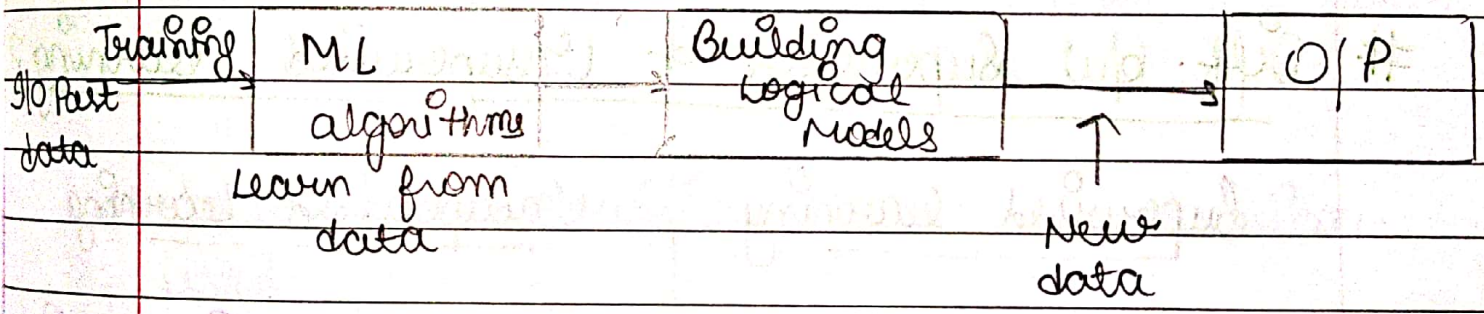
Unit :- 1st

DOMS | Page No 1
Date / /

Machine learning Basics

A subset of AI known as ML focuses primarily on the creation of algorithms that enable a computer to independently learn from data & previous experiences.

- ML algo. create a mathematical model that, without being explicity programmed, aids in making predictions or decisions with the assistance of sample historical data or training data.



→ Features of ML :-

(i) ML uses data to detect various patterns in a given dataset.
(ii) It can learn from past data & improve automatically.
(iii) Data driven technology.
(iv) ML is much similar to data mining as it also deals with huge amount of

data.

→ Importance of ML :-

(i) Rapid Increment in the Production of data.
(ii) Solving complex problems, which are difficult for human.
(iii) Decision making in various sector including finance.
(iv) Finding hidden Patterns & extracting useful info. from data.

Diff. b/w Supervised & Unsupervised learning?

Supervised learning	Unsupervised Learning
(i) Algo. are trained using labeled data.	Algo. are trained using Unlabeled data.
(ii) It takes direct feedback to check if it is predicting correct O/P or Not.	This model does not take any feedback.
(iii) Simpler Method.	Computationally complex.

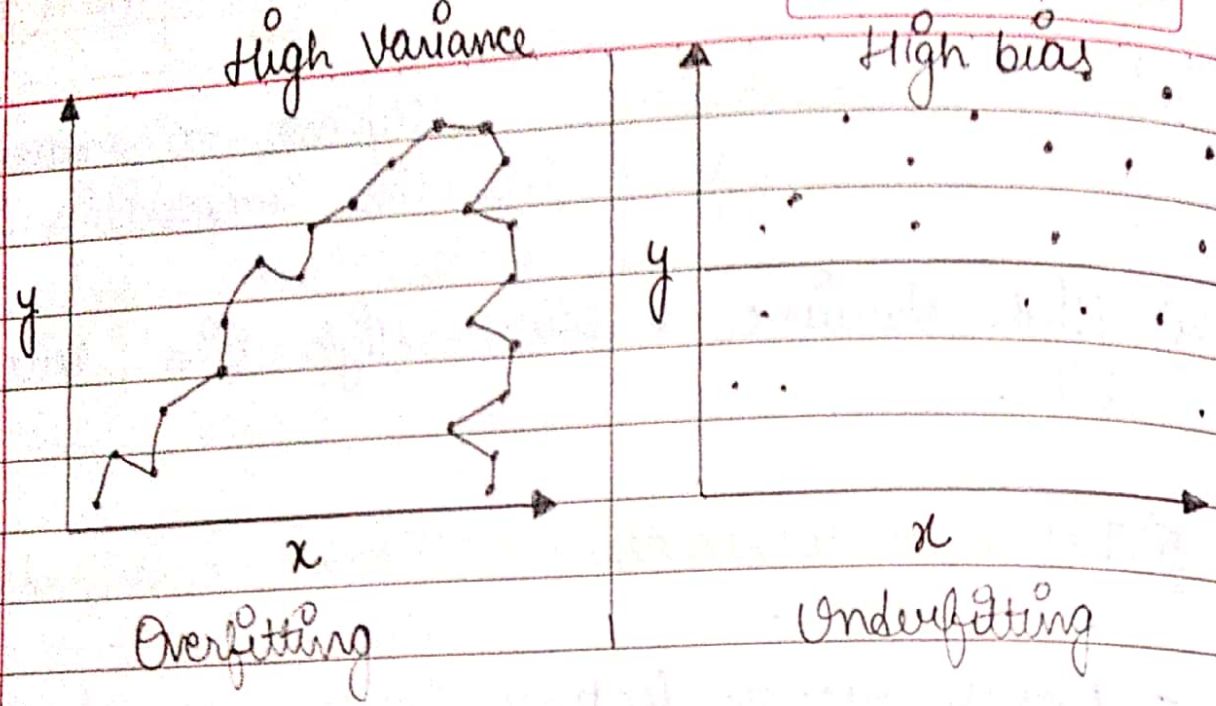
(iv) Highly accurate	less accurate.
(v) No. of classes is known.	No. of classes is Unknown.
(vi) Desired O/P is given.	Desired O/P is not given.
(vii) Used training data to infer the model.	No training data is used.
(viii) we can test our model.	we cannot test our Model.
(ix) External supervision is used.	No supervision.
(x) Its aim is to calculate Outcomes.	Its aim to discover Underlying patterns.
(xi) It can be categorized in - classification - Regression problems.	It can be classified in - clustering - Association problems.
(xii) It includes various algo. such as Linear	It includes various algo. such as KNN,

regression, logistic regression, SVM, Multi class classification, Decision tree, Bayesian logic etc.	Clustering & Apriori algo.

Difference b/w Overfitting & Underfitting?

Overfitting	Underfitting
1) The chances of occurence of overfitting increase as much we provide training to our model.	In case of underfitting, the model is not able to learn enough from the training data & hence it reduces the accuracy & produce unreliable predictions.
2) Overfitting is the main problem that occurs in supervised learning.	The model is not able to capture the data points present in the plot.
3) The model is too complex.	The model is too simple so it may be not

	Capable to represent the complexities in data.
4) High Variance & low bias.	High bias, low Variance.
5) Techniques to reduce overfitting :-	Techniques to reduce underfitting :-
- Reduce no. of features - Reduce no. of parameters - Increase the amount of data. - use dropout for neural N/w to tackle overfitting.	Increase no. of features Increase no. of parameters Reduce the amount of data. Remove noise from the data.
6) Apply Regularization.	Reduce Regularization
7) Reduce complexity of model.	Increase the complexity of model.
8) The size of the training data is enough.	The size of training dataset is not enough
9) Diagram :-	Diagram :-



Bias VS Variance

Bias :- Gap b/w actual value of model & predicted value of data.

- **High Bias :-** Predicted value is more far away then actual value.
(Not accurate) (underfitting)

- **Low Bias :-** Predicted value is near to actual value.
(Accurate) (overfitting)

→ **Ways to reduce high Bias**

(i) use a more complex model ⟷ Polynomial regression, CNN, RNN

(ii) Increase the no. of features

(iii) Reduce Regularization of the model → L1 + L2 regularization
(iv) Increase the size of the training data.

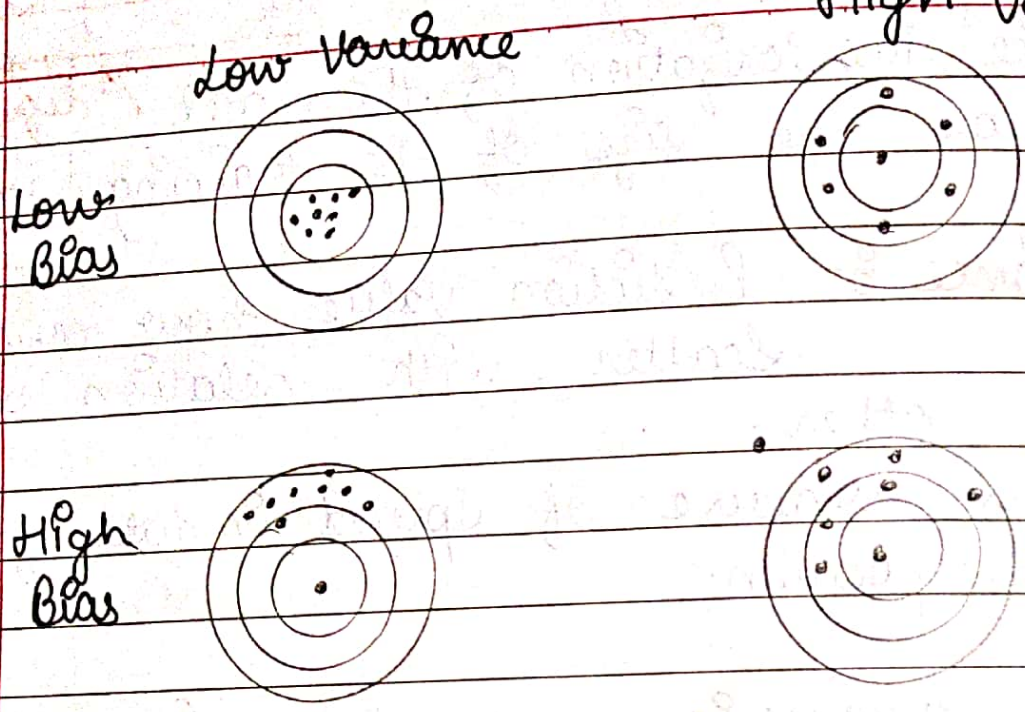
Variance :- Prediction value how much scatter with relation b/w each other.

- It is measure of spread in data from its mean position.
- Low Variance :- group of predicted value does not scatter with each other.
- High Variance :- Scatter with each other. (overfitting).

→ Ways to reduce Variance

(i) Cross Validation
(ii) Feature Selection
(iii) Regularization
(iv) Ensemble methods < Bagging Boosting
(v) Early stopping
(vi) Simplifying the model.

Diff combinations of Bias-Variance



- High Bias, Low Variance = underfitting
- " Variance, " Bias = Overfitting
- " Bias, High Variance = Not able to capture Underlying Patterns
- Low Bias, Low Variance = Model is able to capture the underlying patterns in data & not too sensitive changes in training data.

→

ML Algorithm	Bias	Variance
Linear regression	High	Low
Decision tree	Low	High
Random forest	Low	High
Bagging	Low	High

Estimators

Estimation is a statistical term

for finding some estimate of unknown parameter, given some data. They help a crucial role in model training & evaluation, helping to optimize model parameters & assess model performance.

Types of Estimators :-

(i) Point Estimators :- It provides a single value estimate for the unknown parameter.

eg
Mean, Median & mode.

(ii) Interval Estimator :- It provides a range of values within which the true parameter is likely to lie.

eg
Confidence intervals & prediction intervals.

Applications

(i) Parameter Estimation :- Estimators are used to estimate model parameters, such as weights & biases, during training process.

(ii) Model evaluation :- used to evaluate model performance on unseen data, providing metrices such as accuracy, precision & recall.

(iii) Feature Importance :- It can be used to asess the imp. of diff. features to model, helping to identify the most influential factors in predictions.

(iv) Uncertainity Quantification :- It can provide measures of uncertainity associated with model predictions, indicating the confidence in model's o/p.

Maximum likelihood estimation

It is a method that determines values for the parameters of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.

- Maximum likelihood is an approach commonly used for such density estimation problems, in which a likelihood function is defined to get the probabilities of distributed data.

MLE Intuition (Maximum likelihood estimate plot)

Multiple PDFs over the random sample histogram plot

We calculate likelihood based on conditional probabilities.

$$L = f([X_1 = x_1], [X_2 = x_2] \dots [X_n = x_n] / P) = \prod_{i=1}^{n} p^{x_i} (1-p)^{1-x_i}$$

where,

- $L \rightarrow$ Likelihood Value
- $F \rightarrow$ Probability distribution function
- $P \rightarrow$ probability
- $X_1, X_2, \dots X_n \rightarrow$ random sample of size n taken from whole population.
- $x_1, x_2 \dots x_n \rightarrow$ Values that these random sample (x_i) takes when determining the PDF.
- $\Pi \rightarrow$ Product from 1 to n.

Major steps in MLE

(i) Perform a certain experiment to collect the data.
(ii) choose a parametric model of the data, with certain modifiable parameters.
(iii) Formulate the likelihood as an objective function to be maximized.
(iv) Maximize the objective function & derive the parameters of the model.

Adv &
(i) More consistent
(ii) More efficient
(iii) More Versatile
(iv) flexible

Disadv:- (i) Computationally expensive
(ii) Can be sensitive to choice of optimization algorithm.
(iii) Can be sensitive to choice of probability distribution.

Difference b/w Likelihood & probability?

Likelihood	Probability
(i) Refers to the past events with known outcomes.	Refers to the occurence of future events.
(ii) Likelihood doesn't add up to 1.	Probabilities add up to 1.
(iii) It is conditional model.	Unconditional model.
(iv) It is used in parameter estimation.	used in prediction.
(v) Formula $L(\theta$	Formula $0) = P(0$

Bayesian statistics

It is also known as Bayes rule, Bayes law, Bayesian reasoning.

- It determines the probability of event with uncertain knowledge.
- It is a way to calculate the value of $P(B|A)$ with knowledge of $P(A|B)$.
- It allows updating the probability prediction of an event by observing new info. of real world.

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

$P(A|B)$ → Posterior

$P(B|A)$ → Likelihood

$P(A)$ → Prior

$P(B)$ → Marginal

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

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

$$P(A \cap B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

Applications

(i) It is used to calculate the next step of the robot when the already executed step is given.

(ii) Bayes theorem is helpful in weather forecasting.

(iii) It can solve the Monty hall problem.

Eg

- what is the probability that person has disease dengue with neck pain?

Given :- 80% of time dengue causes neck pain. $P(a|b) = 0.8$

↳ $P(\text{dengue}) = \frac{1}{30,000}$ $P(b) = 1/30,000$

↳ $P(\text{neckpain}) = 0.02$ $P(a) = .02$

Sol

a = Proposition that Person has neck pain

b = Person has dengue.

$P(b|a) = ?$

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

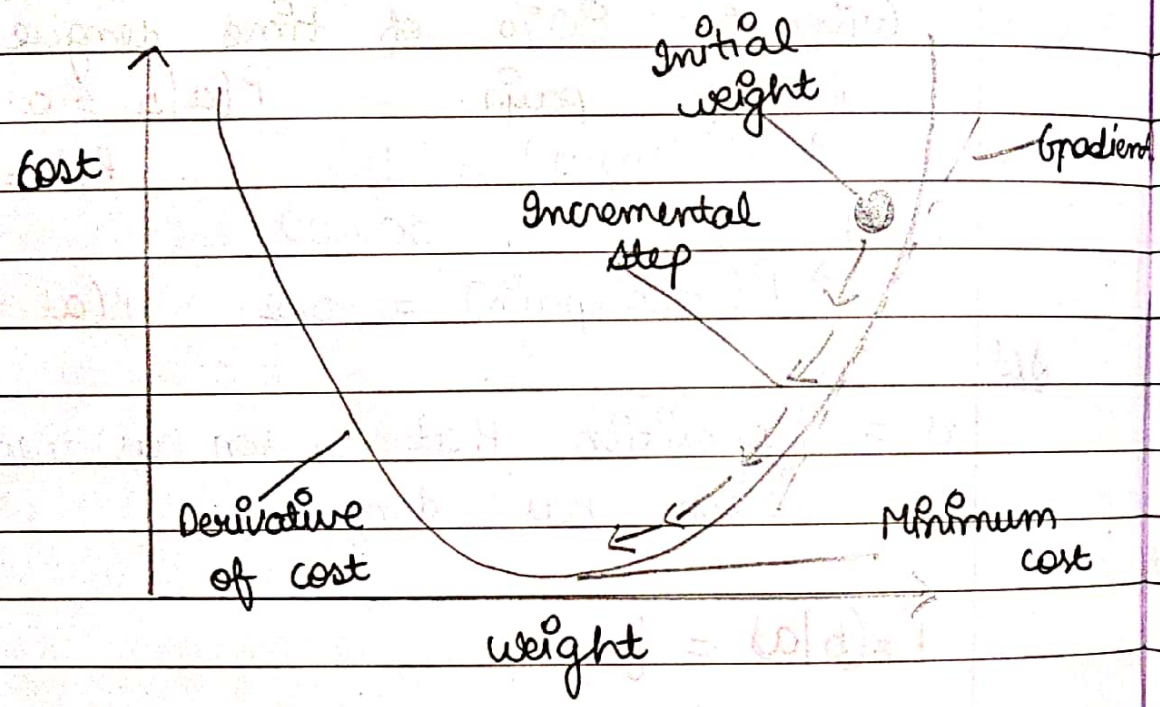
$$= \frac{0.8 \cdot 1/30,000}{0.02}$$

$$= 0.00133 = \text{Ans}$$

Gradient Descent

Gradient Descent is known as one of the most commonly used optimization algorithms to train ML models by means of minimizing the errors b/w actual & expected results.

