24/11/2024

UNIT - 1st

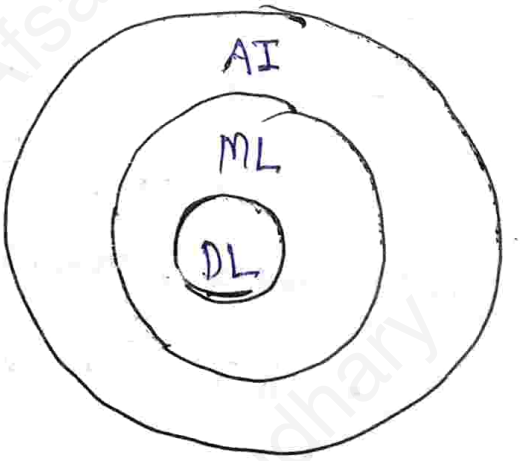
Assar Chaudhary

Machine Learning :-

Machine Learning (ML) is defined as a discipline of Artificial Intelligence (AI) that provides machine the ability to automatically learn from data and past experience to identify patterns and make predictions with minimal human intervention

" Machine learning enables a machine to ~~automatica~~ automatically learn from data, improve performance from experiences, predict things without being explicitly programmed".

Difference b/w AI, ML, AND DL.



- **Artificial Intelligence :-**

AI is the broadcast concept of all, and gives a machine the ability to imitate human behavior.

- **Machine Learning :-**

machine learning uses algorithms and techniques that enables the machine to learn from past experience/trends and predict the output based on that data, their performance improve as they are exposed to more data over time.

- **Deep Learning :-**

DL is subset of machine learning in which multilayered neural networks learn from vast amounts of data.

Types of ML :-

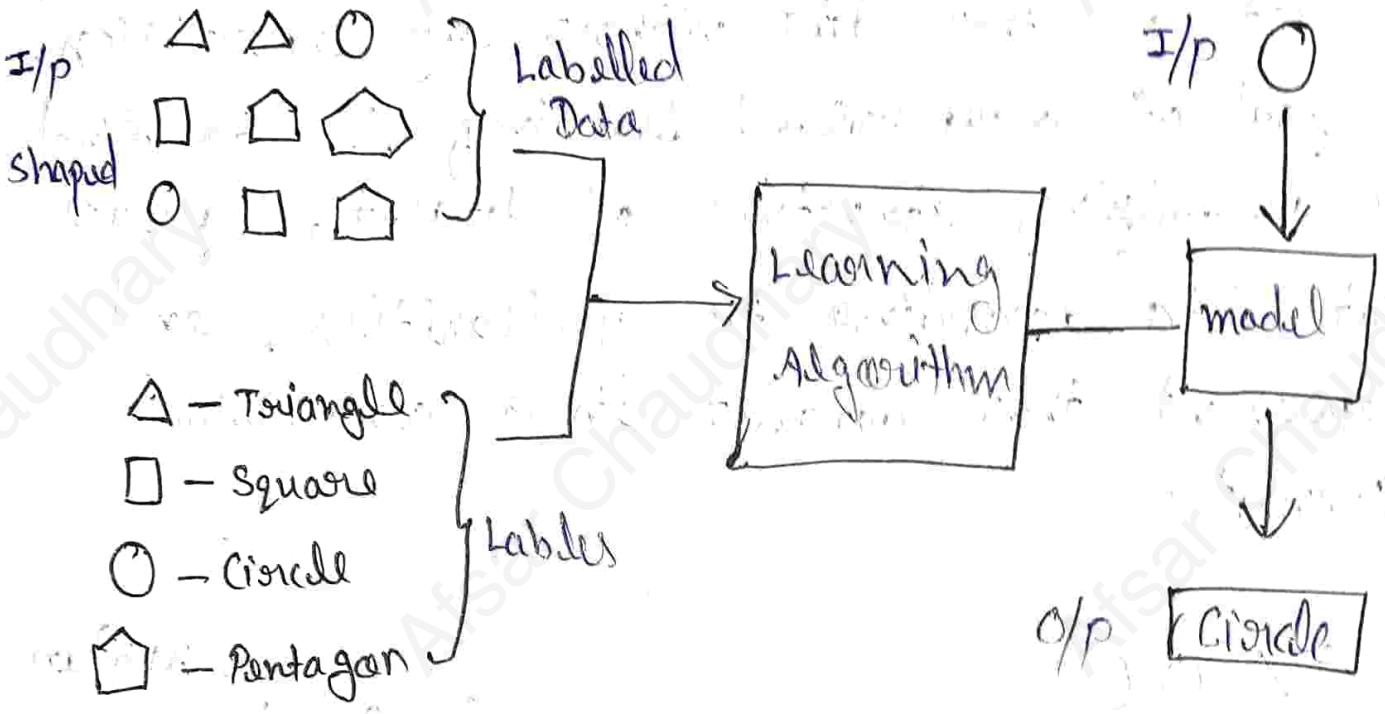
1. **Supervised Learning :-**

Supervised Learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machine predict the output.

The labelled data means some input data

is already tagged with the correct output.

Ex:- Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.



