determines how well the model can learn from the data and make accurate predictions.
- There are several optimization techniques commonly used in deep learning, including:-

(a) Gradient Descent: widely used optimization algorithm in deep learning; involves computing the gradient of the loss function w.r.t the model parameters & updating them.

(b) Weighted Regularization: Used to prevent overfitting by adding penalties to the weights of the model, encouraging sparsity.

(c) Batch Normalization:- Technique used to normalize the inputs to each layer of neural network during training; accelarate training & improve model stability.

(d) Hyperparameter Tuning :- Parameters that are not learned during training but affect the behaviour of the optimization algorithm.

- Optimization is a critical step in training deep learning models.
- It determines how well the model learns from the data & generalizes the unseen data.
- Choosing right optimization techniques & hyperparameter values impact the performance & convergence speed of deep learning models.

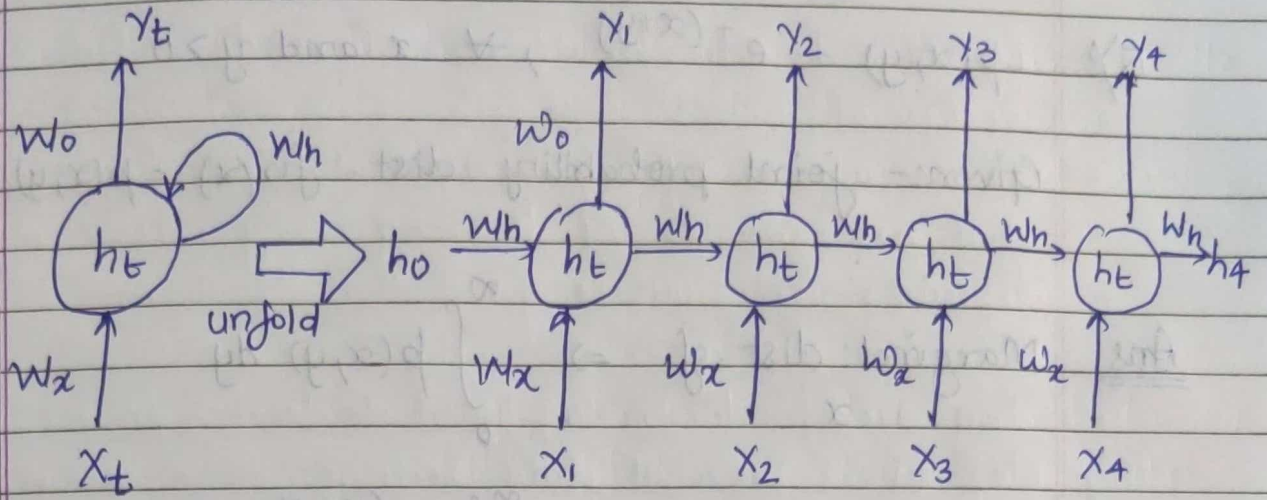
Fun 4
PAGE NO.:
DATE:

*) Recurrent Neural Networks - Architecture & Theory

- Recurrent Neural Networks or RNNs, a very important variant of neural networks heavily used in Natural Language processing.
- Their a class of neural networks that allow previous outputs to be used as inputs while having hidden states.
- RNN has a concept of "memory" which remembers all information about what has been calculated till time step t.
- RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.
- A recurrent neural network can thus, be thought of as multiple copies of a feedforward neural network, each passing a message to a successor.
- The main feature of an RNN is its hidden state, which capture information about a sequence

Fun 4
PAGE NO.:
DATE:

- Below, is the true unfolding of an RNN architecture:-



where:-

h_t - Current state

h_{t-1} - previous state

X_t - input at time step 5 't'

W_h - weight of the recurrent neuron

W_x - weight at input neuron

Y_t - Output at time step 't'