- It helps in finding the local minimum of a function.



formula

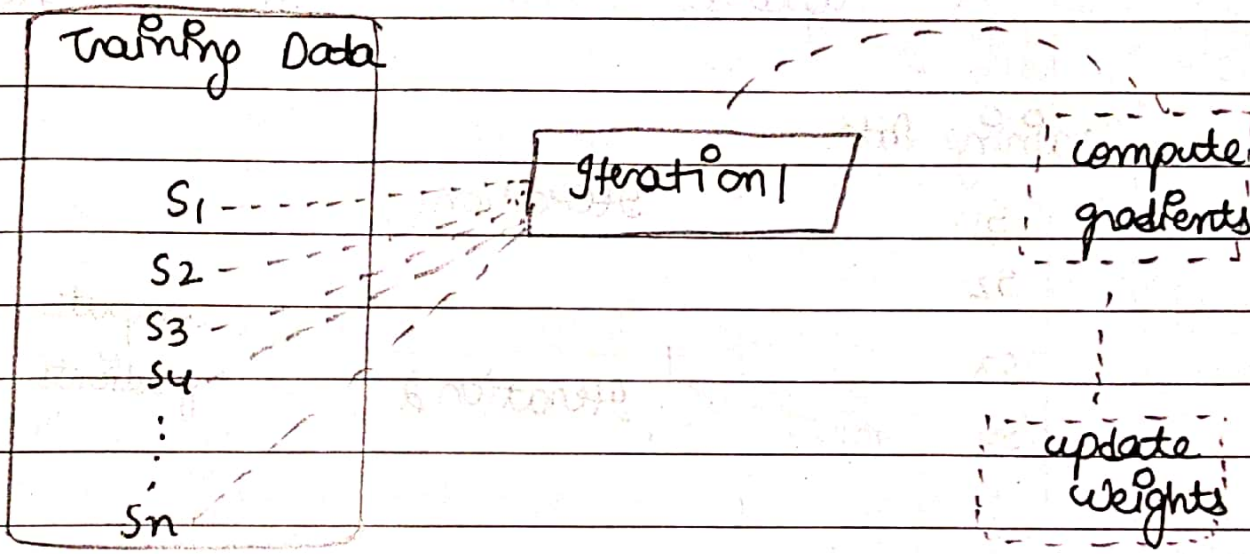
learning rate × slope

new Value = Old Value − step size

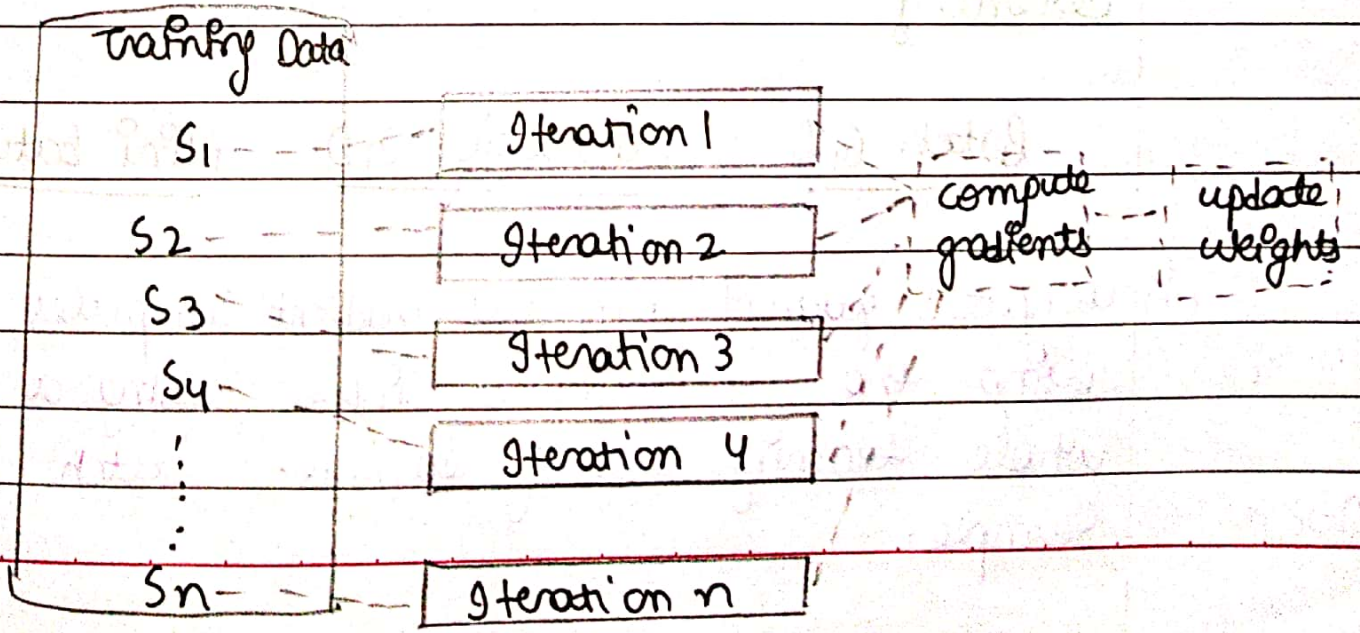
(new guess) (previous guess) (how much you shift)

Types of gradient Descent

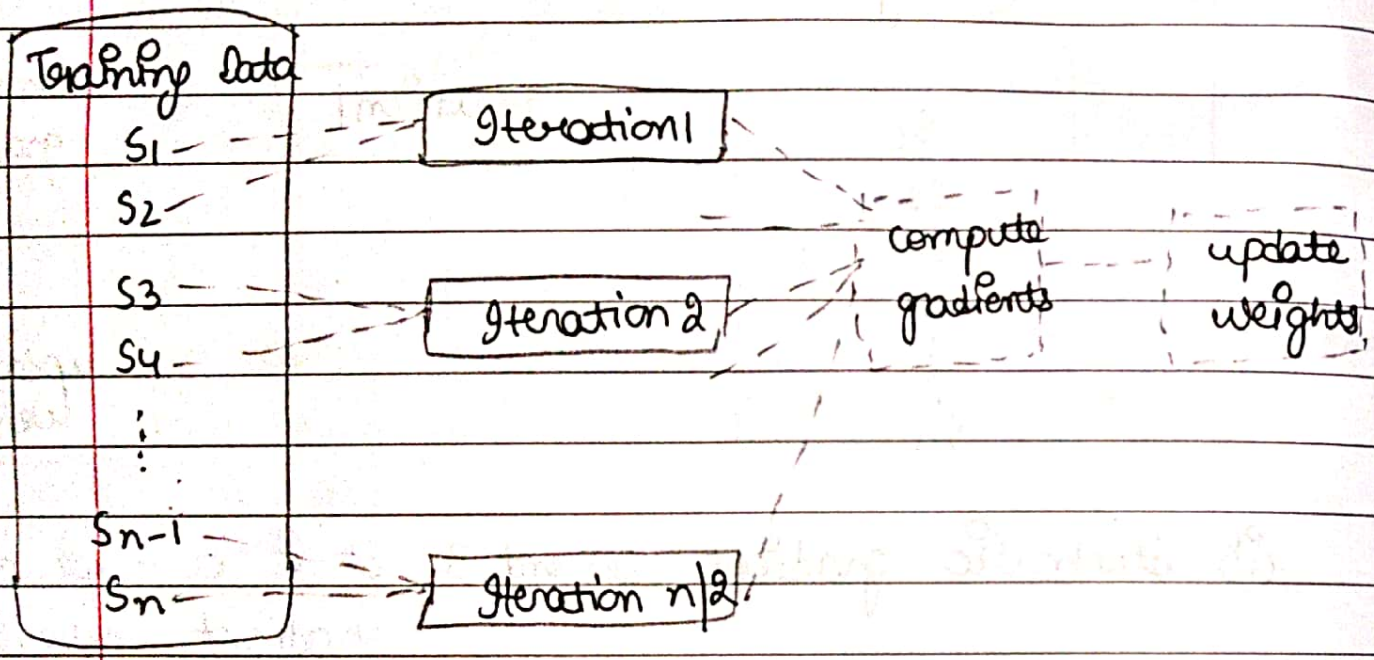
(i) Batch Gradient Descent :- we use all our training data in a single Iteration of algorithm.



(ii) Stochastic gradient Descent :- It is a type of gradient descent that runs one training example per Iteration.



(iii) Mini Batch gradient descent :- It is the combination of both gradient descent & stochastic gradient descent. It divides the training datasets into small batch sizes then performs the updates on those batches separately.



Diff. b/w batch, Stochastic & Mini batch gradient descent ?

Batch GD	Stochastic GD	Mini batch GD
(i) Computes gradient using the whole training Sample.	Computes gradient using a single training sample.	Computes gradient using a small batch sizes.

(i)	Slow & computationally expensive algorithm.	Faster & less computationally expensive than Batch GD.	Faster than BGD, slower than SGD & computation cost is less than BGD.
(ii)	Not suggested for huge training samples.	Can be used for large training samples.	It divides the training datasets into small batch sizes.
(iii)	Deterministic in nature.	Stochastic in nature.	combination of BGD & SGD.
(iv)	Gives optimal solution given sufficient time to converge.	Gives good solution but not optimal.	Not guaranteed to find the optimal solution.
(v)	More accurate.	Less accurate	More accurate than SGD, less accurate than BGD.
(vi)	Cost function reduces smoothly.	lot of variations in cost function.	Smoother cost function as compared as SGD.

(viii)

Adv	Adv	Adv
- It produces less noise in comparison to other GD.	Easier to allocate in desired memory.	Easier to fit in allocated memory.
- Produces Stable GD convergence.	Relatively fast to compute than BGD.	computationally efficient.
- computationally efficient as all resources are used for all training samples.	More efficient for large datasets.	Produces Stable GD convergence.

Q what is cost function?

Ans The cost function is defined as the measurement of difference or error b/w actual values & excepted values at current position & present in the form of a single real number.

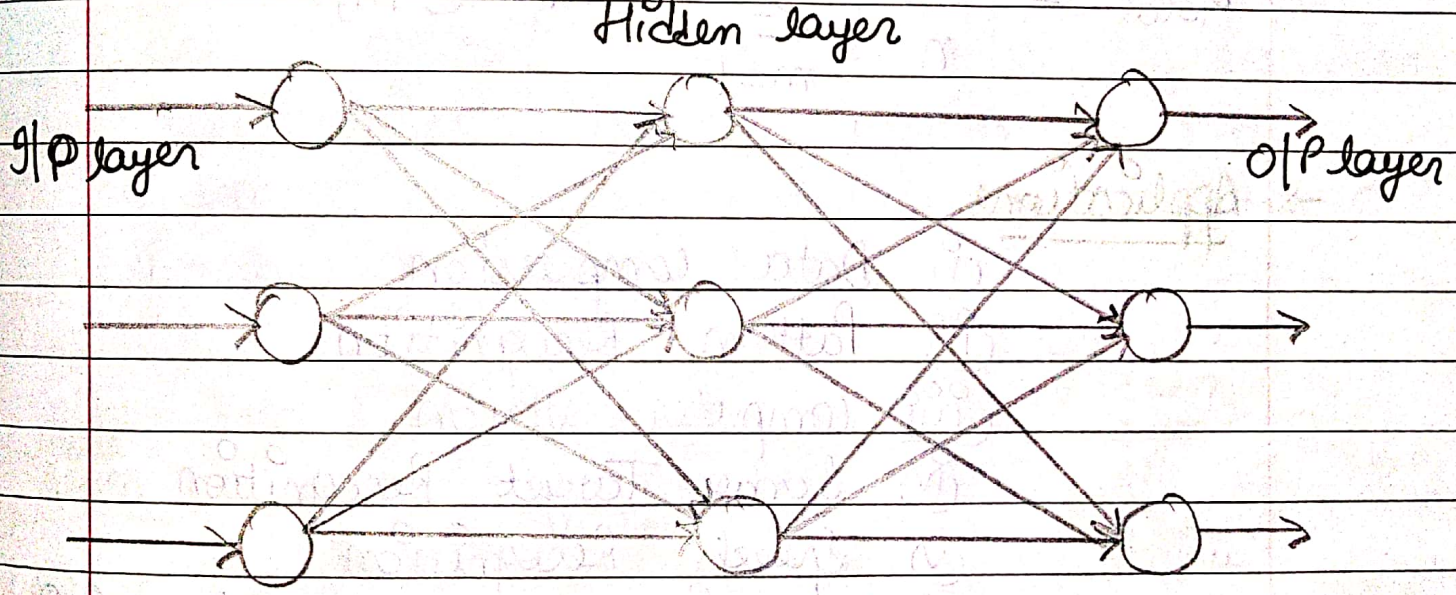
Q Learning rate :- It is defined as the step size taken to reach the minimum or lowest point.

Unit 2nd

DOMS | Page No. 21
Date / /

FFNN

- FFNN stands for "feed forward Neural N/w".
- It is one of simplest forms of ANN.
- A feedforward neural n/w is an ANN where in connections b/w the nodes do not form a cycle.
- Because I/P are processed only in the forward direction.



- FFNN could be a single layer perceptron or multi layer perceptron.
- NO feedback from output to input.
- We don't use it where order of sequence does matter. Like sentence as it predict only for current node, does not depend on previous.
- It has no memory.

- Inputs are Independent to each other.

Mean Squared error (MSE) in FFNN

$$E_{total} = \frac{1}{\text{Total O/P}} \sum (\text{Actual O/P} - \text{Predicted O/P})^2$$

$$E_{total} = \frac{1}{n} \sum_{m=1}^{n} (O_m - \hat{O}_m)^2$$

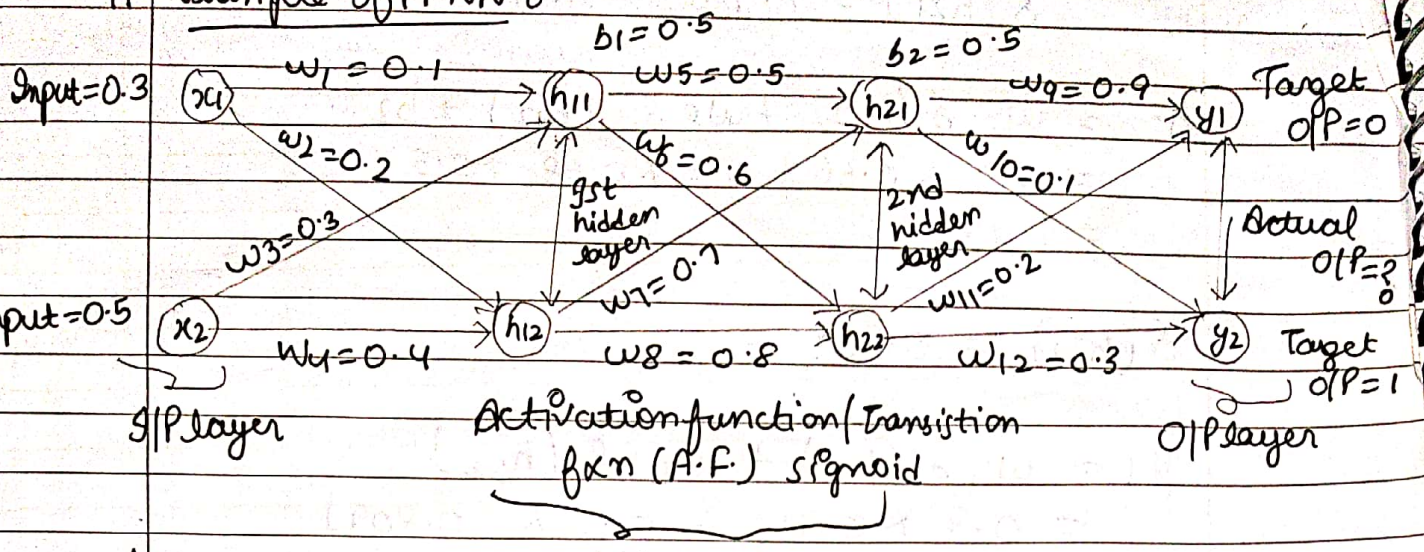
→ **Applications**

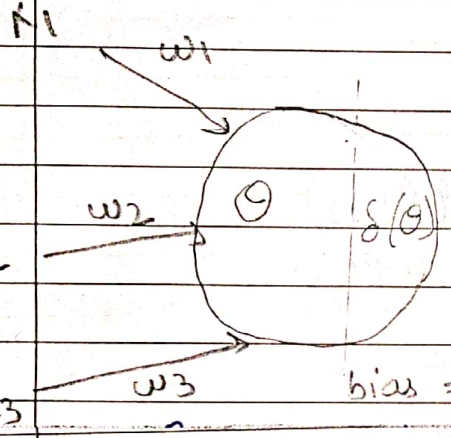
(i) Data compression
(ii) Pattern Recognition
(iii) computer vision
(iv) Sonar Target Recognition
(v) Speech Recognition
(vi) Handwritten characters Recognition.