Types of Supervised Learning :-

(i) Classification :-

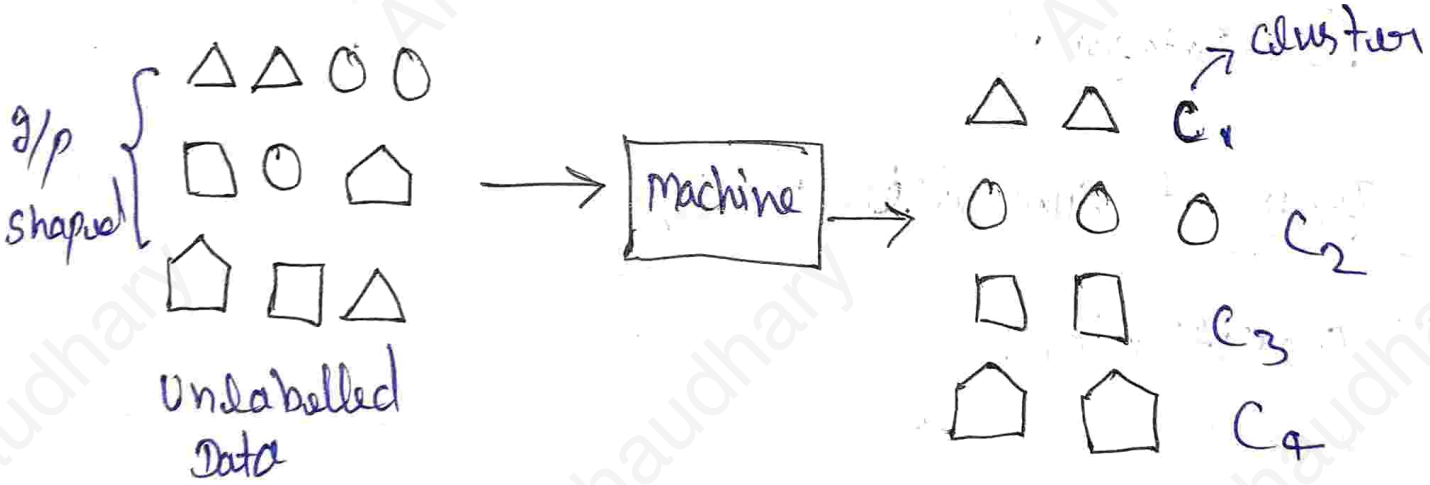
A classification problem is when the output variable is a category, such as "red" or "blue", "disease" and "No disease", Yes-No, male-Female, True-False, etc.

(ii) Regression :-

A regression problem is when the output variable is a real value, such as, Forecasting sales, Weather forecasting etc.

2. Unsupervised Learning :-

- Unsupervised learning is a type of machine learning in which models are trained using an unlabelled dataset and are allowed to act on that data without any supervision.
- The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.



Types of Unsupervised learning :-

(i) Clustering :-

A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour.

(ii). Association :-

Association rule learning is a kind of unsupervised learning technique that tests for the reliance of one data element on another data element and design appropriately so that it can be more cost-effective. It tries to discover some interesting relations or associations b/w the variables of the dataset.

3. Semi-Supervised Learning :-

- Typically, this combination will contain a very small amount of labeled data and a very large amount of unlabeled data.
- The basic procedure involved is that first, the programmer will cluster similar data using an unsupervised learning algorithm and then use the existing labeled data to label the rest of the unlabeled data.

A semi-supervised algorithm assumes the following about the data—

- Continuity Assumption :-

The algorithm assumes that the points which

are closer to each other are more likely to have the same output label.

- Cluster Assumption :-
  The data can be divided into discrete clusters and points in the same cluster are more likely to share an output label.

- Manifold Assumption :-
  The data lie approximately on a manifold of much lower dimension than the input space.

Application of Semi Supervised Learning :-

- Speech Analysis :-
  Since labeling of audio files is a very intensive task.
  Semi-Supervised learning is a very natural approach to solve this problem.

- Internet Content Classification :-
  Labeling each webpage is an impractical and unfeasible process and thus uses Semi-supervised learning algorithms. Even the google search algorithm uses a variant of Semi-Supervised

learning to rank the relevance of a webpage for a given query.

- Protein Sequence Classification :
  Since DNA strands are typically very large in size, the rise of Semi-Supervised learning has been imminent in this field.

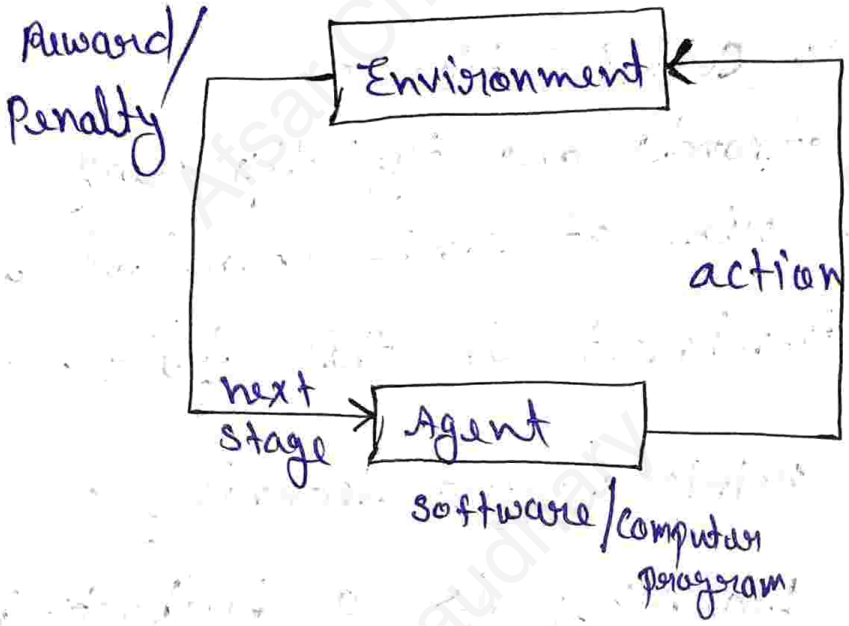
What is Reinforcement learning :-

Reinforcement Learning is a feedback-based machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions, for each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or a penalty.

The main elements of an RL system are :

- The agent or the learner
- The environment the agent ~~follow~~ interacts with
- The policy that the agent follows to take actions
- The reward signal that the agent observes

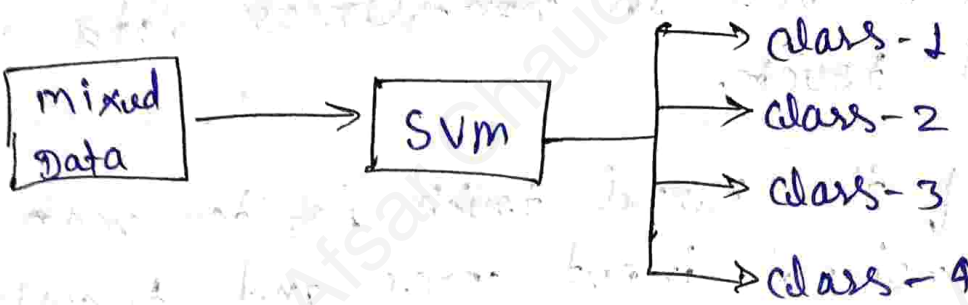
upon taking action.