∴ So, from the above figure, we can write below equations:-

$$h_t = f(W_h^T h_{t-1} + W_x^T X_t + b_h) \quad \therefore f = \text{sigmoid, ReLU etc}$$

$$\Rightarrow Y_t = \text{softmax}(W_o^T h_t + b_o)$$

*) Joint probability dist. & marginal dist. - numerical

Q) $p(x,y) = e^{-(x+y)}$, $\forall$ x and $y > 0$

Given:- joint probability dist. - fn(x) = $p(x,y)$

Ans Marginal dist. of x $\Rightarrow \int_0^\infty p(x,y)\,dy$

$$\Rightarrow \int_0^\infty e^{-(x+y)}\,dy$$

$$\Rightarrow \int_0^\infty e^{-x} \cdot e^{-y}\,dy$$

$$\Rightarrow e^{-x} \cdot \int_0^\infty e^{-y}\,dy$$

$$\Rightarrow e^{-x} \cdot \left[\frac{e^{-y}}{-1}\right]_0^\infty$$

$$\Rightarrow e^{-x} \cdot [-e^{-y}]_0^\infty$$

$$\Rightarrow e^{-x} \cdot [-e^{-\infty} - [-e^{-0}]]$$

$$\Rightarrow e^{-x} [1 - 0]$$

$$\Rightarrow \boxed{e^{-x}}$$

Fun 4
PAGE NO.:
DATE:

Marginal dist of $y \Rightarrow \int_0^\infty e^{-x} \cdot e^{-y} dx$

$$\Rightarrow e^{-y} \int_0^\infty e^{-x} dx$$

$$\rightarrow e^{-y} \cdot 1 = \boxed{e^{-y}}$$

Fun 4
PAGE NO.:
DATE:

*) LSTM architecture with mathematical formulation

- LSTM stands for long-short-term networks.
- It is a variety of recurrent neural networks that are capable of learning long-term dependencies, especially in sequence prediction problems.
- They are predominantly used to learn, process and classify sequential data because these networks can learn long-term dependencies between the time steps of data.
- Unlike standard feedforward neural networks, LSTM has feedback connections.
- Such a type of recurrent neural network can process not only single data points, but also entire sequences of data.
- LSTMs are also much less susceptible to the vanishing gradient problem.

Fun 4
PAGE NO.:
DATE:

*) CNN and its components operations

- Convolutional Neural Networks (CNN) are a type of artificial neural network commonly used for image recognition & computer vision tasks.
- They are designed to process data with a grid-like topology, such as images, and consists of several layers of interconnected neurons that perform different operations on the data.
- These are following operations on CNN:-

a) Convolution :- fundamental operation in CNNs where a small matrix called a filter or kernel is applied to an input image or feature map to extract certain features.

b) Filters : Also known as kernels, are small matrices of weights that learn to detect specific features in the input image or feature map, such as edges or corners. They typically learn through a process called Backpropogation during training.

c) Pooling :- Down-sampling operation in which a small window slides over the feature map and outputs the maximum or average value within the window.

d) Stride :- No. of pixels that the filter shifts when it slides over the input image or feature map.

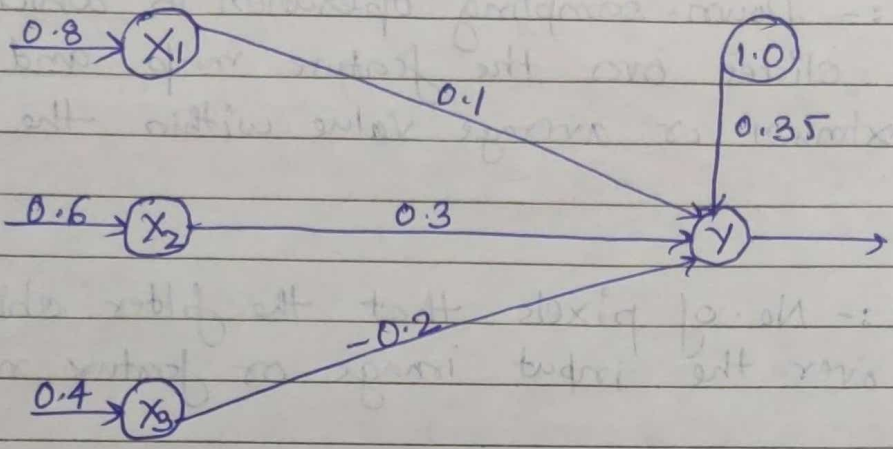
e) Dropout :- Regularization technique used to prevent overfitting of data.

f) Padding :- amount of pixels added to an image when it is being processed by the kernel of a CNN.

- CNNs are used in various fields ; include healthcare, finance, transportation, and security, to name a few.