Adv & (i) simple architecture
(ii) efficient training
(iii) versatility
(iv) Good performance

Disadv & (i) Limited expressiveness
(ii) Prone to Overfitting

Example of FFNN :-





Θ = Overall Input
δ = Activation fn.

$$\Theta = w_1x_1 + w_2x_2 + w_3x_3 + b_1$$

$$\delta(\Theta) = \frac{1}{1+e^{-(w_1x_1 + w_2x_2 + w_3x_3 \dots + b_1)}}$$

- **I/O & O/P of h_{11}**

$$h_{11} = w_1 \cdot x_1 + w_3 \cdot x_2 + b_1$$
$$= 0.1 * 0.3 + 0.3 * 0.5 + 0.5$$
$$= 0.03 + 0.15 + 0.5$$
$$= 0.68$$

$$\delta(h_{11}) = \delta(0.68) = \frac{1}{1+e^{-0.68}}$$
$$= 0.66$$

- Calculate hidden layer h_{12}

$$h_{12} = w_2 x_1 + w_4 \cdot x_2 + b_1$$
$$= 0.2 \times 0.3 + 0.4 \times 0.5 + 0.5$$
$$= 0.06 + 0.2 + 0.5$$
$$= 0.76$$

$\therefore \delta(h_{12}) = 0.68$

- h_{21}

$$h_{21} = w_5 \cdot \delta(h_{11}) + w_7 \cdot \delta(h_{12}) + b_1$$
$$= 0.5 \times 0.66 + 0.7 \times 0.68 + 0.5$$
$$= 0.33 + 0.476 + 0.5$$
$$= 1.306$$

$\therefore \delta(h_{21}) = 0.786$

- h_{22}

$$h_{22} = w_6 \cdot \delta(h_{11}) + w_8 \cdot \delta(h_{12}) + b_1$$
$$= 0.6 \times 0.66 + 0.8 \times 0.76 + 0.5$$
$$= 1.504$$

$\therefore \delta(h_{22}) = 0.818$

- y_1

$$y_1 = w_9 \cdot \delta(h_{21}) + w_{11} \cdot \delta(h_{22})$$
$$= 0.9 \times 0.786 + 0.2 \times 0.818$$
$$= 0.1636$$

$\therefore \delta(y_1) = 0.54$

- y_{22}

$$y_{22} = w_{10}\,\delta(h_{21}) + w_{12}\,\delta(h_{22})$$
$$= 0.1 \times 0.786 + 0.3 \times 0.818$$
$$= 0.324$$

$\therefore \delta(y_2) = 0.58$

→ MSE

$$= \frac{1}{2}\left[(y_A^1 - y_T^1)^2 + (y_A^2 - y_T^2)^2\right]$$

$$= \frac{1}{2}\left[(0.54 - 0)^2 + (0.58 - 1)^2\right]$$

$$= \frac{1}{2}\left[0.2916 + 0.1764\right)$$

$$= \frac{0.468}{2} = 0.234$$

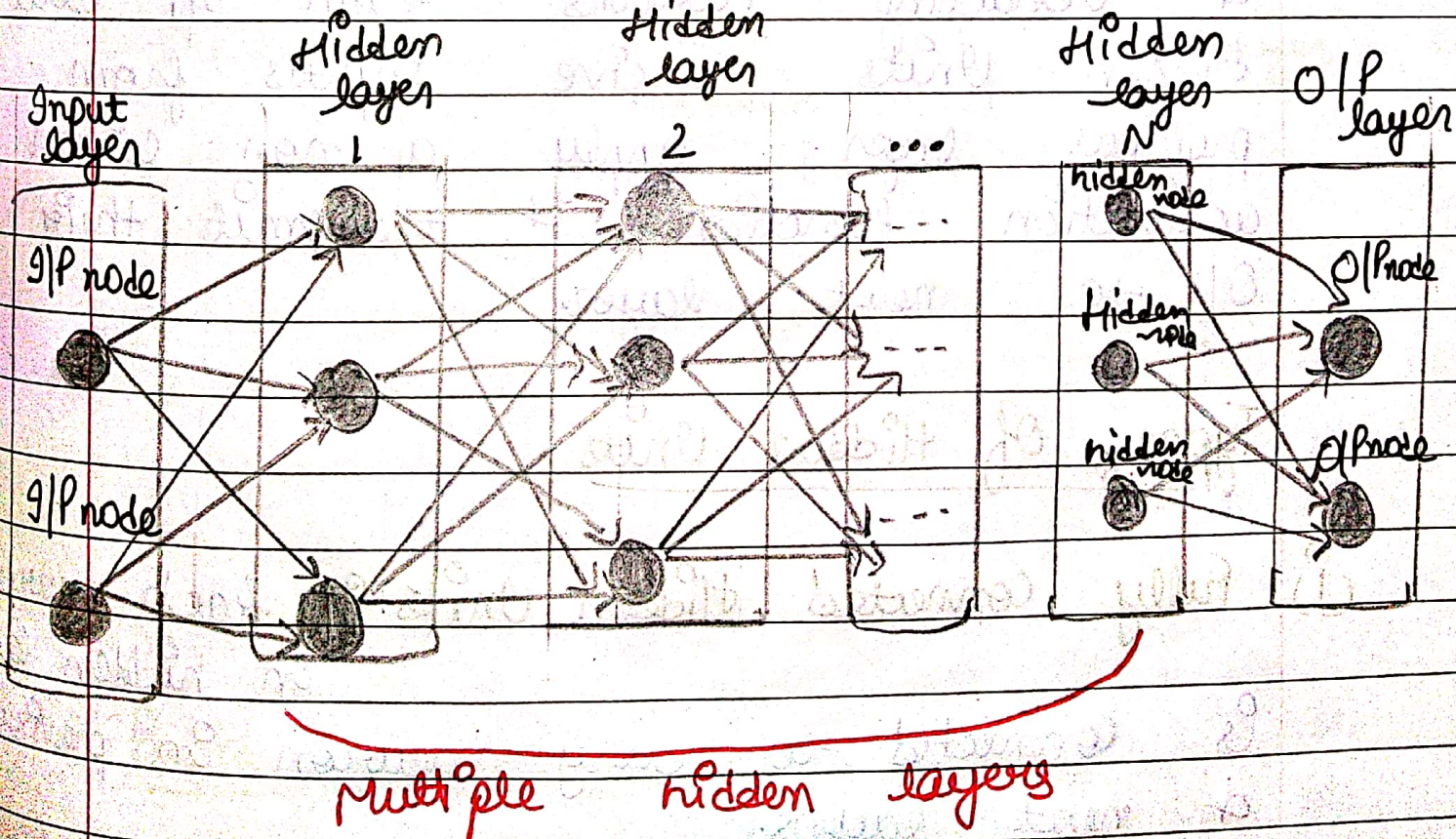
= Any

(iii) Limited ability to handle temporal data.

DFNN

DFNN stands for "Deep feed forward Neural N/w".

- It is also known as multilayer perceptrons (MLP) are composed of multiple layers of interconnected neurons arranged in sequential manner.
- Each neuron receives I/O signals from the previous layer, applies a non linear activation function, & sends O/P signal to next layer.



→ Applications

(i) Image classification
(ii) Natural language processing
(iii) forecasting
(iv) Recommendation systems
(v) Financial Modeling.

Hidden Units

It refer to artificial neurons that constitute the hidden layers of a neural N/w. These units serve as the computational neural N/w, processing info. & extracting patterns from I/P data. Hidden units receive inputs from previous layer, apply a non linear activation function & transmit their O/P to next layer.

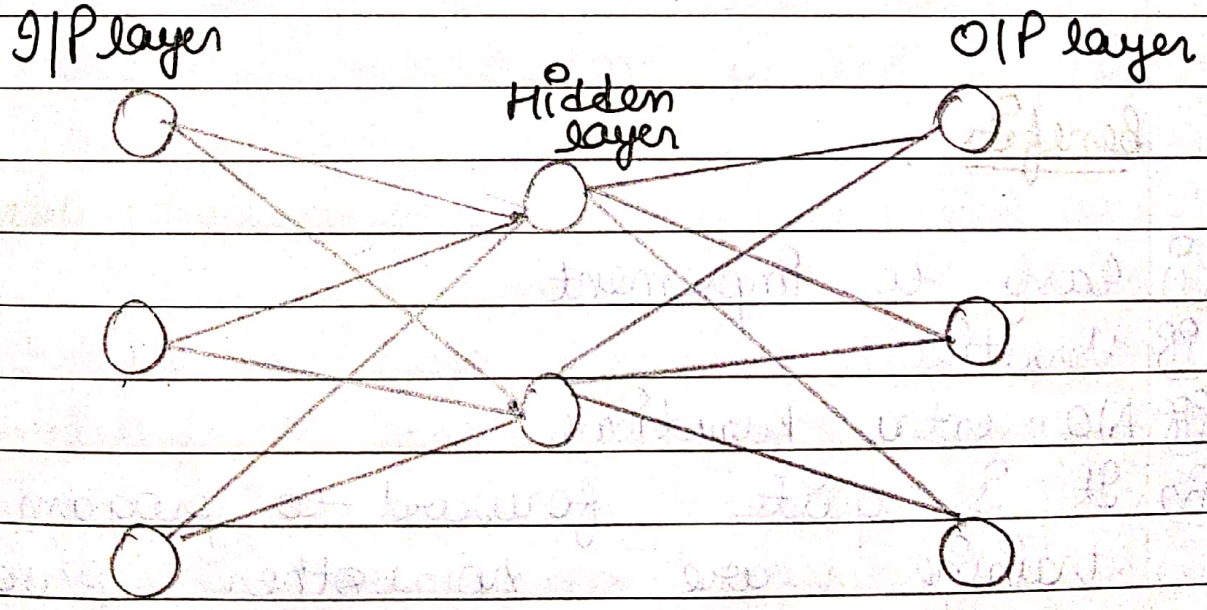
Types Of Hidden Units

(i) Fully connected Hidden Units :- Each neuron in hidden layer is connected to every neuron in previous & next layers.

(ii) Convolutional Hidden Units :- Neurons are connected to local region of I/P or the previous layer, allowing them to extract spatial features in data like images.

(iii) Recurrent hidden units :- Neurons have connections to themselves & previous neurons, enabling them to capture temporal dependencies in sequential data like text or time series.

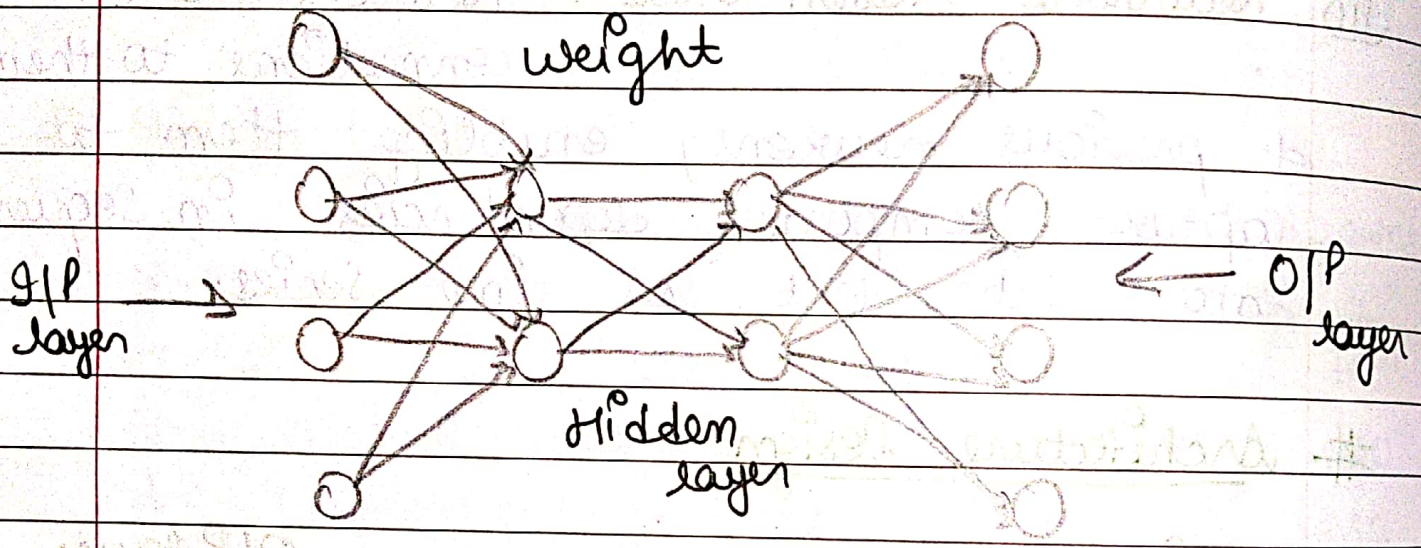
Architecture Design



Back propagation

Back propagation can be called the building block of neural N/w.

- It is an algorithm which is created to test errors which will travel back from I/P nodes to O/P nodes.
- It is similar to Biological N/W that contains neuron coupled with each other.



Benefits

(i) Easy to Implement
(ii) Versatile
(iii) NO extra functions
(iv) It is quite forward to program since training are no other parameters reside the I/O.
(v) flexible.

Applications

(i) Speech recognition
(ii) character "
(iii) Signature "
(iv) Face "

8/9/23 # Computational Graphs :-

- Computational graphs are a type of graph that can be used to represent mathematical expression. This is similar to descriptive language. In the case of DL provided a functional description of the required computation.
- In general, the computational graph is a directed graph that is used for expressing & evaluating mathematical expressions.
- These can be used for two different types of calculations :- a) Forward computation
  b) Backward computation
- A few terminologies in computational graphs are

__/__/___

as follows:-

1) A variable is representing by a node in a graph. It could be scalar, vector, matrix, tensor or in another type of variable.

2) Function Argument & data dependancy are both represented by an edge. These are similar to node pointers.

3) A simple function of one or more variables is called an operation. There is a set of operations that are permitted functions that are more complex than these operations on this set can be represented by combining multiple operations.

$$y = (a+b) * (b-c)$$

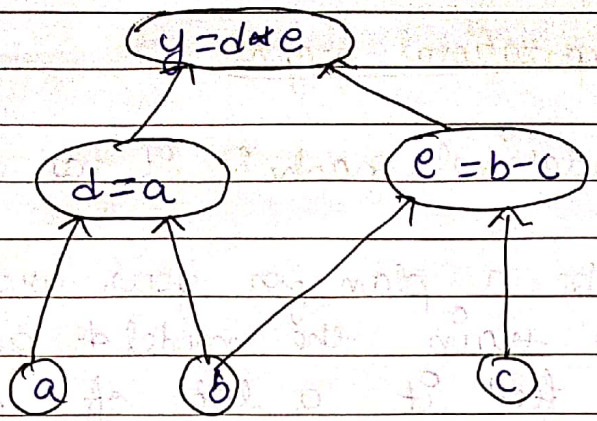
we introduce the two variables d & e, such that every operation has an O/P variable

$$d = a+b \quad \text{(addition)}$$
$$e = b-c \quad \text{(Sub.)}$$
$$y = d * e \quad \text{(mult)}$$

Now, we have three operations: add, sub, mult
To create a computational graph we create nodes each of them has diff. operations along with I/P variables.

The direction of the array shows:-
The direction of I/P being applied the other nodes.

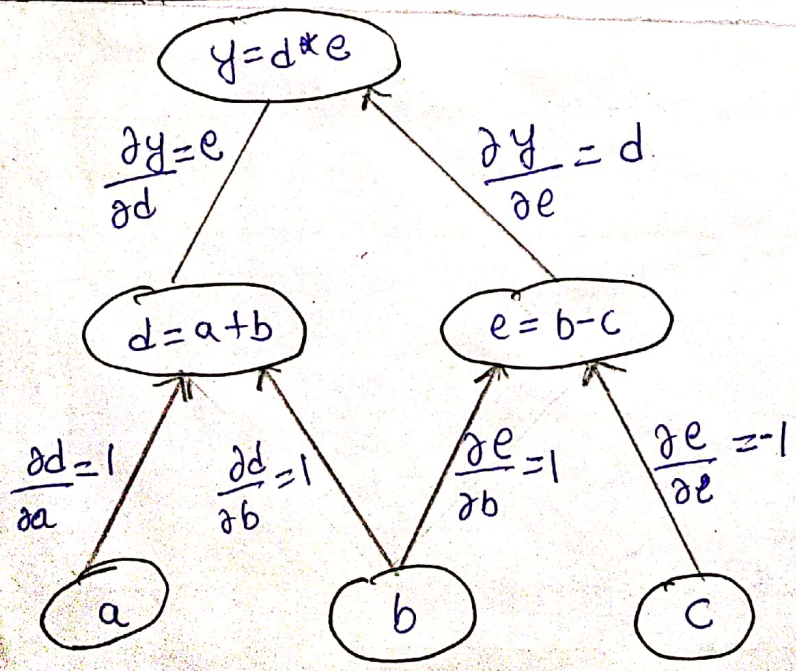
__/__/____



- We can find the final O/P value by initializing I/P variables & accordingly computing nodes of the graphs

Computational Graph in DL's-

- Computations of the neural Networks are organized in terms of forward Pass / forward Propagation. Step in which the re-compute the O/P of neural N/w followed by a backward Pass / Propagation.
Step which we used to compute gradients / derivatives.



Chain rule to evaluate derivatives

$$\frac{\partial y}{\partial a} = \frac{\partial y}{\partial d} \times \frac{\partial d}{\partial a} = e \times 1 = e$$

$$\frac{\partial y}{\partial b} = \frac{\partial y}{\partial d} \times \frac{\partial d}{\partial b} = e \times 1 = e$$

$$\frac{\partial y}{\partial c} = \frac{\partial y}{\partial e} \times \frac{\partial e}{\partial c} = d \times -1 = -d$$

__/__/____

2/9/23

Types of computational graphs

(1) Static computational graph :- It has also two phases :-

Phase 1 :- Make a plan for your architecture
Phase 2 :- To train the model & generate predictions, & feed it a lot of data.