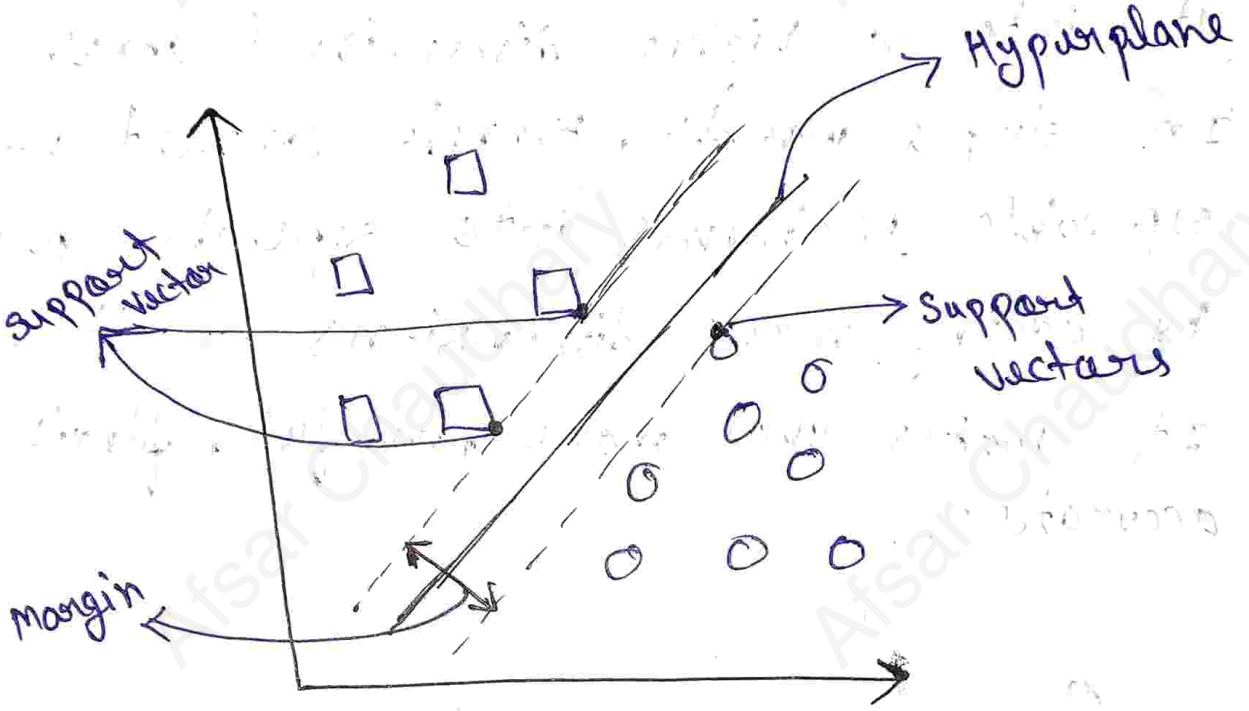


SVM (Support Vector Machine) :-

- SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear or non-linear problems and works well for many practical problems.
- It tries to classify data by finding a hyperplane that maximizes the margin b/w the classes in the training data. Hence, SVM is an example of a large margin classifier.
- The idea of SVM is simple: The algorithm creates a line or a hyperplane which seperates

the data into classes.