*) Sigmoidal Activation function - example

Q)



Ans Net input to the output neuron :-

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

$$= 0.35 + (0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times -0.2)$$

$$= 0.35 + 0.18$$

$$= \underline{0.53}$$

(a) Binary sigmoidal function :- $y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}}$

$$= \frac{1}{1+e^{-0.53}}$$

$$= \underline{0.625}$$

(b) Bipolar sigmoidal function :- $y = f(y_{in}) = \frac{2}{1+e^{-y_{in}}} - 1$

$$= 2 \times 0.625 - 1$$

$$= \underline{0.25}$$

*) McCulloh-Pitts Neuron

- This has no particular training algorithm; only analysis is performed
- Firing of the output neuron is based upon the threshold, the activation function here is defined as:-

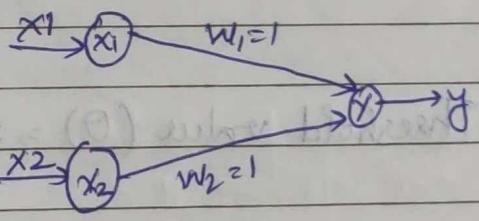
$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

- The Threshold value should satisfy the following condition: $\theta > nw - p$

A) AND gate :-

X1	X2	Y
1	1	1
1	0	0
0	1	0
0	0	0

Let's assume the weights to be:-
$w_1 = 1$ and $w_2 = 1$



$\therefore$ (1,1) :- $y_{in} \Rightarrow 1+1 = 2$

(1,0) :- $y_{in} \Rightarrow 1+0 = 1$

(0,1) :- $y_{in} \Rightarrow 0+1 = 1$

(0,0) :- $y_{in} \Rightarrow 0+0 = 0$

Thus, output of neuron Y can be written:-

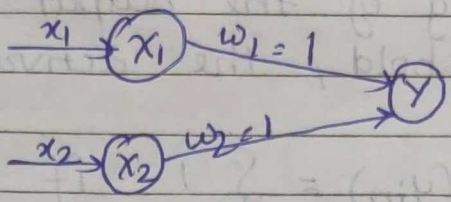
$$Y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$

$\therefore$ This can be obtained as:- $\theta \geq nw - p \Rightarrow n = 2, w = 1$ (inhibitory) & (excitatory) $p = 0$

$\Rightarrow \theta \geq 2 \times 1 - 0 \Rightarrow \boxed{\theta \geq 2}$

B) OR gate :-

x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0



input	Net input
(0,0)	$y_{in} \Rightarrow x_1w_1 + x_2w_2 \Rightarrow 0$
(0,1)	$y_{in} \Rightarrow x_1w_1 + x_2w_2 \Rightarrow 1$
(1,0)	$y_{in} \Rightarrow x_1w_1 + x_2w_2 \Rightarrow 1$
(1,1)	$y_{in} \Rightarrow x_1w_1 + x_2w_2 \Rightarrow 2$

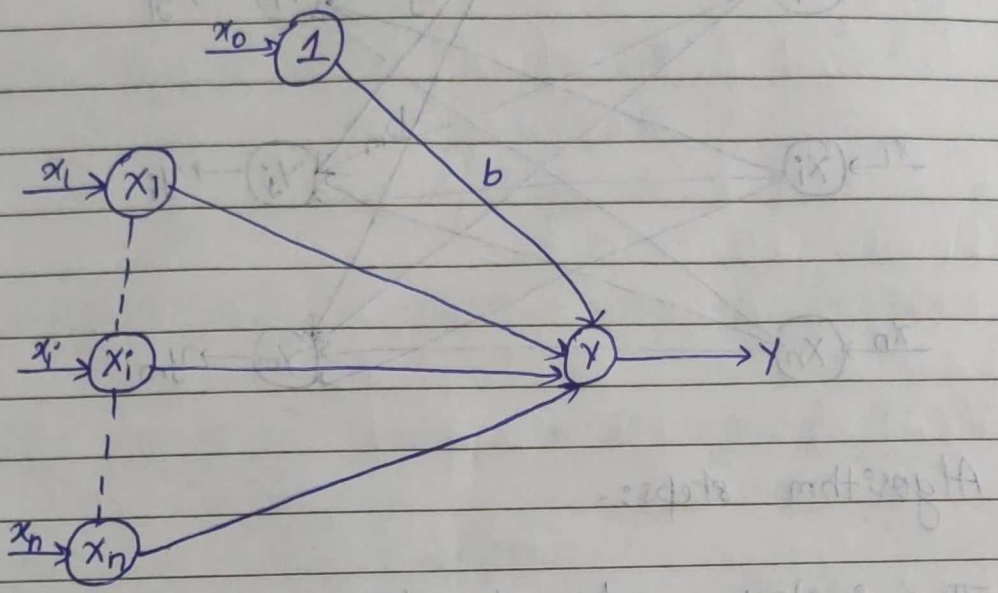
Output Neuron $(Y) = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 1 \\ 0 & \text{if } y_{in} < 1 \end{cases}$