- The benefit of utilising this graph is that it enables powerful offline graph, optimization & scheduling.
- Adv As a result they should be faster than dynamic graphs.
- The drawback is that dealing with structured & even variable sized data is unsightly

(2) Dynamic computational graphs :- As the forward computation is performed the graph is implicitly defined.

- This graph as the advantage of being more adaptable.
- The library is less intrusive & enables interleaved graph generation & evaluation
- The forward computation is implemented in your prefer programming language complete with all of his features & algos. & debugging dynamic graphs is simple. because it permits line by line execution of code & access to all variables
- The disadvantage of implementing this graph is that there is limited time for graph & effort for the wasted if the graph does not change.

Back Propagation

- It is a process involve in training a neural network. It involve taking an error rate of a forward propagation & feeding this loss backward through the neural N/w. pair layers to fine tune the weights.
- It is the essence of neural N/w training. It is the weights of neural N/w based on the error rate obtain in the previous iteration proper training of the weights ensures lower error rates making the model reliable by increasing the generalization.

Advantages :-

(i) No previous knowledge of a neural N/w is needed taking it easy to implement.
(ii) It is quite forward to program since training are no other parameters reside the I/P.
(iii) It doesn't need to learn the feature of

__/__/____

function, speeding up the process.

(iv) The model is flexible because of its simplicity & applicable to many scenarios.

Limitations :-

(i) Training data can impact the performance of model, so high quality data is essential.

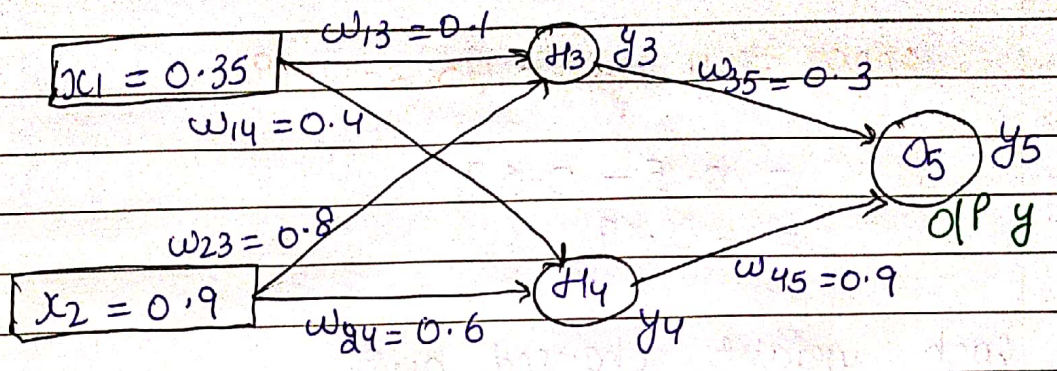
(ii) Noisy data can also effect back propagation, potentially its results.

(iii) It can take a while to train back propagation models & give them up to speed.

(iv) Back propagation requires matrix based approach which can lead to other issues.

Back Propagation Example

1) Assume that the neurons have a Sigmoid activation function, perform a forward pass & a backward Pass on the N/W. Assume that the actual O/P of Y is 0.5 & learning rate is 1. Perform another forward Pass.



Forward Pass: Compute O/P for Y3, Y4 & Y5

$$a_j = \sum_j (w_{i,j} * x_i) \qquad y_j = F(a_j) = \frac{1}{1+e^{-a_j}}$$

→ $a_1 = (w_{13} * x_1) + (w_{23} * x_2)$
$= (0.1 * 0.35) + (0.8 * 0.9)$
$= 0.755$

$y_3 = f(a_1) = 1/(1+e^{-0.755}) = 0.68$

→ $a_2 = (w_{14} * x_1) + (w_{24} * x_2)$
$= (0.4 \times 0.35) + (0.6 \times 0.9)$
$= 0.68$

$y_4 = f(a_2) = 1/(1+e^{-0.68}) = 0.6637$

→ $a_3 = (w_{35} \times y_3) + (w_{45} \times y_4)$
$= (0.3 \times 0.68) + (0.9 \times 0.6637)$
$= 0.801$

$y_5 = f(a_3) = 1/(1+e^{-0.801})$
$= 0.69$

__/__/____

error = $y_{target} - y_5 = -0.19$

$= 0.5 - 0.69$

$= -0.19$

→ Each weight changed by δ

$$\begin{cases} \Delta w_{ji} = \eta \delta_j o_i \\ \delta_j = o_j (1 - o_j)(t_j - o_j) \\ \delta_j = o_j (1 - o_j) \sum_k \delta_k w_{kj} \end{cases} \quad \begin{matrix} \\ j \text{ is O/P unit} \\ j \text{ is hidden unit} \end{matrix}$$

- **Backward Pass & Compute δ3, δ4 & δ5**

→ **For O/P unit**

$\delta_5 = y(1-y)(y_{target} - y)$

$= 0.69 * (1 - 0.69) * (0.5 - 0.69)$

$= -0.0406$

→ **For hidden unit**

$\delta_3 = y_3 (1 - y_3) w_{35} * \delta_5$

$= 0.68 * (1 - 0.68) * (0.3 * -0.0406)$

$= -0.00265$

$\delta_4 = y_4 (1 - y_4) w_{45} * \delta_5$

$= 0.6637 * (1 - 0.6637) * (0.9 * -0.0406)$

$= -0.0082$

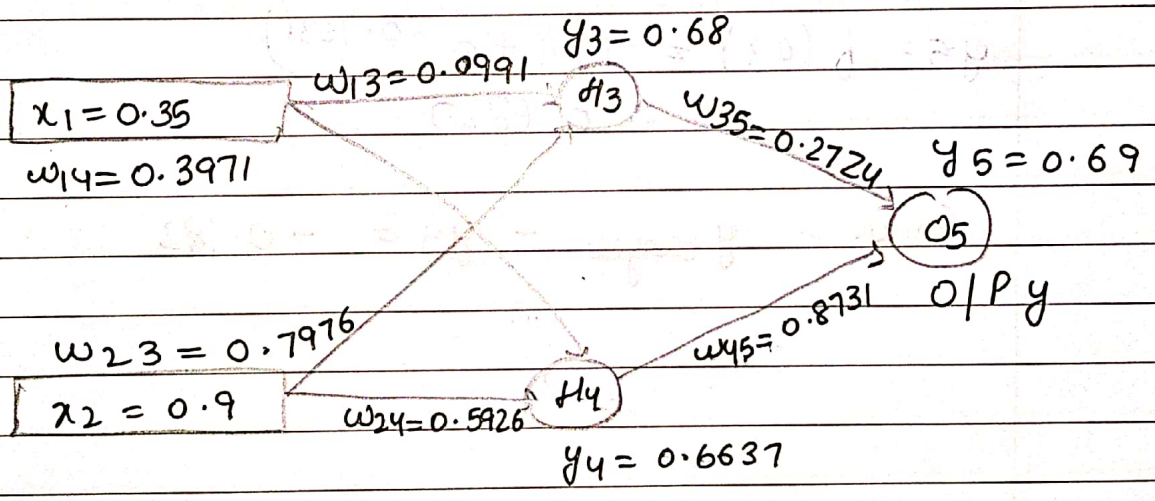
__/__/____

- $\Delta w_{45} = \eta \delta_5 y_4 = 1 * -0.0406 * 0.6637 \Rightarrow -0.0269$

$\Delta w_{45}(new) = \Delta w_{45} + w_{45}(old)$
$= -0.0269 + (0.9)$
$= 0.8731$

- $\Delta w_{14} = \eta \delta_4 x_1$
$= 1 * -0.0082 * 0.35$
$= -0.00287$

$w_{14}(new) = \Delta w_{14} + w_{14}(old)$
$= -0.00287 + 0.4$
$= 0.3971$



Forward Pass & Compute O/P for y_3, y_4 & y_5

→ $a_j = \sum_i (w_{ij} * x_i) \quad y_j = F(a_j) = \frac{1}{1+e^{-a_j}}$

→ $a_1 = (w_{13} * x_1) + (w_{23} * x_2)$
$= (0.0991 * 0.35) + (0.7976 * 0.9)$
$= 0.7525$

__/__/____

$$y_3 = f(a_1) = 1/(1+e^{-0.1525})$$
$$= 0.6797$$

→ $a_2 = (w_{14} * x_1) + (w_{24} * x_2)$
$= (0.3971 * 0.35) + (0.5926 * 0.9)$
$= 0.6723$

$$y_4 = f(a_2) = 1/(1+e^{-0.6723})$$
$$= 0.6620$$

→ $a_3 = (w_{35} * y_3) + (w_{45} * y_4)$
$= (0.2724 * 0.6797) + (0.8731 * 0.6620)$
$= 0.7631$

$$y_5 = f(a_3) = 1/(1+e^{-0.7631})$$
$$= 0.6820$$

Error = $y_{target} - y_5 = -0.182$

Regularization

Regularization is a technique used to reduce errors by fitting the function appropriately on the given training set & avoiding overfitting. The commonly used regularization techniques are :-

(i) Lasso regularization - L1 Regularization
(ii) Ridge regularization - L2 Regularization
(iii) Elastic Net regularization - L1 and L2 regularization

(i) Lasso Regression :- A regression model which uses the L1 regularization technique is called LASSO (Least absolute shrinkage & selection operator) regression.

- It can help us to reduce the overfitting in the model as well as feature selection.

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

where,
m - No. of features
n - No. of examples
y_i - Actual target value
y_i(hat) - Predicted target value.

(ii) Ridge regression :- A regression model that uses the L2 regularization.
- It is used to reduce the complexity of model by shrinking the coefficients.

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

(iii) Elastic Net Regression :- This model is a combination of L1 as well as L2 regularization. That implies that we add the absolute norm of the weights as well as the squared measure of weights.

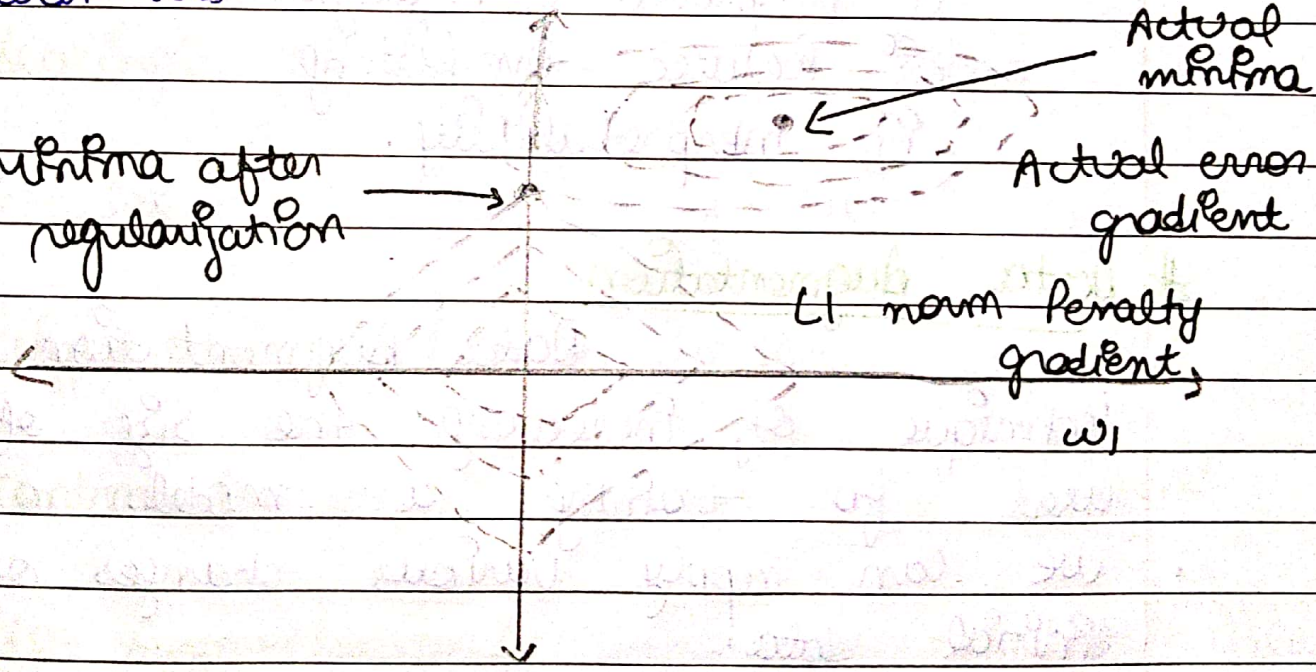
$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\left((1-\alpha)\sum_{i=1}^{m}|w_i| + \alpha\sum_{i=1}^{m} w_i^2\right)$$

Parameter Penalties

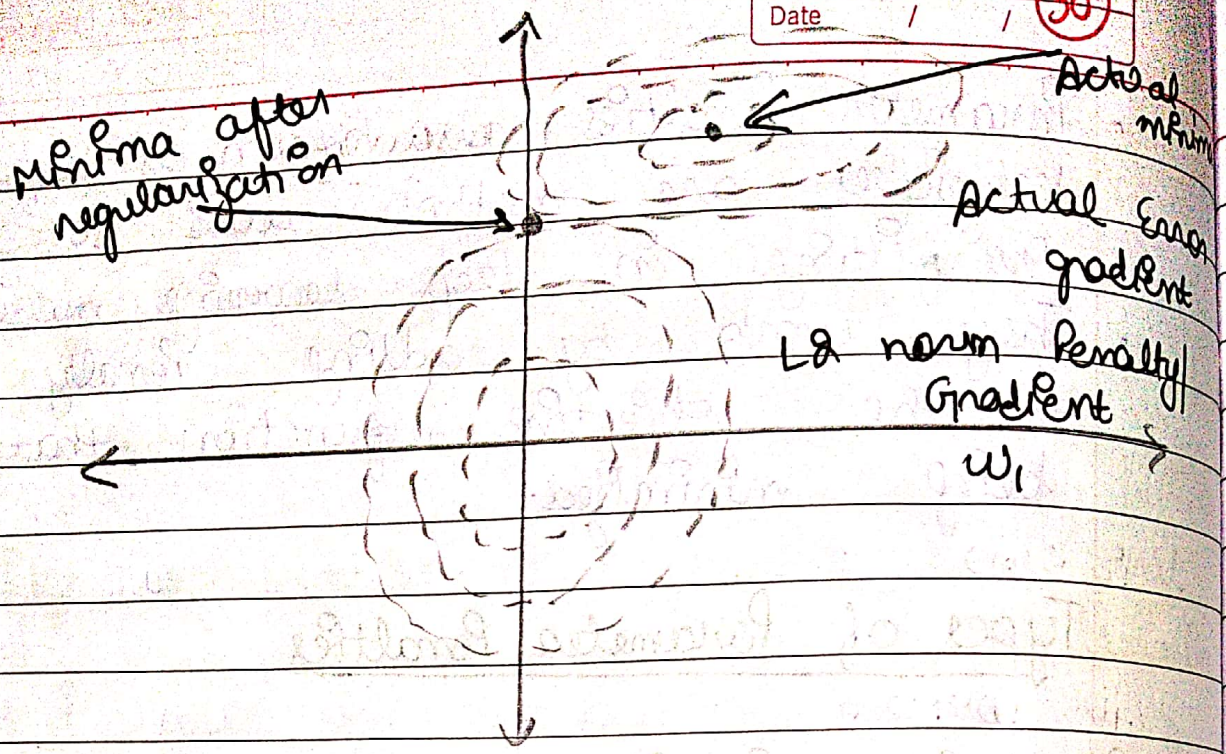
- Parameter Penalties, also known as regularization techniques, are methods used to prevent Overfitting in deep learning models.
- They work by adding Penalty term to the objective function that is being minimized.

Types of Parameter Penalties

(1) L1 Regularization :- L1 norm is sum of the absolute values of model Parameters.



(2) L2 regularization :- L2 norm is the sum of the squares of the model parameters.



Benefits

(i) Improved Generalization
(ii) Reduced Overfitting
(iii) Interpretability.

Data Augmentation

Data Augmentation is a technique of increasing the size of data used for training a model.

- we can apply various changes of initial data.

Eg

for images, we can use - Geometric transformations like, randomly flip, crop, rotate etc.

Types of Data Augmentation

(i) Adding noise :- By "Salt & pepper noise", the images looks like consisting of white & black dots.

(ii) Cropping :- resized original image size.

(iii) Flipping :- Horizontally & vertically flip.

(iv) Rotation :- Rotate degree b/w 0° to 360° degree.

(v) Scaling :- Scaled outward & inward.

(vi) Translation :- x-axis & y-axis.

(vii) Brightness :- Darker or lighter.

(viii) Contrast :- colour aspects.

(ix) Color Augmentation :- changed by new Pixel values

(x) saturation :- depth or intensity of color.

Applications

(i) Image classification
(ii) Speech recognition
(iii) Medical Imaging.

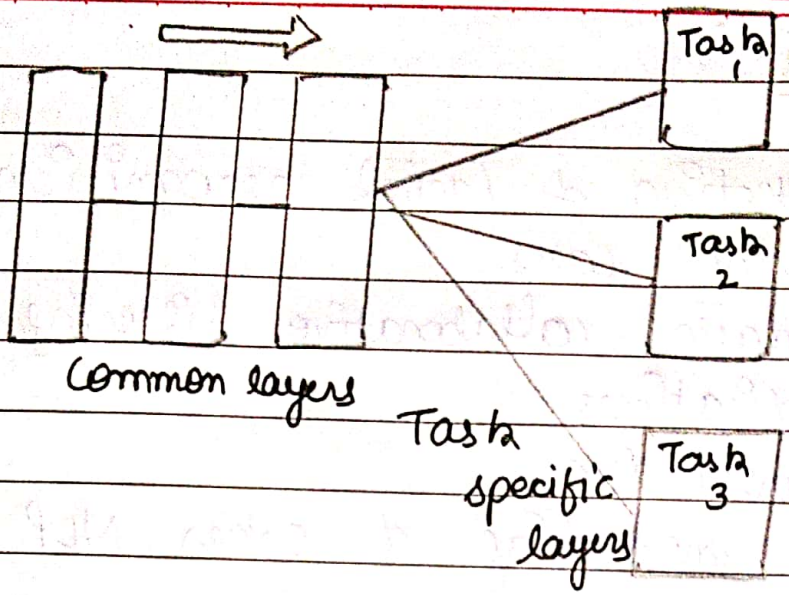
Multi task Learning

MTL is a type of ML technique where a model is trained to perform multiple tasks simultaneously.