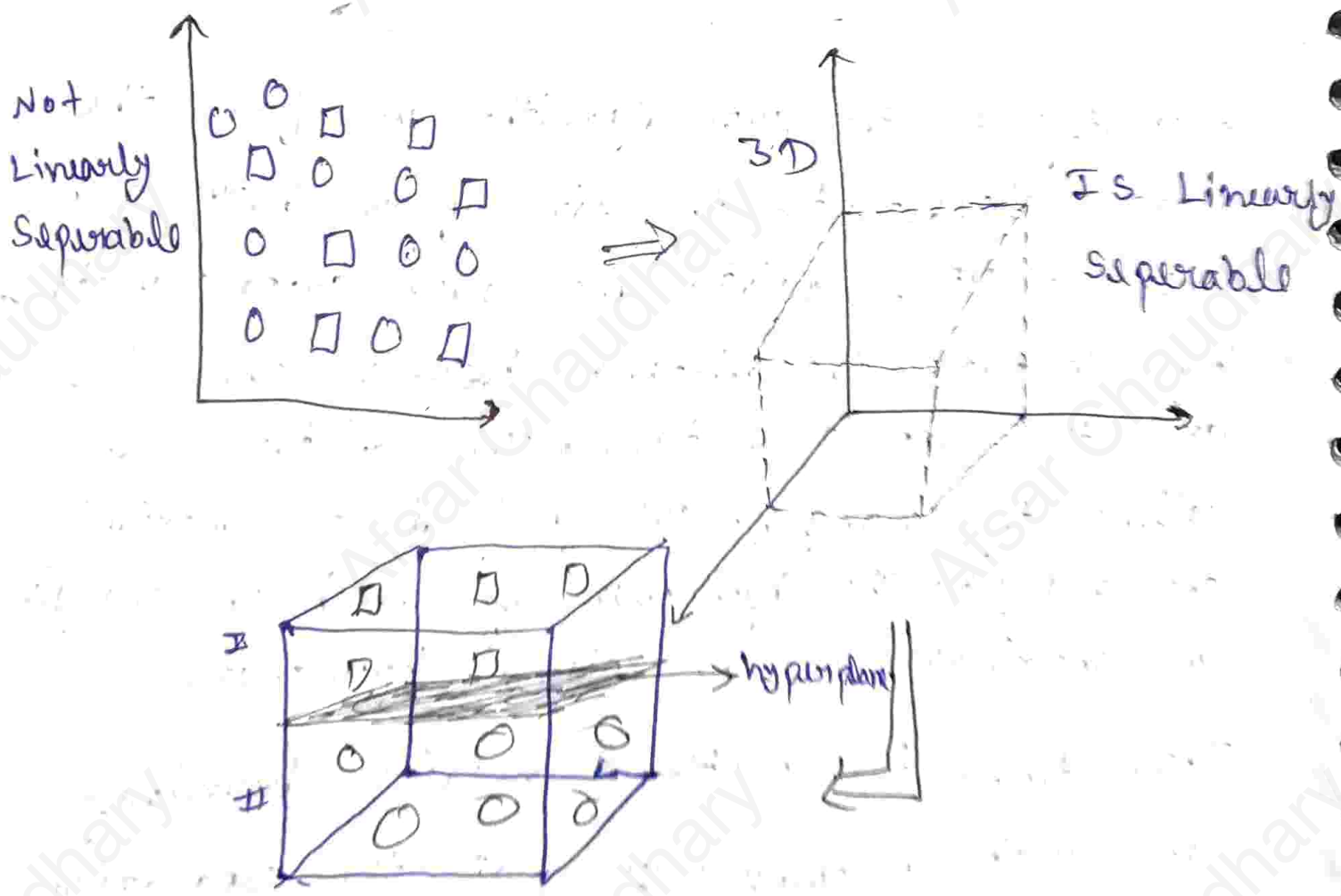


- According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors.
- We compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.

Thus, tries to make a decision boundary in such a ~~way~~ way that the separation b/w the two classes is as wide as possible

SVM KERNELS :-

* SVM can work well in non-linear data cases using kernel trick.
* The function of the kernel trick is to map the low-dimensional input space and transform it into a higher dimensional space.
* In simple words, kernels convert non-separable problems into separable problems by adding more dimensions to it.
* It makes SVM more powerful, flexible and accurate.



Types of SVM Kernels :-

1. **Linear Kernel :-**

It is used when the data is linearly separable.

$$K(x, y) = x \cdot y \longleftarrow \text{i/p features}$$

2. **Polynomial Kernel :-**

It is used when the data is not linearly separable.

$$K(x, y) = (x \cdot y + 1)^d$$

where $d \rightarrow$ degree of polynomial

3. **Gaussian Kernel :-**

It is used to perform transformation when there is no prior knowledge about data.

$$K(x, y) = e^{(-\gamma (x-y)^2)}$$

4. **Exponential or Laplace Kernel :-**

It can be used to measure the similarity or distance between two input feature vectors,

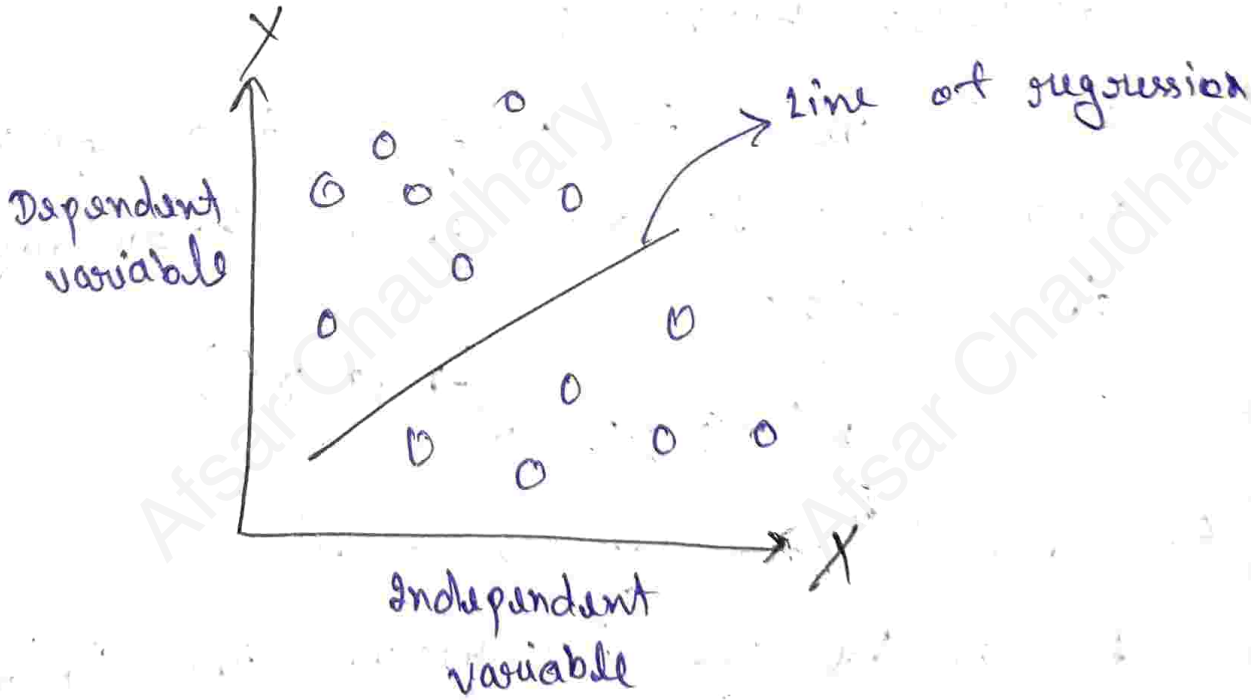
$$k(x, y) = e^{(-\gamma (x-y))}$$

Linear Regression :-

* Linear regression is a statistical method used for modeling the relationship b/w a dependent variable and one or more independent variables by fitting a linear equation to the observed data points that minimizes the sum of the squared differences b/w the observed and predicted values.