Threshold value $(\theta) = 1$

Fun 4
PAGE NO.:
DATE:

*) Perceptron Training Algorithm :-

1) Single Output Class :-



Algorithm steps :-

Step 1) Initialize the weights & bias. Also, initialize the learning rate α $(0 < \alpha < 1)$

Step 2) Until the final stopping condition is false :-
- for each training pair indicated by s:t
  - Set each input unit $i = 1$ to n: $\boxed{x_i = s_i}$
  - Calculate output of the network

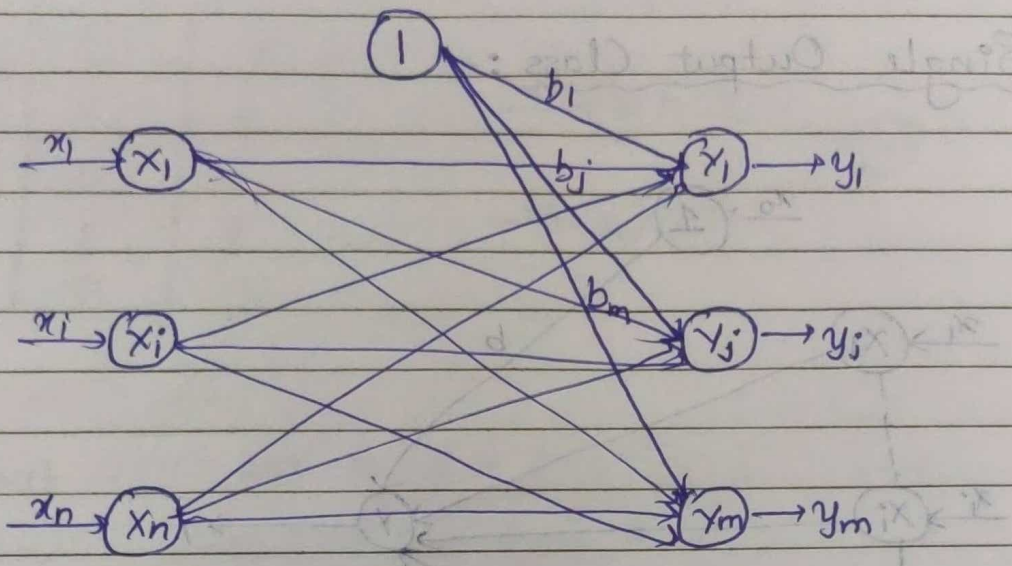
$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

$$\Rightarrow y = f(y_{in})$$

Step 3) Weights & bias adjustment

Step 4) Train the network till there is no weight change.

2) Multiple Output Class :-



Algorithm steps :-

Step 1) The initial weights to be used here are taken from the training algorithms

Step 2) For each input vector X to be classified, perform the following :-

- Calculate the net input of the unit
- Obtain the response of the output unit

$$y_{in} = \sum_{i=1}^{n} x_i w_i$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Fun 4
PAGE NO.:
DATE:

*) Perceptron Learning Rule

- $y_{in} = b + \sum_{i=1}^{n} x_i w_i$

- $y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\alpha < y_{in} < \theta \\ -1 & \text{if } y_{in} < \theta \end{cases}$

- Weights can be updated using the formula:-

a) if $y \neq t \Rightarrow w_i(new) = w_i(old) + \alpha t x_i \rightarrow \Delta w_i$

b) else $\Rightarrow w(new) = w(old)$

A) AND gate:-

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

- initial weights and threshold are set to zero
- Learning rate (α) = 1

*) **Conditional Probability – Example**

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \rightarrow \text{Formulae}$$