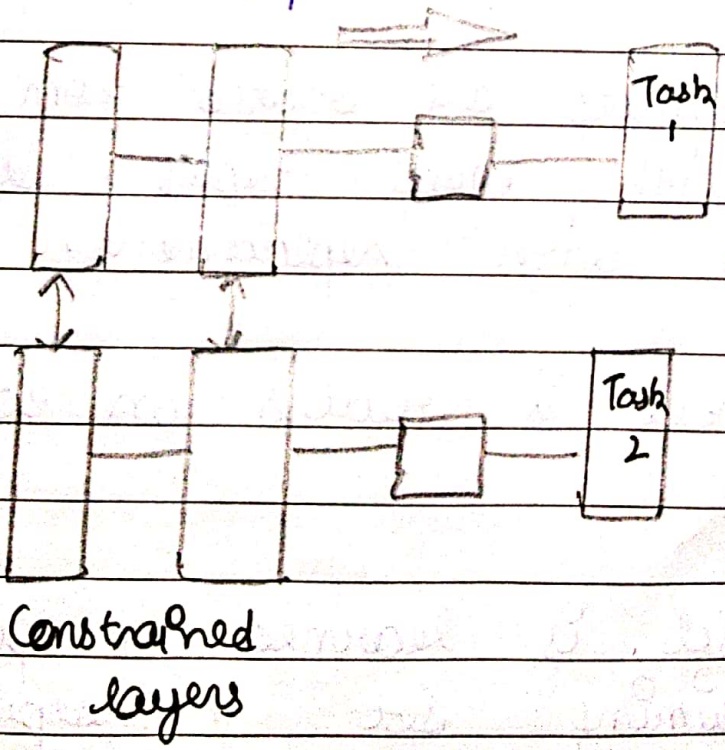
- It refers to training a neural N/W to perform multiple tasks by sharing some of networks layer & parameters across tasks.
- The goal is to improve the generalization performance of model by leveraging info. shared across tasks.

Techniques to use MTL

(i) Hard Parameter sharing :- A common hidden layer is used for all tasks but several task specific layers are kept intact towards end of model. This technique is very useful as learning a representation of various tasks by common hidden layer, we reduce the risk of overfitting.



(ii) Soft Parameter sharing :- Each model has their own sets of weights & biases & the distance b/w these parameters in diff. models is regularized so that the parameters become similar & can represent all the tasks.

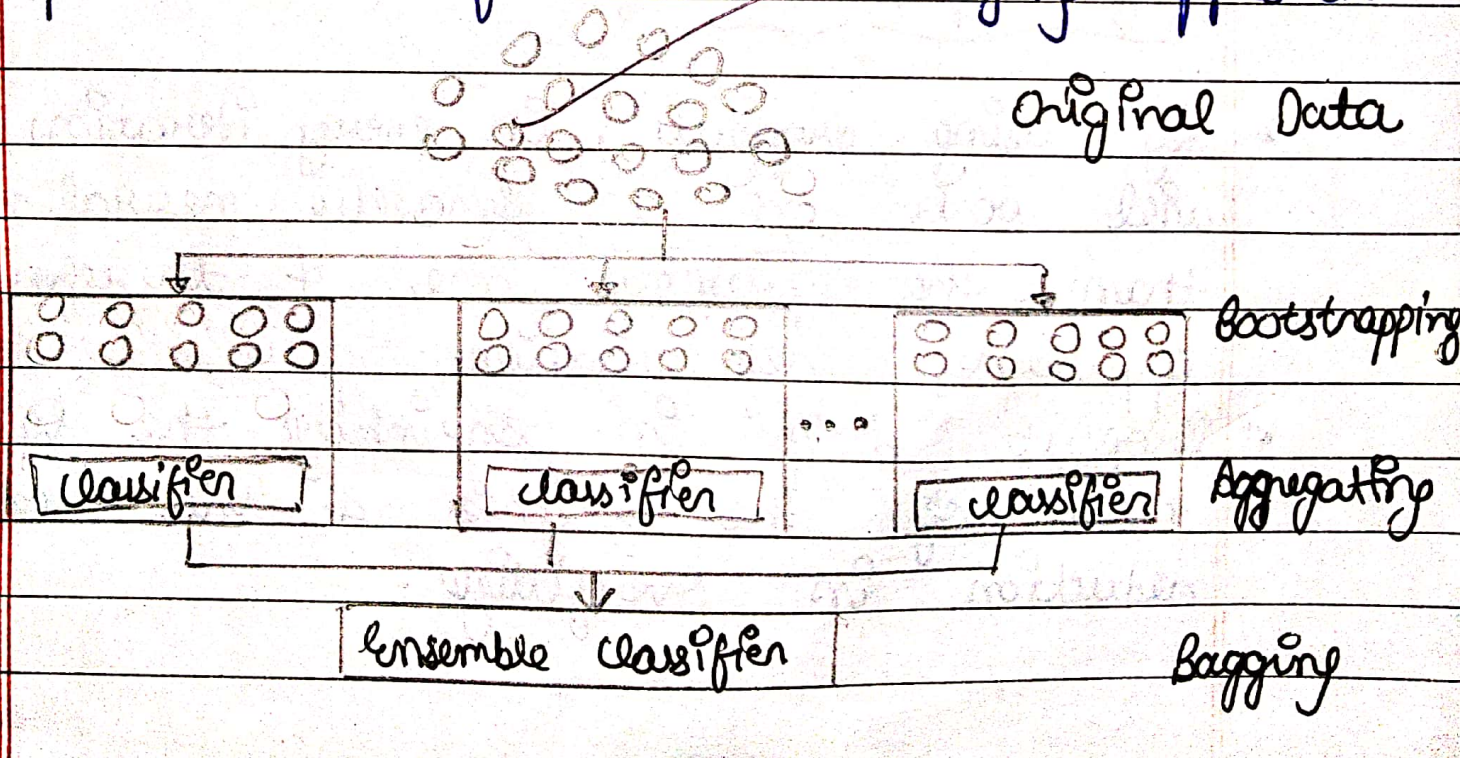


Applications

(i) Object detection & facial recognition
(ii) Self driving cars
(iii) Multi° domain collaborative filtering for web applications.
(iv) stocks prediction
(v) Language modelling & other NLP applications.

Bagging

It is also known as Bootstrap aggregating, is a machine learning ensemble meta-algorithm designed to improve the stability & accuracy of ML algorithms used in statistical classification & regression. It decreases the variance & helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of model averaging approach.



Bagging

eg Random forest model.

Implementation steps of Bagging

Step 1 Multiple subsets are created from the original dataset with equal tuples, selecting observations with replacement.

Step 2 A base model is created on each of these subsets.

Step 3 Each model is learned in parallel with each training set & independent of each other.

Step 4 The final predictions are determined by combining the predictions from all the models.

Advantages

(i) Minimizes the Overfitting of data.
(ii) Improves model accuracy
(iii) It deals with higher dimensional data efficiently.

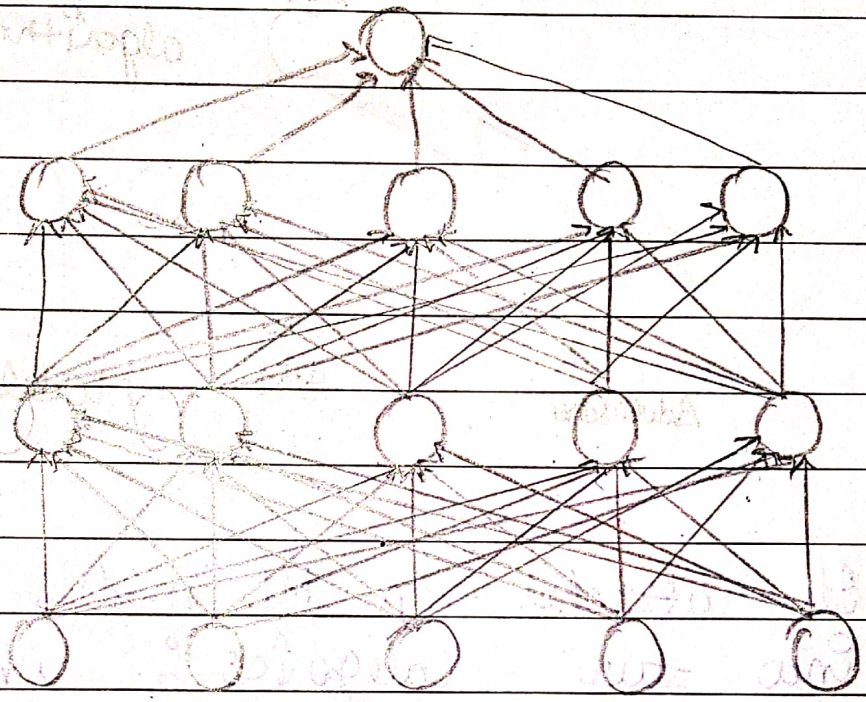
Dropout

Dropout refers to the pratice of disregarding certain nodes in a layer at random during training. A dropout is a regularization approach that prevents overfitting by ensuring that no units are codependent with one another.

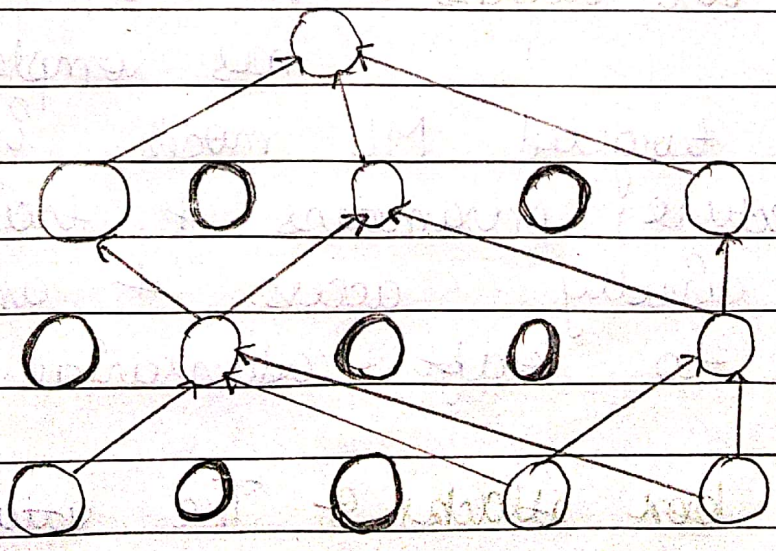
why dropout works?

- By using dropout, in every iteration, you will work on a smaller neural network than the previous one & therefore, it approaches regularization.
- Dropout helps in shrinking the squared norm of the weights & this tends to a reduction in Overfitting.

Dropout



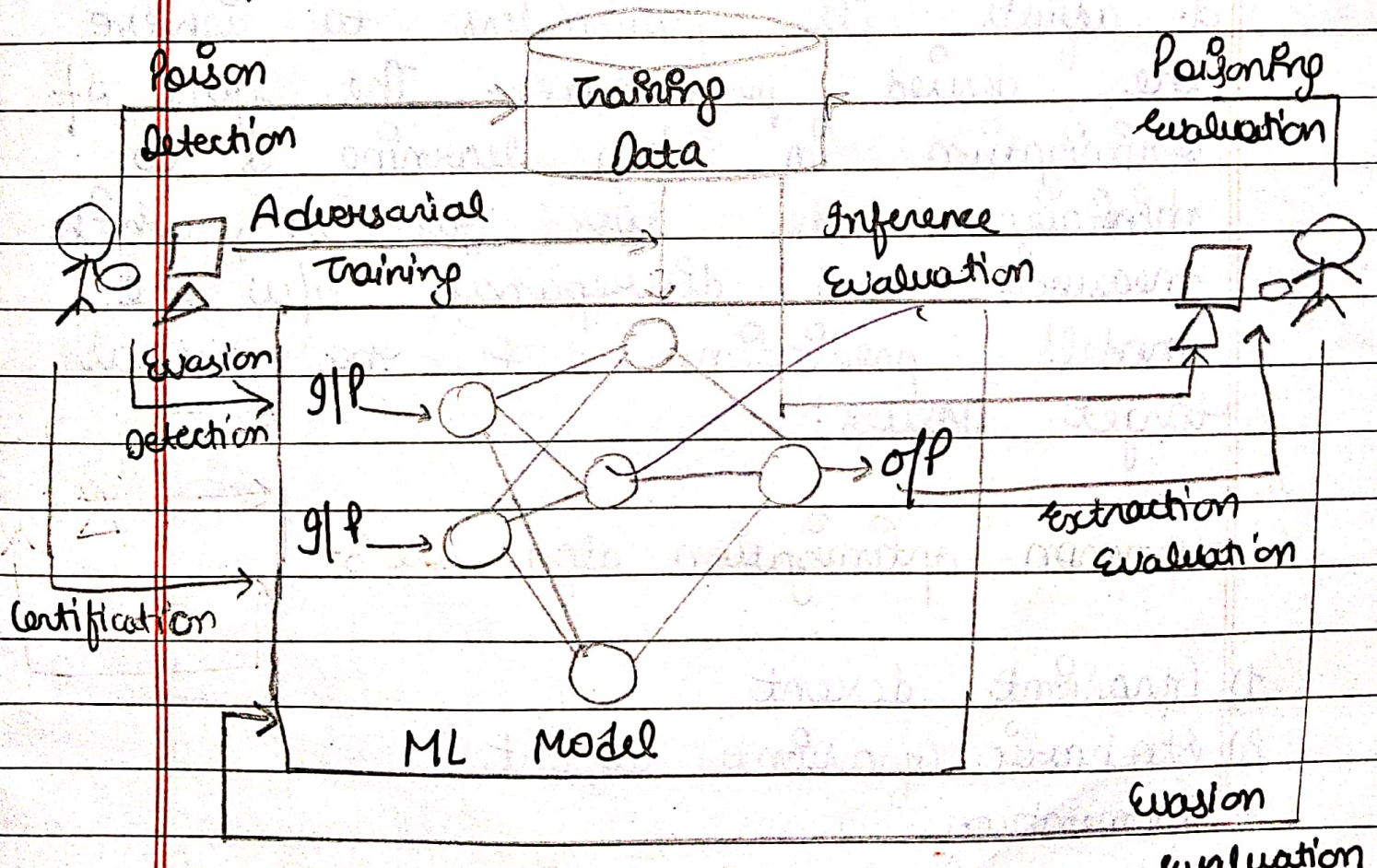
(a) Standard Neural Net



(b) After applying dropout

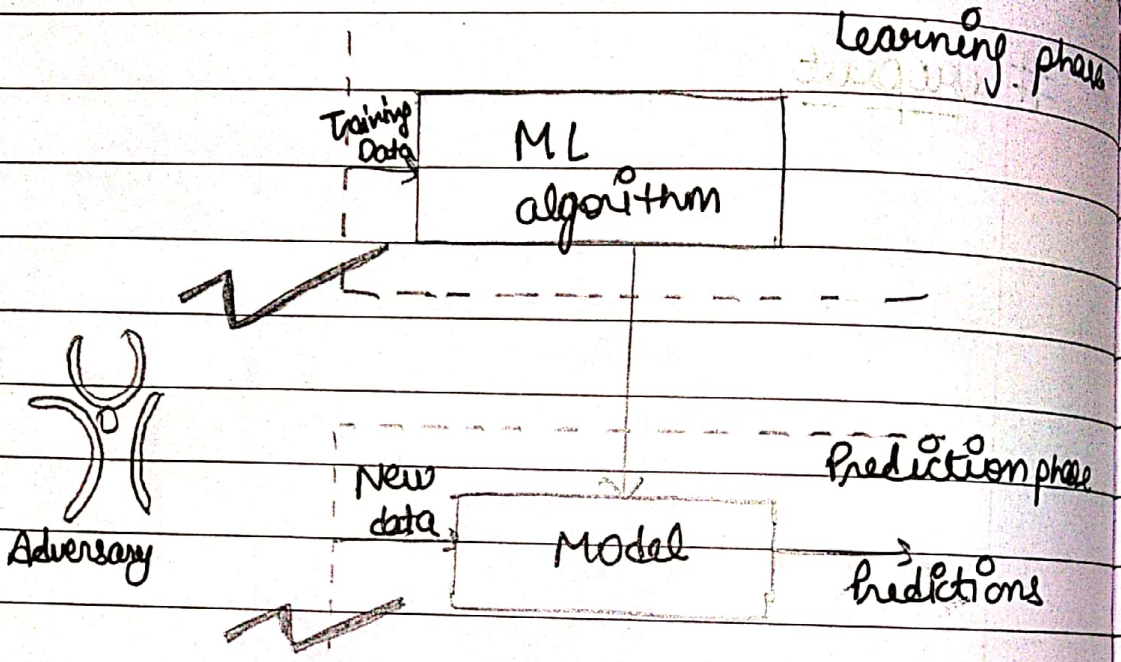
Adversarial training

It is a process where examples adversarial instances are introduced to the model & labeled as threatening. This process can be useful in preventing further adversarial ML attacks from occuring, but require large amounts of maintenance.