* The linear regression equation for a simple linear regression with one independent variable can be expressed as —

$$y = mx + b$$



where →

$y \rightarrow$ Dependent variable

$x \rightarrow$ Independent variable

$m \rightarrow$ slope of the line

$b \rightarrow$ y-intercept

* In the case of multiple linear regression, where there are more than one independent variable, the equation becomes:

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \text{---}$$

Logistic Regression :-

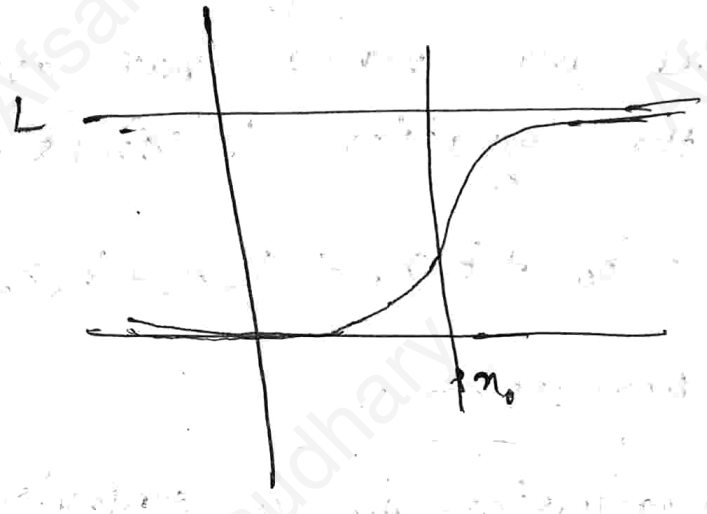
* Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not.

* It is the powerful tool for decision-making. For example - email spam or not.

* It can be either 'Yes' or 'No', 0. or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie b/w 0 and 1.

* Logistic Regression is much similar to the Linear Regression except that how they are used. Linear regression is used for solving Regression problems, whereas Logistic

regression is used for solving the classification problems.



$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$

$k \rightarrow$ Growth Rate

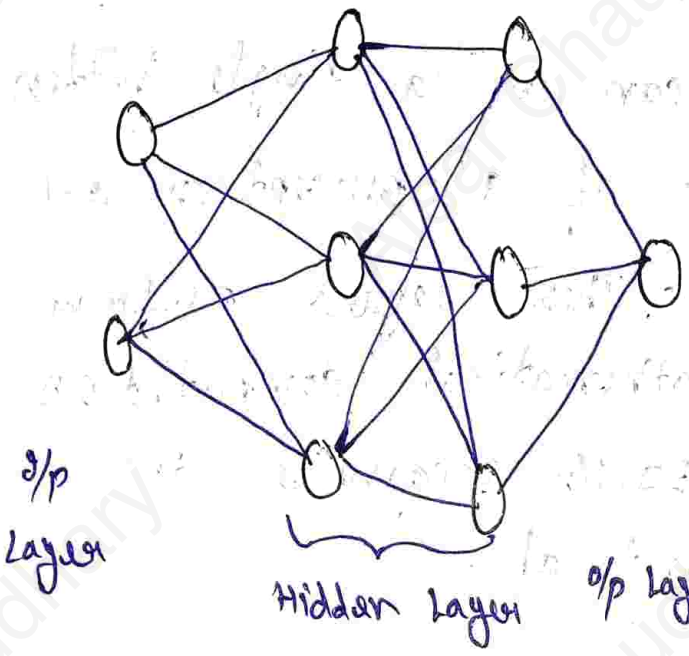
$x_0 \rightarrow$ value of mid point

$L \rightarrow$ max. value

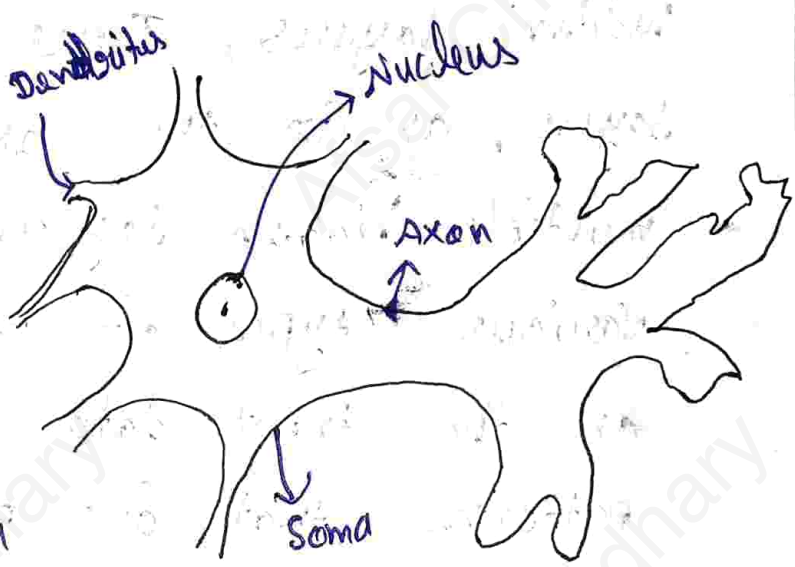
ANN Vs BNN :-

The term "Artificial Neural Network" is derived from Biological Neural Networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the

network. These neurons are known as nodes.



ANN



BNN

BNN	ANN
Soma	Node
Dendrites	Input
Synapse	Weights
Axon	Output

In a neural network, there are three essential layers –

(1) Input Layer :-

The I/p layer is the first layer of an ANN that receives the I/p info. in the form of various texts, numbers, audio files, image pixels, etc.

(ii) Hidden Layer :-

In the middle of the ANN model are the hidden layers, There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These layers perform various types of mathematical computation on the input data BEING recognize the patterns that are part of.

(iii) Output Layer :