$P(B)$ → Already occured event

→ find prob. of A, given B has already occured

$$P(B) = \frac{30}{100} = 0.3$$

$$P(A \cap B) = \frac{20}{100} = 0.2$$

$$P(A/B) = \frac{0.2}{0.3} = 0.67$$

$y_{in} = b + x_1 w_1 + x_2 w_2$

$\Delta b = \alpha t$ $\quad (w_1 = w_2 = 0)$

$\Delta w_1 = \alpha t x_1$; $\Delta w_2 = \alpha t x_2$

$\Delta b = \alpha t$

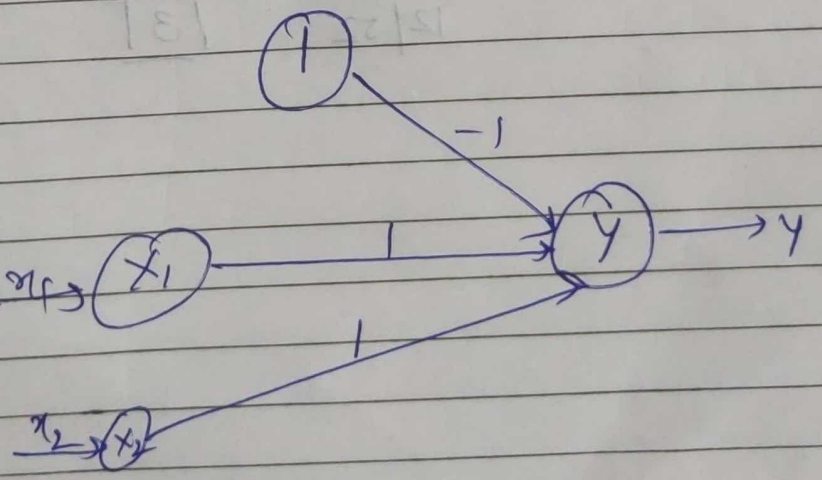
$\alpha = 1$

∴

	Input			Target	Net input	Weight changes			Weights		
	x_1	x_2	b	t	y_{in}	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
Epoch-1	1	1	1	1	0	1	1	1	1	1	1
	1	-1	1	-1	1	-1	1	-1	0	2	0
	-1	1	1	-1	2	1	-1	-1	1	1	-1
	-1	-1	1	-1	-3	0	0	0	1	1	-1

Epoch-2 Solve by same thing:-

$[w_1, w_2, b] = [1, 1, -1]$



*) Bayes Theorem

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \quad — (1) \Rightarrow P(A \cap B) = P(A/B) \cdot P(B)$$

$$P(B/A) = \frac{P(B \cap B)}{P(A)} \quad — (2) \Rightarrow P(B \cap A) = P(B/A) \cdot P(A)$$

$$\therefore P(A \cap B) \Rightarrow P(A/B) \cdot P(B) = P(B/A) \cdot P(A)$$

$$\Longrightarrow P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)} \rightarrow \text{formulae}$$

- $P(A/B)$ ← evidence
- B ↓ hypothesis
- $P(B)$ → Marginal
- $P(A)$ → Prior
- $P(B/A)$ → Likelihood
- $P(A/B)$ → Posterior

QY Ans

$$P(\text{King}/\text{Face}) \Rightarrow \frac{P(\text{Face}/\text{King}) \cdot P(\text{King})}{P(\text{Face})}$$

$$\Rightarrow \frac{1 \cdot 4/52}{12/52} = \boxed{\frac{1}{3}}$$

Fun 4
PAGE NO.:
DATE:

*) Naive Bayes Classifier - Example solving

(1) Main label → individual probabilities

$P(\text{PlayTennis} = \text{yes}) = 9/14$

$P(\text{PlayTennis} = \text{no}) = 5/14$

(2) Conditional probabilities of each attribute w.r.t label

(3) Use formula:-

$$v_{NB} = \underset{v_j \in \{yes, no\}}{\text{argmax}} \; p(v_j) \prod_i P(a_i | v_j)$$

v_j → label, a_i → attribute

(4) Check for higher probability of the labels

(5) Normalize the values

Fun 4
PAGE NO.:
DATE:

CNN architecture

i) Kernel or Filter or feature detectors :- $O = [i - k] + 1$

ii) Stride :- $O = \left[\frac{i - k}{s}\right] + 1$

(s) → stride = 2

iii) Padding :- $O = \left[\frac{i - k + 2p}{s}\right] + 1$ → padding = 1

(Border problem solve)

(s) → stride = 1

iv) Pooling → max pooling ⇒ check for 2×2 max matrix
→ avg pooling ⇒ check for 2×2 avg matrix

v) Flatten → 1-d array

*) Back propogation in deep learning with Algorithm.