Adversarial training

Adversarial Attacks on ML models



Its attacks on ML models can be classified into two categories & white box & black box attacks.

(i) white box attacks :- In this, the attacker has complete knowledge of a targeted ML model, including its architecture, parameters & training data. They can directly access & analyze the model to craft adversarial examples.

(ii) Black box attacks :- The attacker has limited or no knowledge

of targeted model's internal details. They can only query the model with I/P & observe the corresponding O/P.

Eg
- Image classification
- NLP (Natural language processing)
- Speech recognition
- self driving cars

Optimization

Optimization is a crucial aspect of deep learning, as it determines how effectively a model learns from data & adjusts its parameters to achieve the desired performance. The goal of optimization in deep learning is to minimize the loss function, which measures the discrepancy b/w the model's predictions & the actual target values.

Common optimization algo. are :-

1) Gradient descent
2) Stochastic Gradient descent
3) Momentum
4) Nesterov Accelerated gradient
5) Adaptive Gradient Methods

Unit 3rd

DOMS | Page No. 38
Date / /

Convolution

It is a mathematical operation that is applied in a variety of fields, such as image processing, audio & signal processing tasks to extract useful features from I/P data by applying various filters (also known as kernels).

1) What is a Kernel?

Ans A Kernel is a CNN is a small matrix that is used while performing convolution of I/P data. It is known as filter or weight. Depending on the size of I/P data & the required level of granularity for extracted features, the kernel shape is chosen. Generally, it is a small matrix like 3x3, 5x5, or 7x7.

2) What is stride?

Ans Stride is the no. of Pixels or units that a kernel is moved across the I/P data while performing convolution operations in CNN. It is one of hyperparameters of a CNN that can be manipulated to control the O/P feature map's size.

3) What is Padding?

Ans Padding is a technique used in CNN to preserve the spatial dimensions of the I/P data & prevent the loss of info. of at the edges of image, it is done by adding additional layers of zeroes around the edges of I/P matrix.

Types of Padding

Same padding Valid padding

1) Same padding:- It describes the process of adding padding to an image or feature map such that the O/P has the same spatial dimensions as I/P. The same padding adds additional rows & columns of pixels around the edges of I/P data so that the size of O/P feature map will be same as the size of I/P data.

2) Valid Padding:- Valid padding is used when it is desired to reduce the size of O/P feature map in order

to reduce the no. of parameters in the model & improve the computational efficiency.

Pooling

The pooling operation involves sliding a two dimensional filter over each channel of feature map & summarising the features lying within region covered by filter.

$$(n_h - f+1)/S \times (nw - f+1)/S \times n_c$$

Where,

- n_h → height of feature map
- n_w → width " " "
- n_c → no. of channels in feature map
- f → size of filter
- S → stride length.

why to use pooling layers?

- Pooling layers are used to reduce the dimensions of feature maps. Thus, it reduces the no. of parameters to learn

of the amount of computation performed in the NW.

- It summaries the features present in the region of feature map generated by a convolution layer.

Types of Pooling layers

(i) Max Pooling :- It is a pooling operation that selects the maximum element from the region of feature map covered by the filter. Thus, the O/P after max pooling layer would be a feature map containing the most prominent features of previous feature map.

2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

Max Pool
filter - (2×2)
stride - (2,2)

9	7
8	6

(ii) Average Pooling :- It computes the average of the elements present in region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch

of feature map, average pooling gives the average of features present in a patch.

2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

Average pool
Filter - (2×2)
Stride - (2,2)

4.25	4.25
4.25	3.5

iii) Global Pooling :- It reduces each channel in feature map to a single value. Thus, an $n_h \times n_w \times n_c$ feature map is reduced to $1 \times 1 \times n_c$ feature map.

Advantages of Pooling layer

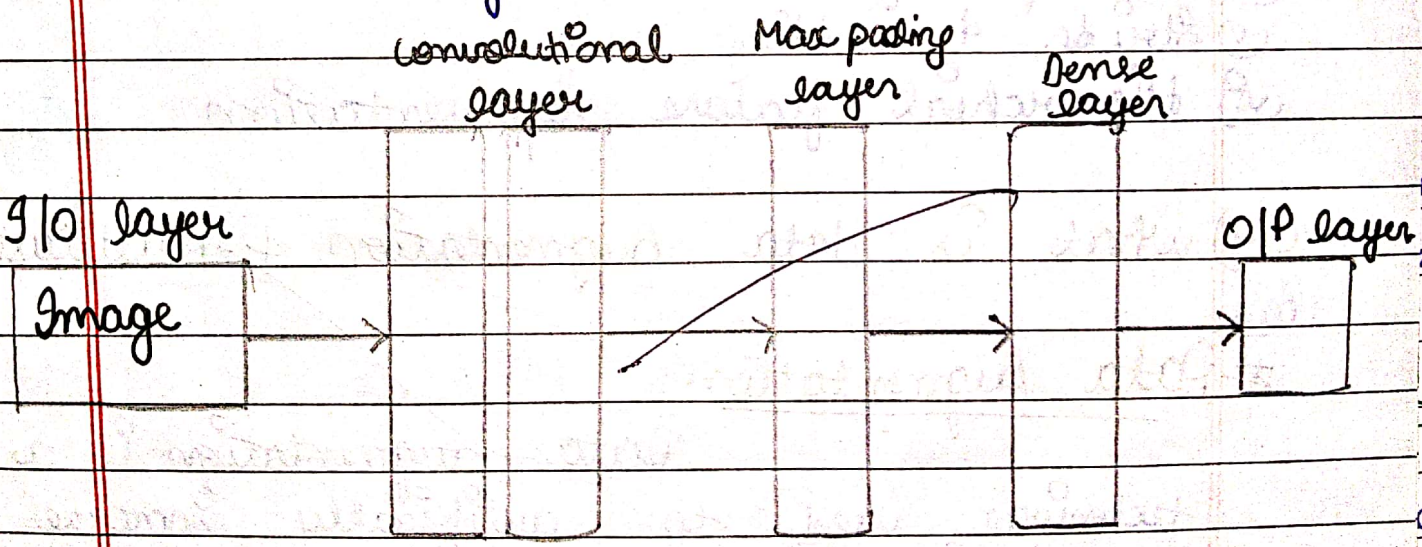
1) Dimensionality reduction
2) Translation Invariance
3) Feature selection

Disadvantages of Pooling layer

1) Info. Loss
2) Over - Smoothing
3) Hyperparameter tuning

Convolutional Algo

A convolutional neural N/W (CNN) is a type of DL algo. that is particularly well suited for image recognition & processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers & fully connected layers.



CNN ARCHITECTURE

Functions

i) Feature extraction

(ii) Parameter sharing
(iii) Dimensionality Reduction
(iv) Efficient computation
(v) Locality Preservation

Operations

(i) Filtering
(ii) Summation
(iii) Translation Invariance
(iv) Multiple filters
(v) Stride & Padding
(vi) Hierarchial feature representation

__/__/____

23

Convolutional Neural N/W (CNN)s-

It is type of deep learning architecture used in computer vision. It is a field of AI that enables a computer to understand and interprate the image & visual data.

Computer Vision

It is concerned with automatic extraction analysis & understanding of useful info. from a single image or a sequence of images. It deals with how computers can be made to gain high level understanding from digital images or videos.

→ CNNs- It is a class of DL. CNN is one of main category to do image recognition, images classification, object detection recognition faces & many more.
It is similar to basic neural N/W. CNN also have learnable parameter like neural N/W that is weights, biases & many more.
It is heavily used in CV.

Three main components of CNNs-

(i) convolutional layer
(ii) The Pooling layer
(iii) O/P layer / fully connected layer.

(i) Convolutional layer - computer reads images as pixels & it is expressed

in matrix (N×N×3) (height × weight × depth). It makes a use of set of learnable filters. A filter is used to detect the presence of specific purpose/feature or pattern present in the original image or I/O. It is usually smaller dimensions but the same depth as the I/O file. This filter is slided across the width & height of the I/O file & dot product is computed to give an activation map.

(ii) Pooling layer :- It can be seen in b/w the convolution layer & O/P or fully connected layer in CNN architecture. This layer basically reduces the amount of parameters & computation in the N/W. Pooling is done for the sole purpose of using the spatial size of images. Pooling is done independently on each depth dimension This depth of image remain unchanged.

(iii) O/P layer :- After multiple layers of convolution & Padding we would need the O/P in the form of the class.

- The convolution & pooling layer would only be able to extract features & reduces the no. of parameters from the original images. However, to generate the final O/P we need to apply a fully connected layer to generate our O/P equal to the no. of classes we need convolution layer generates 3-D activation maps.

- while we just need the O/P as whether or not an image belong to particular class.
- The O/P layer will have the loss fn. like cross entrophy to compute the error in prediction once the forward Pass is complete the back propagation begins to update the weights & biases for error & loss reduction.

Unsupervised features

In this, features are learned with Unlabeled I/P data by analyzing the relationship b/w points in dataset.

Eg

Include dictionary learning, Independent component analysis, matrix factorization etc.

Several approaches Including :-

(i) K-means clustering
(ii) Principal component analysis (is used for dimension reduction).
(iii) Local linear embedding
(iv) Unsupervised dictionary learning
(v) Independent component analysis.

Neuroscientific for convolution N/W

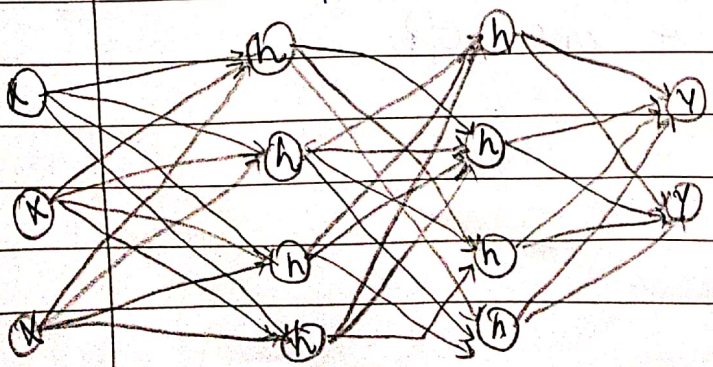
CNN are inspired by the visual processing system in the brain, particularly the primary visual cortex (V1). V1 consists of neurons with receptive fields that respond to specific visual features like edges & orientations. This biological inspiration

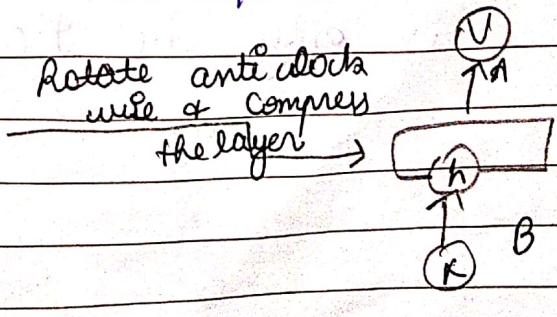
translates into CNN features :-

- Local connectivity
- Shared weights
- Hierarchial processing.

12/10/23 # RNN

- RNN stands for Recurrent Neural N/w is a type of neural N/w where the O/P from the previous step is fed as I/O to the current step. In traditional neural N/w, all the I/O & O/P are independent of each other, but in cases when it is required the predict the next word of a sentence, the previous words are required & hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of hidden layer.
- The main & most imp feature of RNN is hidden state, which remembers some info. about a sequence. The state is also referred as memory state since it remembers the previous I/O to network.
- It uses the same parameters for each I/O as it performs the same tasks on all the I/O or hidden layers to produce the O/P. This reduces the complexity of parameters, unlike other neural N/w.





__/__/____

Adv :- (i) Possibility of processing I/P of any length.
(ii) Model size not increasing with the size of I/O.
(iii) computation takes into account historical info.
(iv) weights are shared across time.

Disadv :- (i) computation being slow
(ii) Difficulty of processing info from a long time ago.
(iii) Can't consider any future I/P for the current state.

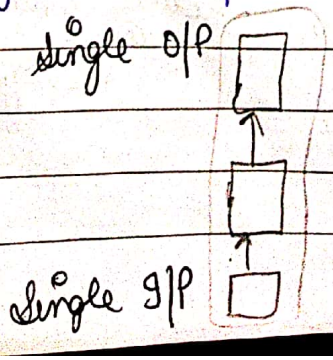
Applications :-

(i) Speech Recognition
(ii) Image "
(iii) Face detection
(iv) Time series forecasting
(v) Natural Language Processing

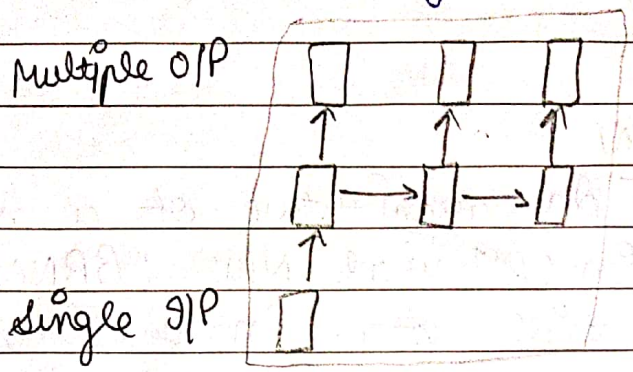
→ Types of RNN

There are four types of RNNs based on the no. of inputs & O/P in the N/w.

(1) One to one :- It is also known as Vanilla Neural N/w. In this Neural N/w, there is only one I/O & one O/P.



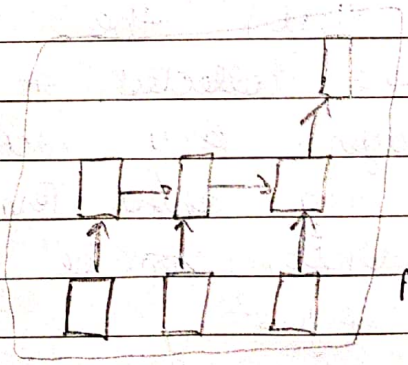
(2) One-to Many :- There is one I/O & many O/P associated with it. one of most used eg Image captioning ; where given an Image we predict a sentence having multiple words ;.



(3) Many to one :- Many I/P are fed to the N/w at several states of N/w generating only O/P.

eg

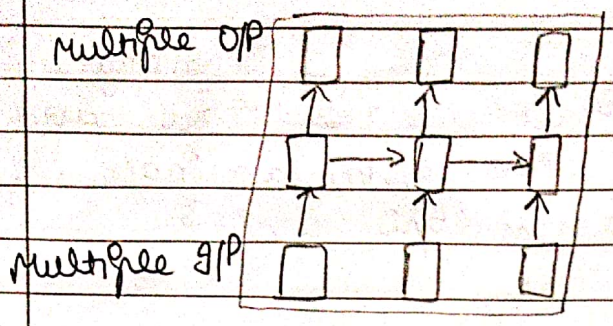
Sentimental Analysis ; where we give multiple words as I/O & predict only the sentiment of sentence as O/P.



(4) Many to many :- There are multiple I/Ps & multiple O/Ps corresponding to a problem.

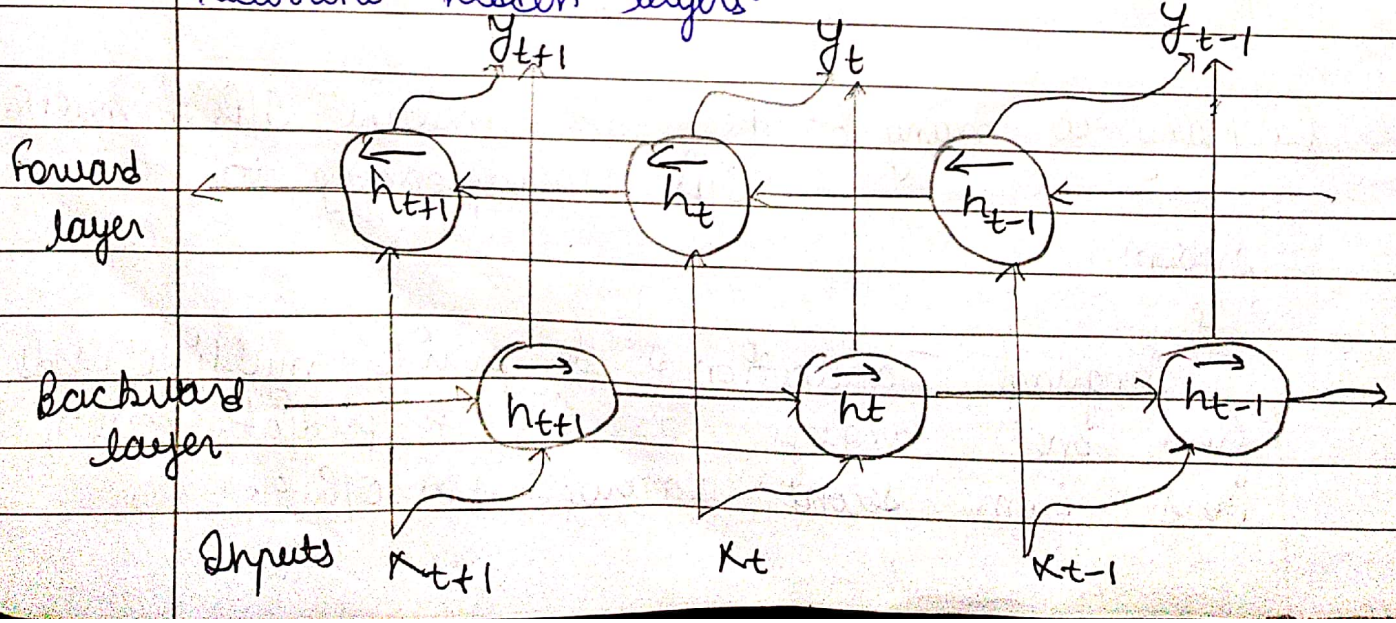
eg

Language Translation ; we provide multiple words from one language as I/O & predict multiple words from second language as O/P ;.