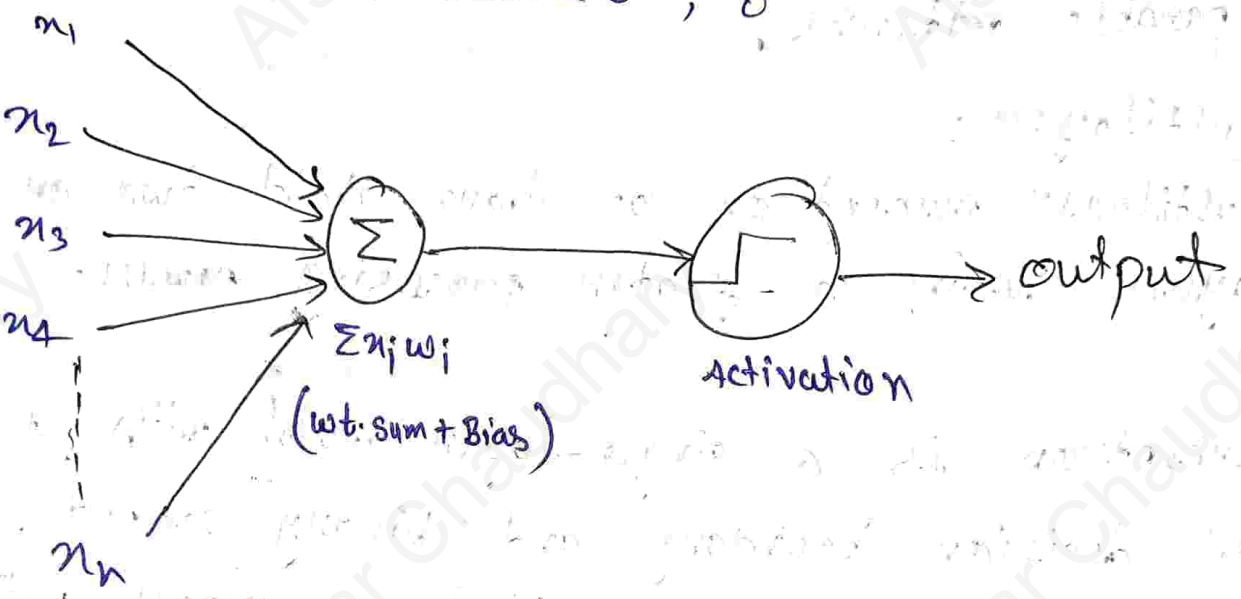
In the output layer, we obtain the result that we obtain through rigorous computations performed by the middle layer.

Perceptron :-

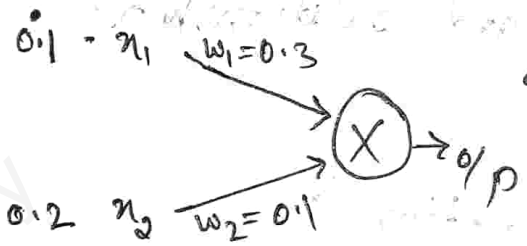
* A perceptron is a binary classifier that has one layer of input nodes and it directly produces output.
* The inputs are multiplied by weights and the weighted sum is calculated.
* A bias term b is added to the weighted sum.
* The result is passed through an activation function, commonly a step function or a threshold function.

* The output is binary (0 or 1) based on the result of the activation.

wt sum ≥ 0 , 1
wt sum < 0 , 0



Example :-



& Bias = 0.02

Soln :- $x_1 w_1 + x_2 w_2 + bias$

$\Rightarrow 0.1 \times 0.3 + 0.2 \times 0.1 + 0.02$

$\Rightarrow 0.03 + 0.02 + 0.02 = 0.07$

Step func^n

Sum ≥ 0 , 1

Sum < 0 , 0

$\therefore 0.07 \geq 0$ So o/p = 1

Types of perceptron :-

(i) **Single Layer :-**

Single Layer perceptron can learn only linearly separable patterns.

(ii) **Multilayer :-**

Multilayer perceptron can learn about two or more layers having a greater processing power.

A perceptron is a single-layer model with a linear decision boundary and binary output, while an MLP is a multi-layer model with non-linear activation functions, capable of learning complex patterns and relationships in the data.

Cost Function & Loss Function :-

Cost Function :-

When discussing the overall performance of the model across the entire training dataset, you might refer to the cost function

Loss Function :-

When discussing the error for a single data point or a mini-batch of data during the training

process, you might refer to the loss function.

Fast "mean square error" (MSE) it would be:

$$\text{Loss} = \frac{1}{2}(y - \hat{y})^2 \quad , \quad \text{Cost} = \frac{1}{2n}\sum (y_i - \hat{y}_i)^2$$

(y: actual value, $\hat{y}$: predictable value)

For Example

Study Hours	Mid-term Score	Predicted Final exam Score	Actual Final exam score
5	85	88	80
3	75	74	71
8	90	91	93
6	80	81	84
7	88	90	85

$\text{Loss} = \frac{1}{2}(y - \hat{y})^2$

$L_1 = \frac{1}{2}(80 - 88)^2 = 32$

$L_2 = \frac{1}{2}(71 - 74)^2 = 4.5$

$L_3 = \frac{1}{2}(93 - 91)^2 = 2$

$L_4 = \frac{1}{2}(84 - 81)^2 = 4.5$

$L_5 = \frac{1}{2}(85 - 90)^2 = 12.5$

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

Here $n = 5$

$\therefore C = \frac{1}{2\times 5}\left[(80-88)^2 + (71-74)^2 + (93-91)^2 + (84-81)^2 + (85-90)^2\right]$

$= \frac{1}{10}\{64 + 9 + 4 + 9 + 25\}$

$= \frac{1}{10} \times 111 = 11.1$ Ans

Various types of Loss Function :-

Loss functions, also known as objective functions or cost functions, are used in machine learning to quantify the difference b/w predicted values and actual values. The choice of a specific loss function depends on the nature of the task, such as regression, classification, or other specialized tasks. Here are some common types of loss functions categorized by task along with their formulas :

[A] Regression Loss Functions :-

1. **Mean Square Error (MSE) :**

Formula : $L = \frac{1}{2}(y - \hat{y})^2$, $C = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Used for : Regression tasks where predicted values are continuous.

2. **Mean Absolute Error (MAE) :**

Formula : $L = |(y - \hat{y})|$, $C = \frac{1}{n}\sum_{i=1}^{n}|(y_i - \hat{y}_i)|$

Used for :- Regression tasks where the absolute diff. b/w actual and predicted values is important.

3. Huber Loss :-

Formula : $\frac{1}{n}\sum_{i=1}^{n} \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & , \text{if } |y - \hat{y}| \le \delta \\ \delta(y_i - \hat{y}_i) - \frac{1}{2}\delta^2 & , \text{otherwise} \end{cases}$

Used for :- A robust regression loss that is less sensitive to outliers.