Ans

- It is a widely used algorithm for training artificial neural networks in deep learning.
- It is an optimization technique that enables the neural network to learn from its mistakes and improve the performance over time.
- The back propogation algorithm can be summarized as follows:-

1) Initialize the weights and biases randomly.
2) Feed the input data to the neural network and compute the output of each neuron using forward propogation algorithm.
3) Compute the error of the output layer by comparing the predicted output with the true output using a loss function.
4) Compute the error of the hidden layers using the errors of the output layer and the weights connecting.
5) Update the weights and biases of the neural network using the gradient descent method.
6) Repeat steps 2-5 for a fixed no. of epochs or until the desired accuracy is achieved.

Fun 4
PAGE NO.:
DATE:

- The key step in the back propogation algorithm is the calculation of the gradients of the loss function w.r.t the weights & biases of the network.
- This is done by using the chain rule of calculus to propogate the error from the output to the hidden layer.
- The main goal of this above method is to minimize the loss function
- Overall, the back propogation is a powerful algorithm for training deep neural networks and has led to many break throughs.

21/04/23

NAIVE BAYES - EXAMPLE 2

Q)

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

(a) $P(M) = \frac{4}{8} = \frac{1}{2}$

$P(H) = \frac{4}{8} = \frac{1}{2}$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

∴

Color	Species M	Species H
White	2/4	3/4
Green	2/4	1/4

Legs	Species M	Species H
2	1/4	4/4
3	3/4	0/4

Height	Species M (M)	Species H (H)
Short	3/4	2/4
tall	1/4	2/4

Smelly	Species M	Species H
Yes	3/4	1/4
No	1/4	3/4

$\therefore$

$\therefore v_{NB} = p(v_j) \cdot \prod_i p(a_i / v_j)$ (attribute, label)

$\Rightarrow v_{NB} = p(M) \cdot p(\text{Color} = \text{green} / M) \cdot p(\text{legs} = 2 / M) \cdot p(\text{Height} = \text{tall} / M) \cdot p(\text{Smelly} = \text{No} / M)$

$= \frac{1}{2} \cdot \frac{2}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4}$

$= \frac{1}{64} = \boxed{0.0156}$

$\Rightarrow v_{NB} = p(H) \cdot p(\text{color} = \text{green} / H) \cdot p(\text{legs} = 2 / H) \cdot p(\text{Height} = \text{tall} / H) \cdot p(\text{smelly} = \text{No} / H)$

$= \frac{1}{2} \cdot \frac{4}{4} \cdot \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{3}{4}$

$= \frac{3}{64} = \boxed{0.0468}$

$\therefore p(H / \text{new instance}) > p(M / \text{new instance})$

$\Rightarrow$ Hence, the new instance belongs to Species H

Fun 4
PAGE NO.:
DATE:

Q4

Example no.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domstic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

New instance = (Red, SUV, Domestic) → (Yes or No)

(a) $P(Yes) = \frac{5}{10} = \underline{0.5}$

$P(No) = \frac{5}{10} = \underline{0.5}$

	Stolen?	
Color	Yes	No
Red	3/5	2/5
Yellow	2/5	3/5

	Stolen?	
Type	Yes	No
Sports	4/5	2/5
SUV	1/5	3/5

	Stolen?	
Origin	Yes	No
Domestic	2/5	3/5
Imported	3/5	2/5

(b)

$$P(\text{Yes} \mid \text{New instance}) = P(\text{Yes}) \cdot P(\text{color} = \text{red} \mid \text{Yes}) \cdot P(\text{Type} = \text{"SUV"} \mid \text{Yes}) \cdot P(\text{Origin} = \text{Domestic} \mid \text{Yes})$$

$$= \frac{5}{2} \times \frac{3}{5} \times \frac{1}{5} \times \frac{2}{5}$$

$$= \frac{3}{125} = \boxed{0.024}$$

$$P(\text{No} \mid \text{New instance}) = \frac{1}{2} \times \frac{2}{5} \times \frac{3}{5} \times \frac{3}{5}$$

$$= \frac{9}{125} = \boxed{0.072}$$

$\therefore P(\text{No} \mid \text{new instance}) > P(\text{Yes} \mid \text{New instance})$

Fun 4
PAGE NO.:
DATE:

Q4 $P(Y_A) = \frac{65}{100} = 0.65$

$P(Y_B) = \frac{35}{100} = 0.35$

$P(X_B|Y_B) = 0.75$

$P(X_B|Y_A) = 0.35$

Y – Winning football match

X – hosting football match

(a) $P(Y_B|X_B) = \frac{P(X_B|Y_B) \cdot P(Y_B)}{P(X_B)}$

$= \frac{0.75 \times 0.35}{0.75 \times 0.35 + 0.35 \times 0.65}$

$= 0.5357$

(b) $P(Y_A|X_B) = \frac{P(X_B|Y_A) \cdot P(Y_A)}{P(X_B)}$

$= \frac{0.35 \times 0.65}{0.35 \times 0.65 + 0.35 \times 0.75}$

$= 0.4642$

Fun 4
PAGE NO.:
DATE:

*) Autoencoders and its types

- An autoencoder is a type of artificial neural network used learn efficient data codings in an unsupervised manner.
- Goal of an autoencoder is to learn a representation for a set of data, usually for dimensionality reduction by training the network.
- When a representation allows a good reconstruction of its input then it has retained much of the information present in the input
- Recently, the autoencoder concept has become more widely used for learning generative models of data.
- There are, basically, 5 types of autoencoders:-

→ Denoising autoencoder

→ Sparse autoencoders

→ Contractive autoencoder

→ Concrete autoencoder

→ Variational Autoencoder

(i) Sparse Autoencoder:- These have hidden nodes greater than input nodes. They can still discover important features from the data. A generic sparse autoencoder is visualized where the obscurity of a node corresponds with the level of activation.

EQ^n :- $\text{Sparsity}(\theta, \phi) = E_{x \sim \mu_x}\left[\sum_{k \in 1:K} \omega_k S(\hat{\rho_k}, \rho_k(x))\right]$

(ii) Denoising Autoencoder :- They create a corrupted copy of the input by introducing some noise. This helps to avoid the autoencoders to copy the input to the output without learning about the features of the data.

EQ^n :- $\min_{\theta, \phi} L(\theta, \phi) = E_{x \sim \mu_x, T \sim \mu_T}(d(x, (D_\theta \circ E_\phi \circ T)$

(iii) Concrete Autoencoder :- Designed for discrete feature selection. This forces the latent space to consist only of a user; specified number of features.

(iv) Variational Autoencoder :- These models make strong assumptions concerning the distribution of latent variables. It gives significant control over how we want to model our latent distribution unlike the other models.

$$p_\theta(x) = \int_z p_\theta(x, z)\, dz = \int_z p_\theta(x, z) \cdot p_\theta(z)\, dz$$

(v) Contractive Autoencoder :- This adds the contractive regularization loss to the standard autoencoder loss. Objective of this autoencoder is to have a robust learned representation which is less sensitive to small variation in the data.

Eqn :- $$\min_{\theta, \phi} L(\theta, \phi) + \lambda L_{contractive}(\theta, \phi)$$

where :-

$$L_{contractive}(\theta, \phi) = E_{x \sim \mu_{ref}} \| \nabla_x E_\phi(x) \|_F^2$$

*) Hebbian Learning Rule

→ It is one of the first and easiest learning rules in ANN.

→ It is a single layer network i.e. has one input and one output layer.

→ Hebbian learning works by updating the weights between neurons in the neural network for each training sample.

→ It is an unsupervised learning rule.

Hebbian Learning Algorithm

1. Set all weights to zero; $w_i = 0$ for $i = 1$ to n and bias to zero.

2. For each input vector, s(input vector) : t(output pair); repeat step 3-5

3. Set activations for input units with the input vector $x_i = s_i$ for $i = 1$ to n

4. Set the corresponding output value to the output neuron i.e. $y = t$

5. Update the weights & bias by hebb rule :-

$$w_i(new) = w_i(old) + x_i y$$
$$b_i(new) = b_i(old) + y$$

*) Normal Distribution - Check for probabilistic dist.

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{-\left\{\frac{(x-\mu)^2}{2\sigma^2}\right\}}$$

∴ for $p(x)$ to be a normal distribution:-

$$\Rightarrow \int_{-\infty}^{\infty} p(x)\,dx = 1$$

$$\frac{x-\mu}{\sigma} = t$$

$$\frac{dx}{\sigma} = dt$$

$$dx = \sigma\,dt$$

$$\Rightarrow \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\left\{\frac{(x-\mu)^2}{2\sigma^2}\right\}} dx$$

$$\Rightarrow \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2}} \cdot \sigma\,dt$$