13/10/23 # Bi-directional RNN

- An architecture of a neural N/w called a bidirectional recurrent neural N/w (BRNN) is made to process sequential data. In order for the N/w to use info. from both the Past & future context in its predictions, BRNNs process I/O sequences in both the forward & backward directions. This is the main distinction b/w BRNNs & conventional recurrent neural N/w.
- A BRNN has two distinct recurrent hidden layers, one of which processes the I/O sequence forward & the other of which processes it backward. After that, the results from these hidden layers are collected & I/O into a prediction making final layer. Any recurrent neural N/w cell, such as Long short-Term Memory (LSTM) or Gated Recurrent Unit, can be used to create the recurrent hidden layers.



__/__/__

→ Working of BRNN

1) Inputting a sequence
2) Dual processing
3) Computing the hidden state
4) Determining the O/P
5) Training

To calculate the O/P from an RNN unit, we use the following formulas

- H_t (Forward) = A (X_t * W_{XH} (forward) + H_{t-1} (forward) * W_{HH} forward + b_H (forward)

- H_t (Backward) = A (X_t * W_{XH} (Backward) + H_{t+1} (Backward) * W_{HH} (Backward) + b_H (Backward)

where
A = Activation function
W = Weight matrix
b = bias

The hidden state at time t is given by a combination of H_t (forward) + H_t (Backward). The O/P at any given hidden state is :-

$$Y_t = H_t * W_{AY} + b_Y$$

→ Applications of BRNN

1) Sentiment Analysis
2) Named entity Recognition

__/__/____

(iii) Part of speech Tagging
(iv) Machine Translation
(v) Speech Recognition

Adv:- (i) Context from both Past & future
(ii) Enhanced accuracy
(iii) Efficient handling of Variable length sequences.
(iv) Resilience of noise & irrelevant info.
(v) Ability to handle Sequential dependencies.

Disadv:- (i) Computational Complexity
(ii) Long Training Time
(iii) Difficulty in Parallelization
(iv) Overfitting
(v) Interpretability.

Parameters	RNN	BRNN
(i) Full form	RNN stands for Recurrent neural N/W.	BRNN stands for Bidirectional RNN.
(ii) Data type	Sequential Data.	Sequential Data.
(iii) Processing mechanism	Processes data in a sequential manner, maintaining info. from previous I/O.	Processes a data in sequential manner, maintaining info. from both previous & future I/P.
(iv) Architecture	Single directional recurrent connections.	Forward & backward recurrent connections.
(v) Applications	1) Image Recognition 2) Speech " 3) Face detection 4) Time series forecasting 5) NLP (Natural language processing).	1) Sentiment Analysis 2) Named Entity Recognition 3) Part of Speech Tagging 4) Machine Translation 5) Speech Recognition.
(vi) Computational cost	Low	High.
(vii) complex Architecture	NO	Yes

Ans

Seq 2 Seq Model

The encoder decoder architecture also known as the Sequence-to-Sequence (Seq2 Seq) Model, is a widely used deep learning architecture for tasks that involve mapping one sequence to another.

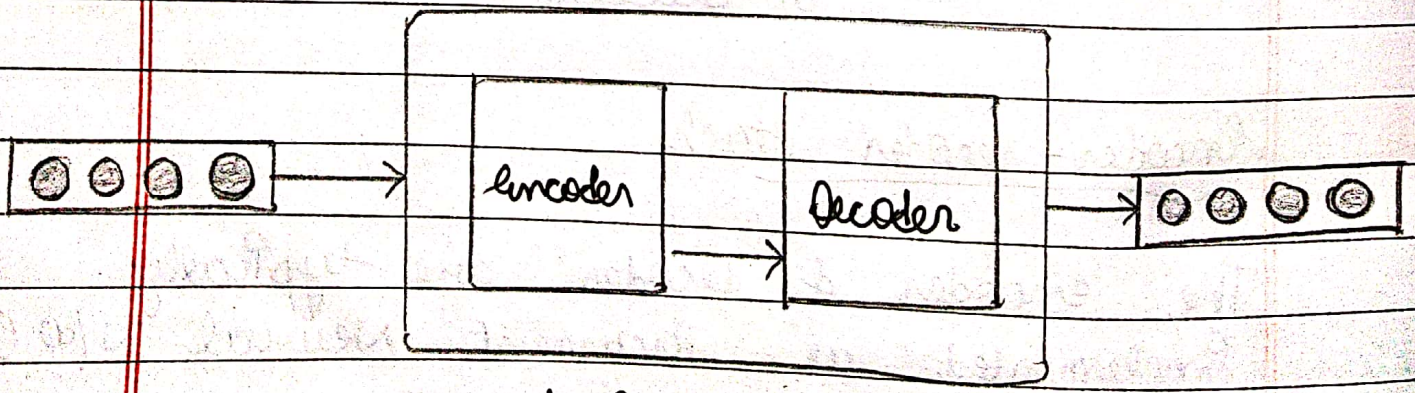
- It is a type of model in ML that is used for tasks such as machine translation, text summarization & image captioning.
- The model consists of two main components :- a) Encoder
  b) Decoder

Encoder-decoder stack

The encoder & decoder are typically implemented as Recurrent Neural N/w (RNN) or Transformers.

a) Encoder Stack :- It uses deep neural N/W layers & converts the input words to corresponding hidden vectors. Each vector represents the current word & the context of word. The encoder takes the I/O sequence, one token at a time & uses RNN.

b) Decoder Stack :- It is similar to the encoder. It takes the I/O the hidden vector generated by the encoder, its own hidden states, & the current word to produce the next hidden vector & finally predict the next word. The decoder uses the context vector & an initial hidden state to generate the O/P sequence, one token at a time.



Seq2Seq Model

Advantages of Seq2Seq Models

(i) Flexibility
(ii) Handling sequential data
(iii) Handling context
(iv) Attention Mechanism

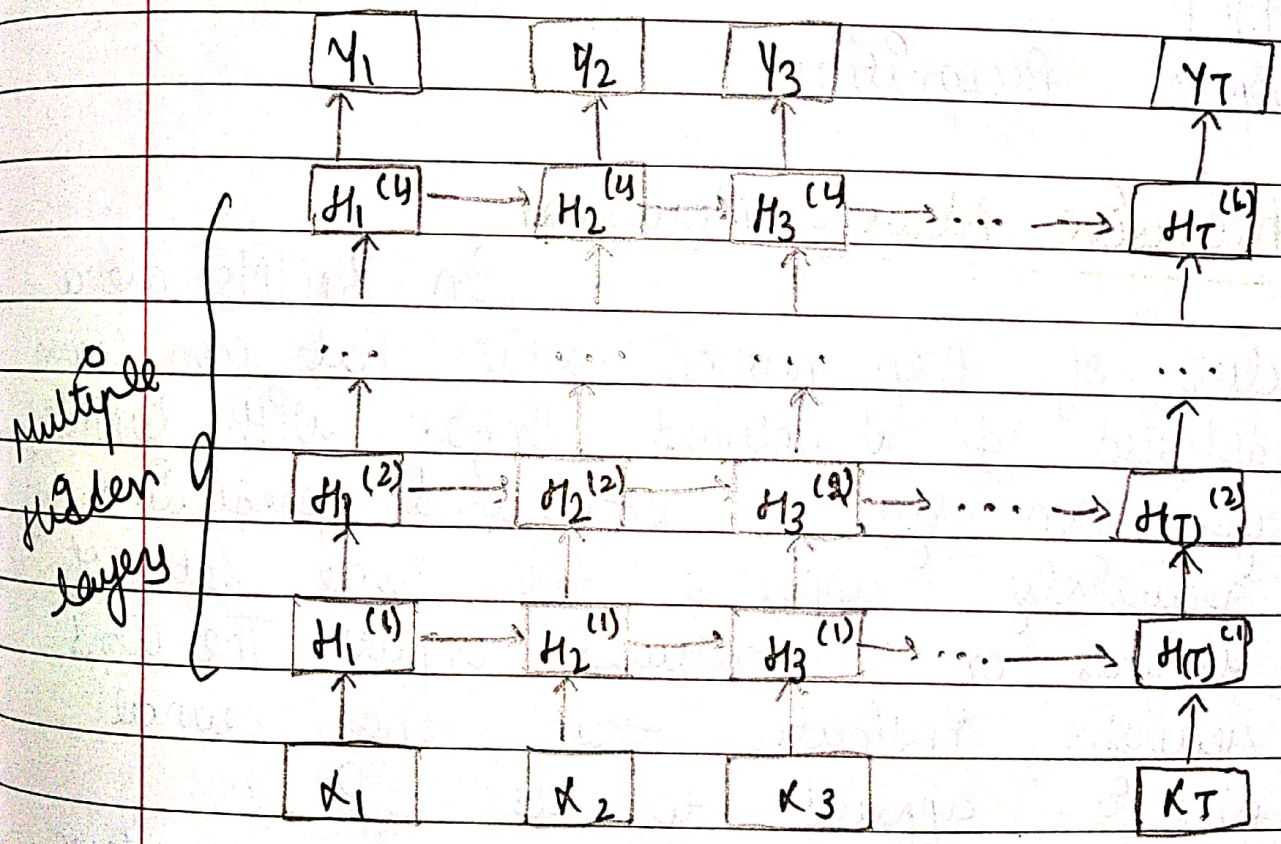
Disadvantages of Seq2Seq Models

(i) computationally expensive
(ii) Limited Interpretability
(iii) Overfitting
(iv) Handling Rare words
(v) Handling Long I/O sequences

Unit 4th

DRNN

DRNN stands for "Deep Recurrent Neural N/w". It refers to a neural N/w architecture that has multiple layers of recurrent units. A Deep RNN takes the O/P from one layer of recurrent units & feeds it into the next layer, allowing the N/w to capture more complex relationships b/w the I/P & O/P sequences.



Steps to develop a deep RNN application

1. Data preparation
2. Model architecture design

(iii) Training the model
(iv) Evaluating the model
(v) Deploying the model

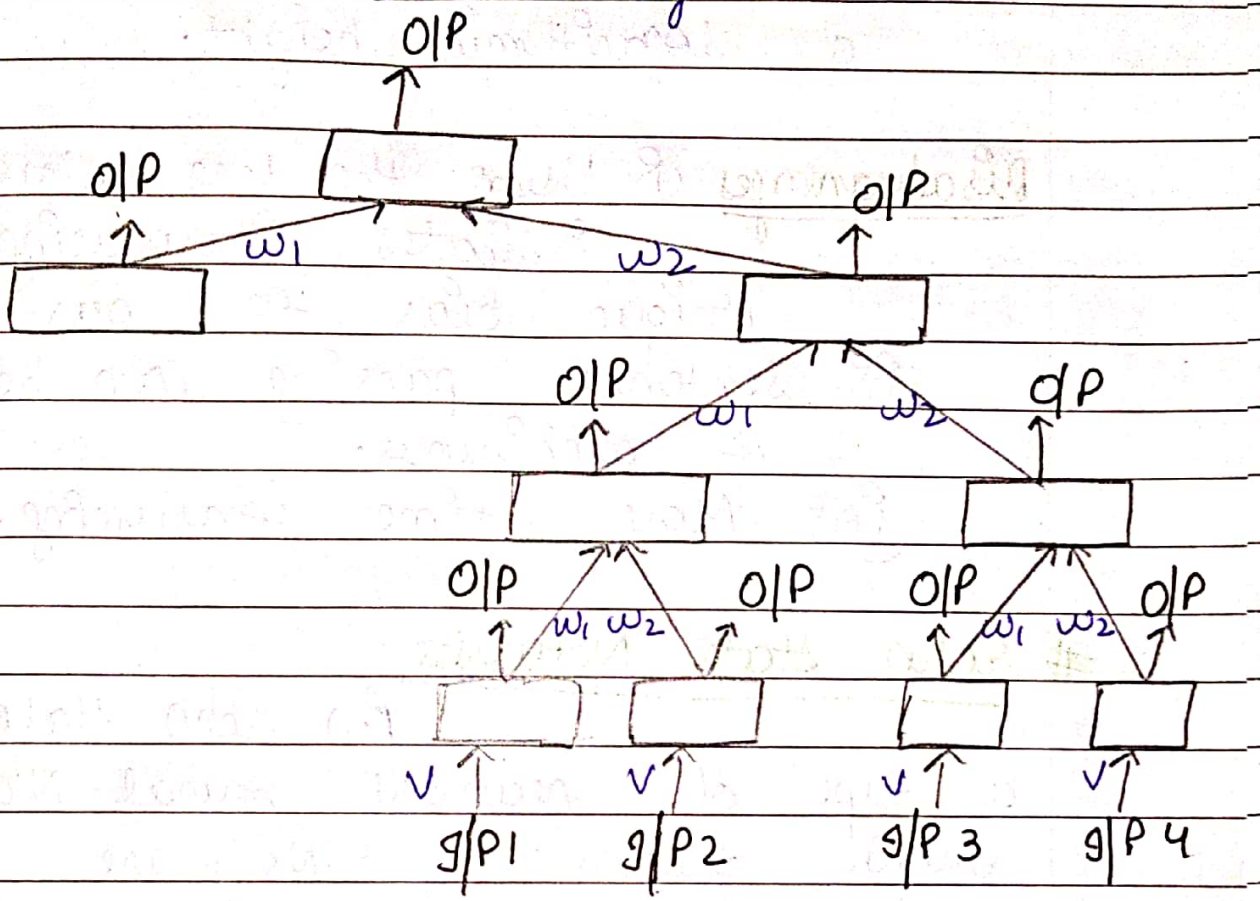
Applications of DRNN

(i) Music Generation
(ii) Autonomous Driving
(iii) Image Captioning
(iv) NLP
(v) Speech Recognition

Recursive Neural Networks

In (RvNNs) are a class of deep neural N/w that can learn detailed & structured Info. With RvNNs, you can get a structured prediction by recursively applying the same set of weights on structured Inputs. The word recursive indicates that the neural N/w is applied to its ofp.
Due to their deep tree like structure, Recursive Neural N/w can handle hierarchical data. The tree structure means combining child nodes & producing parent nodes. Each child-

Parent bond has a weight matrix, & similar children have same weights.



Eg

NLP

↓

it identifies whether the sentence showcase a constructive form of writing or negative word choices.

Benefits ⓘ NLP are their structure & reduction in N/W depth.

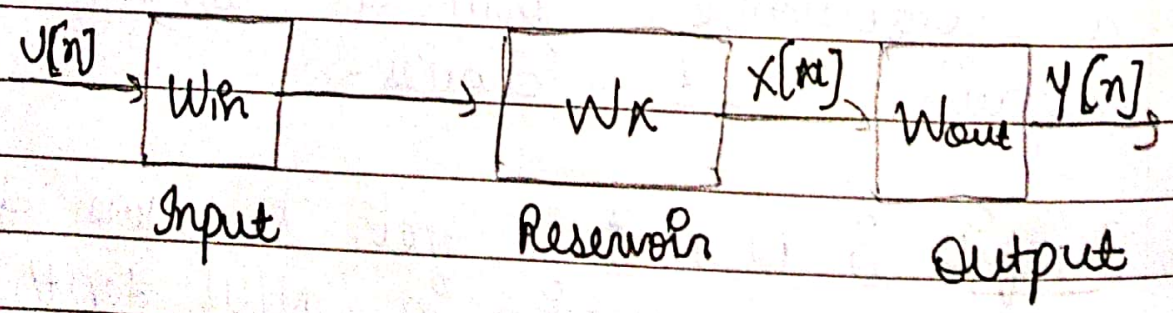
ⓘⓘ It can manage hierarchical

data like in parsing problems.
iii) RNN is that the trees can have a logarithmic height.

Disadvantages i) Using the tree structure indicates introducing a unique bias to our model.
ii) Sentence parsing can be slow & ambiguous.
iii) More time consuming.

Echo State Networks

An Echo state N/w is a type of recurrent neural N/w. The unique feature of ESNs, are opposed to traditional RNNs, & the way they're trained. The connectivity & weights of hidden neurons are fixed & randomly assigned.



How does an ESN work?

Here, is a breakdown of the working of ESN:-

(i) Reservoir :- This is the heart of the ESN. It is a collection of interconnected neurons, where the connections & their weights are randomly initialized & fixed. The reservoir acts as dynamic memory & its purpose is to transform the I/P data into high dimensional space.

(ii) Input layer :- The input signal is fed into the reservoir. This can be single value, a vector, or even a sequence of data points.

(iii) Output layer :- After passing through the reservoir, the transformed data is then used to train only the O/P layer. This is typically done using a linear regression method, making the training process relatively fast & efficient.

Adv. of ESN

(i) Easy to train
(ii) Fast

(iii) Robust

Disadvantages of ESN

(i) Black box (can be difficult to understand & works)

(ii) Limited capacity.