(B) Classification Loss Functions:

1. Binary Cross-Entropy Loss (Log Loss):

Formula: $C = -\frac{1}{n}\sum [y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)]$

Used for : Binary classification tasks.

2. Categorical Cross-Entropy Loss :

Formula : $\text{Loss} = -\sum_{j=1}^{K} y_j \log \hat{y}_j$

Used for : Multiclass classification tasks.

How to train a Neural Network ?

(i) Data collection and preprocessing :

- Gather a labeled dataset that includes input data and corresponding target labels.
- Clean and preprocess data. This may involve normalization, standardization, handling missing

values, and other preprocessing step to prepare the data for training.

(ii) Model Architecture Design :-

Choose the architecture of your neural network, including the number of layers, the number of neurons in each layers, and the activation functions. This step defines the model's capacity to learn and represent patterns.

(iii) Initialization :-

Initialize the weights and biases of neural network. Proper initialization is crucial for efficient training. Common techniques include Xavier/Glorot initialization or He initialization.

(iv) Forward Propagation :-

Perform forward propagation to compute predictions for the input data. Each layer's output is calculated by applying weights and biases, followed by an activation function.

(v) Loss Computation :-

Compare the predictions with the true labels and compute the loss using a suitable loss function. The choice of the loss function depends on the nature of the task (e.g., mean square error for regression,

(cross - entropy for classification).

$$\text{Loss} = \frac{1}{2}(y - \hat{y})^2 \quad (\text{MSE})$$

(vi) Backpropagation :-

Perform backpropagation to compute the gradients of the loss with respect to the weights and biases in the network. This involves calculating the partial derivatives of the loss with respect to each parameter using the chain rule.

(vii) Gradient Descent Optimization :-

Use an optimization algorithm (e.g., stochastic gradient descent, Adam, RMSprop) to update the weights and biases in the direction that minimizes the loss.

$$W_{new} = W_{old} - \eta \frac{dL}{dW_{old}}$$

(L → Loss; η → Learning-Rate)

(viii) Repeat :-

Step 4 to 7 are repeated for multiple epochs (Passes through the entire dataset). Each epoch refines the model's parameters to improve performance on the training data.

(ix) Evaluation on Test Set :-

Evaluate the trained model on a separate test set to assess its generalization performance. This set provides an unbiased estimate of the model's performance on new, unseen data.

(x) Deployment :-

Once satisfied with the model's performance, deploy it for making predictions on new, real-world data.

Gradient Descent :-

Gradient descent is an optimization algorithm which is commonly used to train machine learning models and neural networks, to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning (DL) to minimize a cost/loss function.

Types of Gradient Descent :-

1. Batch Gradient Descent

2. mini Batch Gradient Descent
3. stochastic Gradient Descent

1. Batch Gradient Descent :-

* Batch gradient descent, also known as vanilla gradient descent, calculates the errors for each example within the training dataset.
* The gradients are based on the average of the gradients of the entire dataset.
* The batch gradient descent method typically requires the entire training dataset in memory and is implemented for use in the algorithm.

For Example

Study Hours	mid-term score	Predicted final exam score	Actual - final exam score
5	85	88	80
3	75	74	71
8	90	91	93
6	80	81	84
7	88	90	85

$$e_1 = \frac{1}{2}(y-\hat{y})^2 = \frac{1}{2}(80-88)^2 = 32$$

$$e_2 = \frac{1}{2}(71-74)^2 = 4.5$$

$$e_3 = \frac{1}{2}(93-91)^2 = 2$$

$$e_4 = \frac{1}{2}(84-81)^2 = 4.5$$

$$e_5 = \frac{1}{2}(85-90)^2 = 12.5$$

$$E_{avg} = \frac{e_1+e_2+e_3+e_4+e_5}{5} = \frac{32+4.5+2+4.5+12.5}{5}$$

$$= \frac{55.5}{5} = 11.1$$

Update

$$W_{new} = W_{old} - \eta \frac{d E_{avg}}{d W_{old}}$$

2. Stochastic Gradient Descent (SGD) :-

* In SGD, only one randomly chosen data point is used to compute the gradient at each iteration.
* Gradients are based on individual data points.
* Computationally more efficient, especially for large.

3. mini Gradient Descent :-

* mini-batch gradient descent combines the ideas of batch gradient descent with SGD.
* It uses a randomly selected subset (mini batch) of the training data at each iteration.
* Choice of mini-batch size is a hyperparameter that needs tuning

Difference

Batch Gradiant Descent	Stochastic Gradient Descent	minibatch Gradient Descent
Use all training samples for one forward pass and then adjust weights.	Use one (randomly picked) sample forward pass and then adjust weights.	Use a batch of (randomly picked) samples for a forward pass and then adjust weights.