$$\Rightarrow \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2}}\,dt$$

$$\frac{t^2}{2} = z$$

$$\frac{2t\,dt}{2} = dz$$

$$dt = \frac{dz}{t}$$

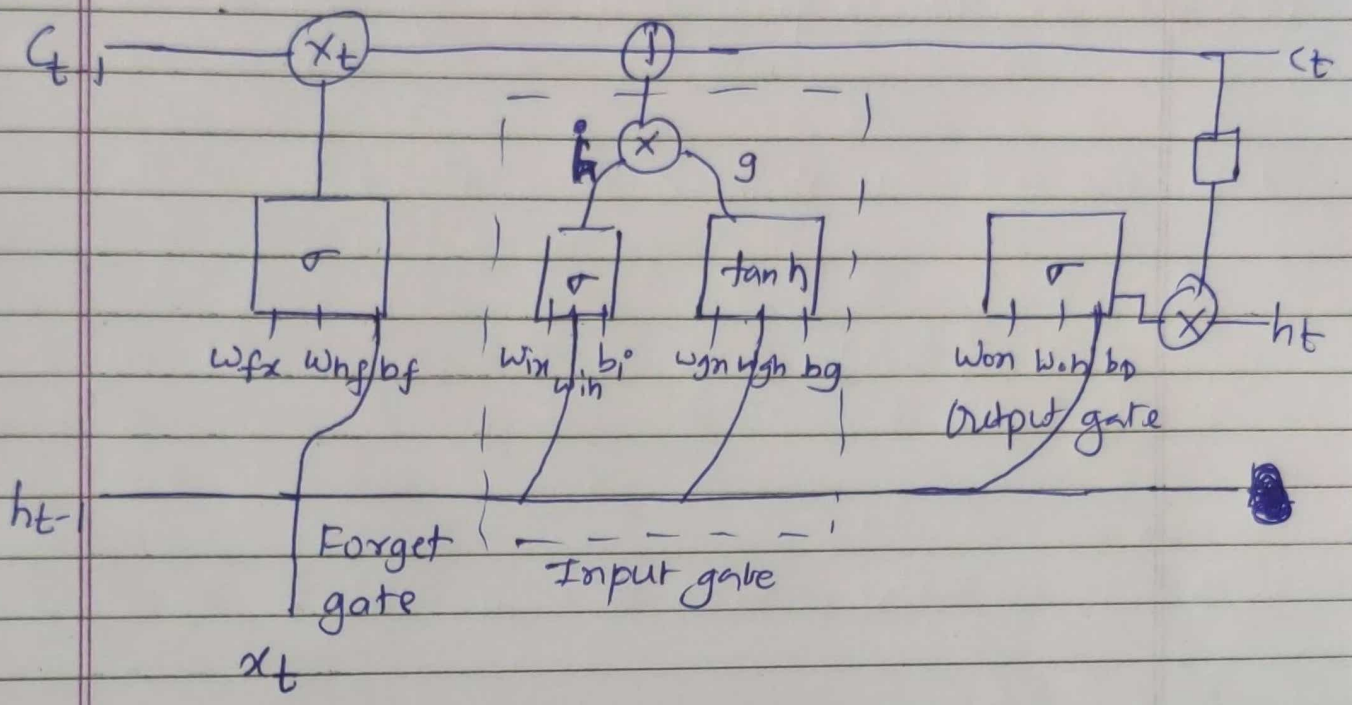
$$\Rightarrow 2\int_{0}^{\infty} \frac{1}{\sqrt{2\pi}} \cdot \frac{e^{-z}}{\sqrt{2z}} \cdot dz$$

$$\Rightarrow \int_{0}^{\infty} \frac{1}{\sqrt{\pi}} \cdot z^{-1/2} e^{-z}\,dz$$

$$\Rightarrow \Gamma(1/2) \cdot \frac{1}{\sqrt{\pi}}$$

$$\Rightarrow \frac{\sqrt{\pi}}{\sqrt{\pi}} = 1$$

*) LSTM Architecture and Derivation



Derivation :-

(i) Forget gate :- $f_t = \sigma[w_{fx} * x_t + w_{hf} * h_{t-1}]$

$$C_t^f = C_{t-1} * f_t$$

(ii) Input gate :- $i_t = \sigma[w_{ix} * x_t + w_{ih} * h_{t-1} + b_g]$

$$g_t = \tanh[w_{gx} * x_t + w_{gh} * h_{t-1} + b_g]$$

(iii) Output gate :- $O_t = \sigma[w_{ox} * x_t + w_{oh} * h_{t-1} + b_o]$

∴ final output $(h_t) = [C_t * \tanh + O_t]$