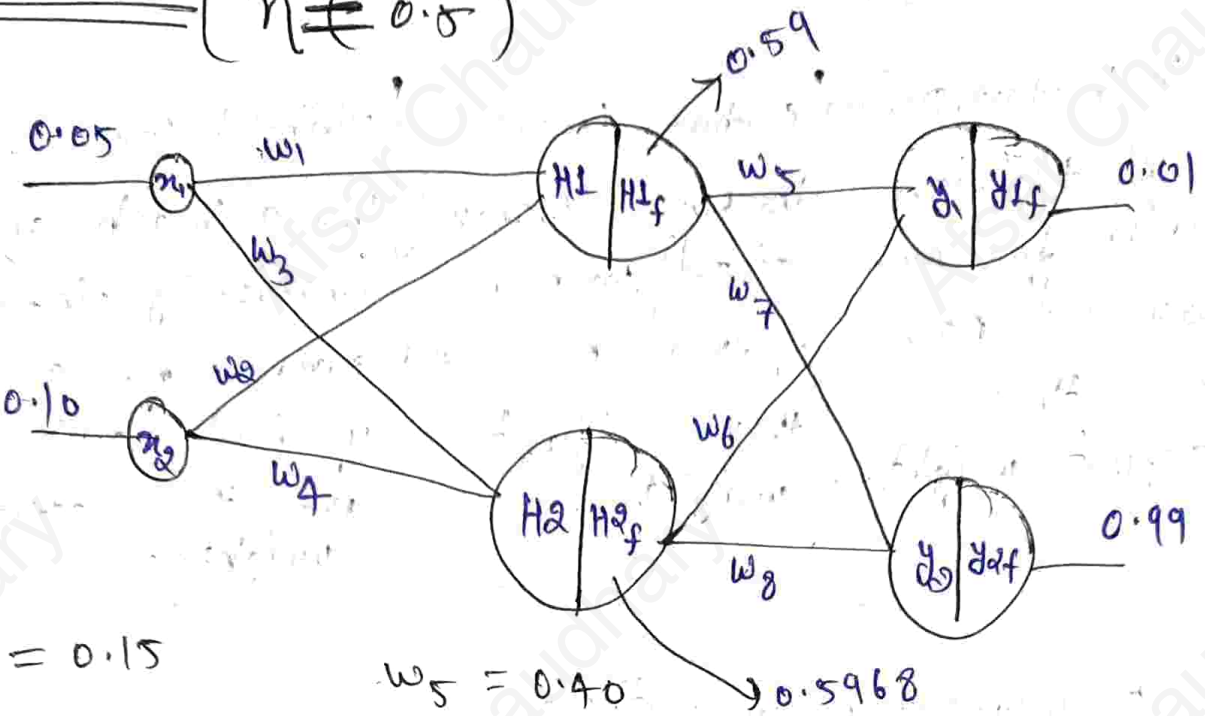
Back propagation Algorithm

The algorithm for BPN is as classified into four major steps as follows:

1. Initialization of Bias, weights
2. Feedforward process - calculate $\Sigma x_i w_i$
   - Use activation function (Sigmoid function)
   - Find Loss (mean square error)
3. Back Propagation of Errors - using Gradient Descent
4. Updating of weights & biases

Repeat 2-w for n number of epochs

Numerical $(\eta = 0.5)$



$w_1 = 0.15$
$w_2 = 0.20$
$w_3 = 0.25$
$w_4 = 0.30$
$b_1 = 0.35$

$w_5 = 0.40$
$w_6 = 0.45$
$w_7 = 0.50$
$w_8 = 0.55$
$b_2 = 0.60$

Soln:- Step-1 $H_1 =$

$H_1 = 0.05 \times 0.15 + 0.10 \times 0.20 + 0.35 = 0.3775$

$$H_{1f} = \frac{1}{1+e^{-H_1}} = \frac{1}{1+e^{-0.3775}} = 0.59$$

$H_2 = 0.05 \times 0.25 + 0.10 \times 0.30 + 0.35 = 0.3925$

$$H_{2f} = \frac{1}{1+e^{-H_2}} = \frac{1}{1+e^{-0.3925}} = 0.5968$$

$y_1 = H_{1f} \cdot w_5 + H_{2f} \cdot w_6 + b_2$

$y_1 = 0.59 \times 0.40 + 0.5968 \times 0.45 + 0.60$

$= 1.105$

$$y_{1f} = \frac{1}{1+e^{-y_1}} = \frac{1}{1+e^{-1.105}} = 0.7513$$

Similarly -

$$y_{2f} = 0.773$$

Step-II Loss/Error (using mean Square Error) $= \frac{1}{2}(\text{target} - o/p)^2$

$$E_{total} = \frac{1}{2}(t_1 - y_{1f})^2 + \frac{1}{2}(t_2 - y_{2f})^2$$

$$= \frac{1}{2}(0.01 - 0.773)^2 + \frac{1}{2}(0.99 - 0.77)^2 = 0.2983$$

Step-III Back Propagation / Gradient Descent

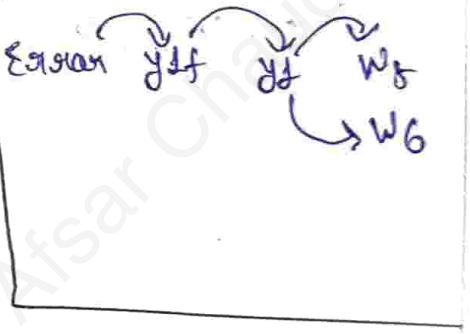
$$w_{new} = w_{old} - \eta \frac{\partial \text{Error}}{\partial w_{old}}$$

adjust w_5 by Back propagation

$$w_{5new} = w_5 - \eta \frac{\partial \text{Error}}{\partial}$$

$$\frac{\partial \text{Error}}{\partial w_5} = \frac{\partial \text{Error}}{\partial y_{1f}} \times \frac{\partial y_{1f}}{\partial y_1} \times \frac{\partial y_1}{\partial w_5}$$

Note chain Rule



$$\frac{\partial y_1}{\partial w_5} = \frac{\partial}{\partial w_5}(H_{1f} \cdot w_5 + H_{2f} \cdot w_6 + b_0)$$

$$\boxed{\frac{\partial y_1}{\partial w_5} = H_{1f}}$$

$$\frac{\partial E_{total}}{\partial y_{1f}} = \frac{\partial}{\partial y_{1f}}\left[\frac{1}{2}(t_1 - y_{1f})^2 + \frac{1}{2}(t_2 - y_{2f})^2\right] = -(t_1 - y_{1f})$$

$$\boxed{\frac{\partial E_{total}}{\partial y_{1f}} = -(t_1 - y_{1f})}$$

$$\frac{\partial y_{1f}}{\partial y_1} = \frac{\partial}{\partial y_1}\left(\frac{1}{1+e^{-y_1}}\right) = \frac{0 - e^{-y_1}(-1)}{(1+e^{-y_1})^2} = e^{-y_1}(y_{1f})^2$$

$\because$ $y_{1f} + y_{1f}e^{-y_1} = 1 \Rightarrow e^{-y_1} = \frac{1 - y_{1f}}{y_{1f}}$

$\therefore$ $\frac{\partial y_{1f}}{\partial y_1} = \left(\frac{1 - y_{1f}}{y_{1f}}\right)(y_{1f})^2 = (1 - y_{1f})(y_{1f})$

Similarly we can adjust w_1, w_2, w_3 w_4, w_6, w_7 and w_8.