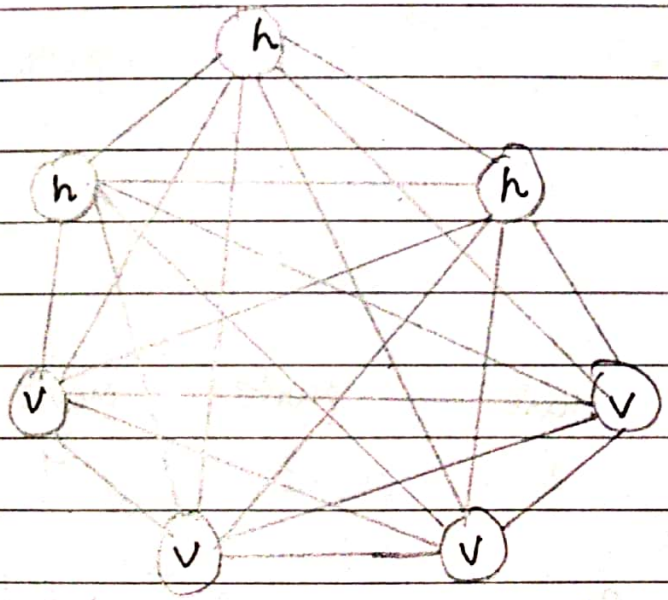
→ Applications of ESN

1) Time series forecasting
2) System Identification
3) Speech recognition
4) NLP
5) Robotics

Unit 5th

Boltzmann Machines

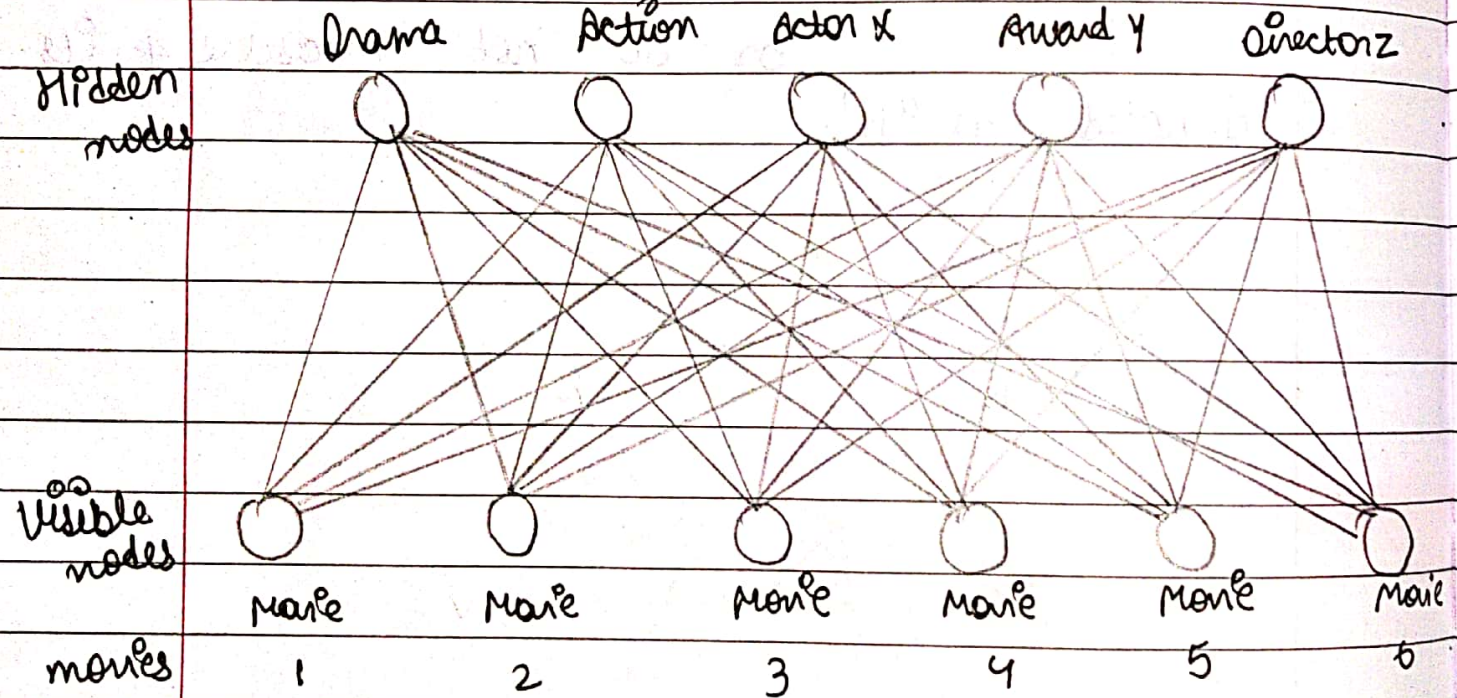
- It is an unsupervised DL model in which every node is connected to every other node. Boltzmann Machines are not a deterministic DL model but a stochastic or generative DL model.
- There are two types of nodes in Boltzmann Machines –
- Visible nodes = those nodes which we can & do measure & the visible layer is denoted as "v".
- Hidden nodes = those nodes which we cannot or do not measure & its denoted as "h".



Types of Boltzmann Machines

1) Restricted Boltzmann Machines (RBMs) :- In a fully Boltzmann machine, every node is connected to each other & hence the connections grow exponentially. This is the reason we use RBMs. The restrictions in the node connections in RBMs are as follows -

- Hidden nodes cannot be connected to one another.
- Visible nodes connected to each other.



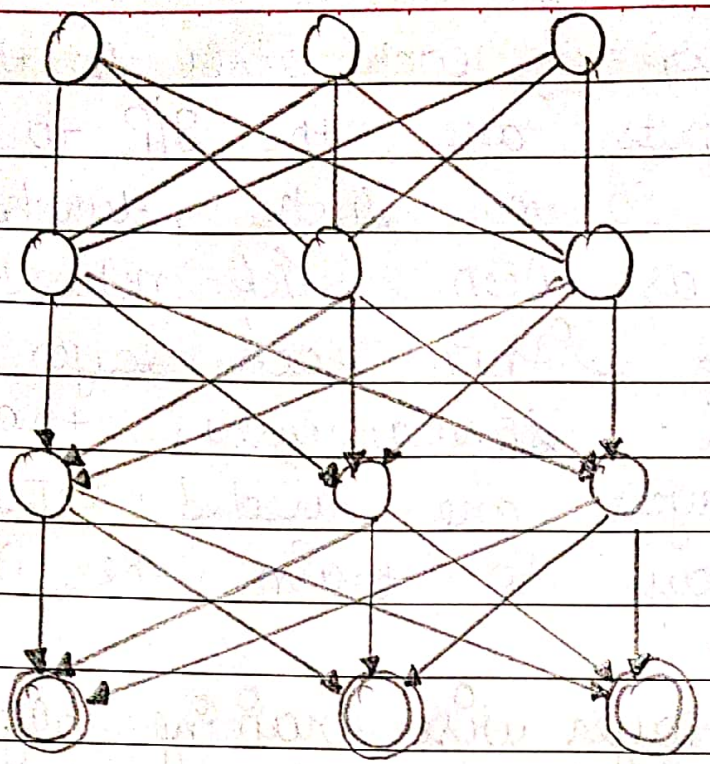
2) Deep Belief Networks (DBNs) :- Suppose we stack several

RBMs on top of each other so that the first RBM outputs are the I/P to the second RBM & so on. Such networks are known as Deep Belief networks. The connections within each layer are undirected. Simultaneously, those in b/w the layers are directed. There are two ways to train the DBNs :-

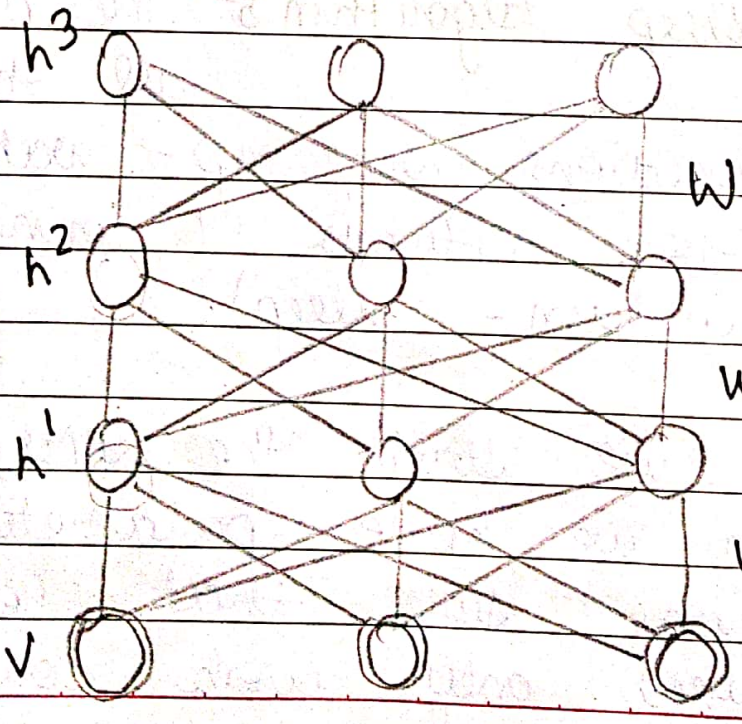
1) Greedy layer wise training algorithms :- The RBMs are trained layer by layer. Once the individual RBMs are trained, the direction is setup b/w the DBN layers.

2) Wake - sleep Algorithm :- The DBN is trained all the way up (connections going up - wake) & then down the network (connections going down - sleep).

Therefore, we stacks the RBMs, train them & once we have parameters trained, we make sure that connections b/w layers only work downwards.



3) Deep Boltzmann Machines :- DBMs are similar to DBNs except that apart from the connections within layers, the connections b/w the layers are also undirected.



- DBMs can extract more complex or sophisticated features & hence can be used for more complex tasks.

Generative Adversarial Network (GAN)

- GAN are a powerful class of neural N/w that are used for unsupervised learning. GANs are made up of two neural N/w, a discriminator & a generator. They use adversarial training to produce artificial data that is identical to actual data. The generator attempts to fool the discriminator, which is tasked with accurately distinguishing b/w produced & genuine data, by producing random noise samples.

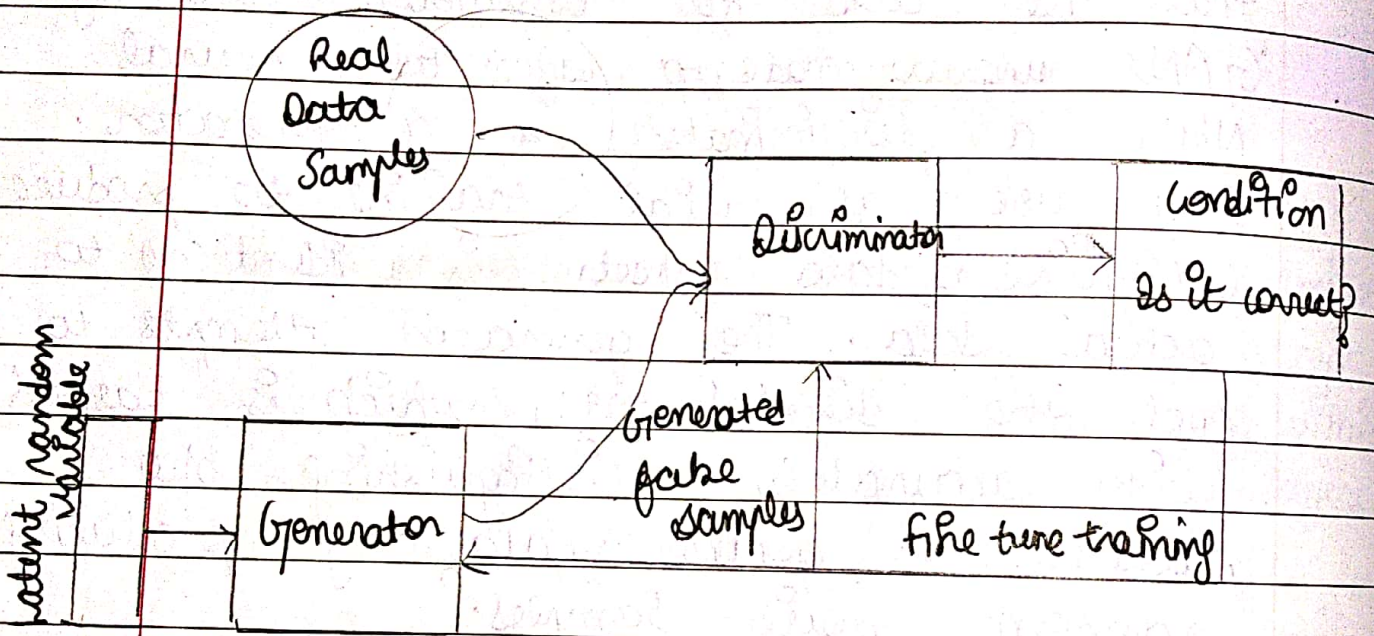
- It can be broken down into three parts :-

(i) Generative :- To learn a generative model, which describes how data is generated in terms of probabilistic model.

(ii) Adversarial :- A mechanism known as a

Discriminator is used to apply a model that attempts to distinguish b/w real & fake images.

(iii) Networks :- Use deep neural N/w as AI algorithms for training purposes.



GAN Architecture

→ Different types of GAN Models

(i) Vanilla GAN :- Here, the generator & discriminator are simple multi° layer perceptrons. The algorithm is really simple, it tries to

optimize the mathematical eqn using stochastic GD.

(ii) Conditional GAN (CGAN):- It can be describe as DL method in which some conditional parameters are put into place.

(iii) Deep convolutional GAN (DCGAN):- The ConvNets are implemented without max pooling, which is in fact replaced by convolutional stride.

(iv) Laplacian Pyramid GAN (LPGAN):- It is linear invertible image representation consisting of a set of and pass images, spaced an octave apart, plus a low frequency residual.

(v) Super resolution GAN (SRGAN):- It is used along with an adversarial N/w in order to produce higher resolution images.

→ Applications of GAN

1) Image synthesis & generation
2) Image to Image translation
3) Text to Image Synthesis
4) Data Augmentation
5) Data generation for Training.

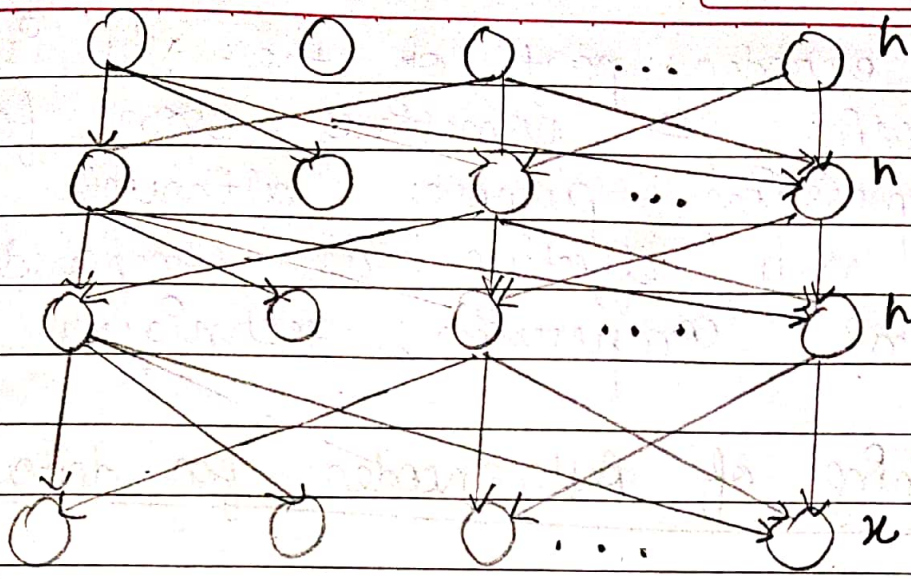
Advs- 1) Synthetic data generation
2) High quality results
3) Unsupervised learning
4) Versatility.

Disadv s- 1) Training Instability
2) Computational cost
3) Overfitting
4) Bias & fairness
5) Interpretability & Accountability

Sigmoid Belief Networks

SBNs are a type of probabilistic generative model that uses sigmoid activation functions in its hidden units. They are a type of DL model, meaning that they have multiple layers of interconnected units

Diagram



→ Applications of SBNs

1) Image recognition
2) NLP
3) Recommendation systems
4) Bioinformatics

Adv :- 1) easy to train
2) generative models
3) Scalable to large datasets.

Disadv :- 1) slow to train
2) sensitive to initialization
3) They may not always converge to global optimum.

dng

Directed Generative nets

Directed generative nets are a class of deep generative models that use directed graphs to represent the relationships b/w variables. These models are typically trained using maximum likelihood estimation or variational inference.

Types of DGN

There are several diff. types of directed generative nets, including :-

(i) Sigmoid belief nets :- These are the simplest type of directed generative nets, & they use sigmoid activation functions to model the relationships b/w variables.

(ii) Differentiable generation nets :- These nets use differentiable functions to model the relationships b/w variables, which makes them more

flexible.

(iii) Variational Autoencoders (VAEs) :- These nets use a combination of an encoder & a decoder to generate New data. The encoder learns a latent representation of data & decoder generates new data from this latent representation.

(iv) Generative Adversarial Networks (GANs) :- These net consists of two networks :- a generator
- a discriminator

The generator learns to generate new data, & the discriminator learns to distinguish b/w real data & generated data.

Adv :- (i) More expressive
(ii) More efficient
(iii) More interpretable

Disadv :- (i) They can be more difficult to train.

(ii) They can be more prone to overfitting.

Drawing samples for Auto encoders

The encoder part of the N/W is used for encoding & sometimes even for data compression purposes although it is not very effective as compared to other general compression techniques like JPEG.

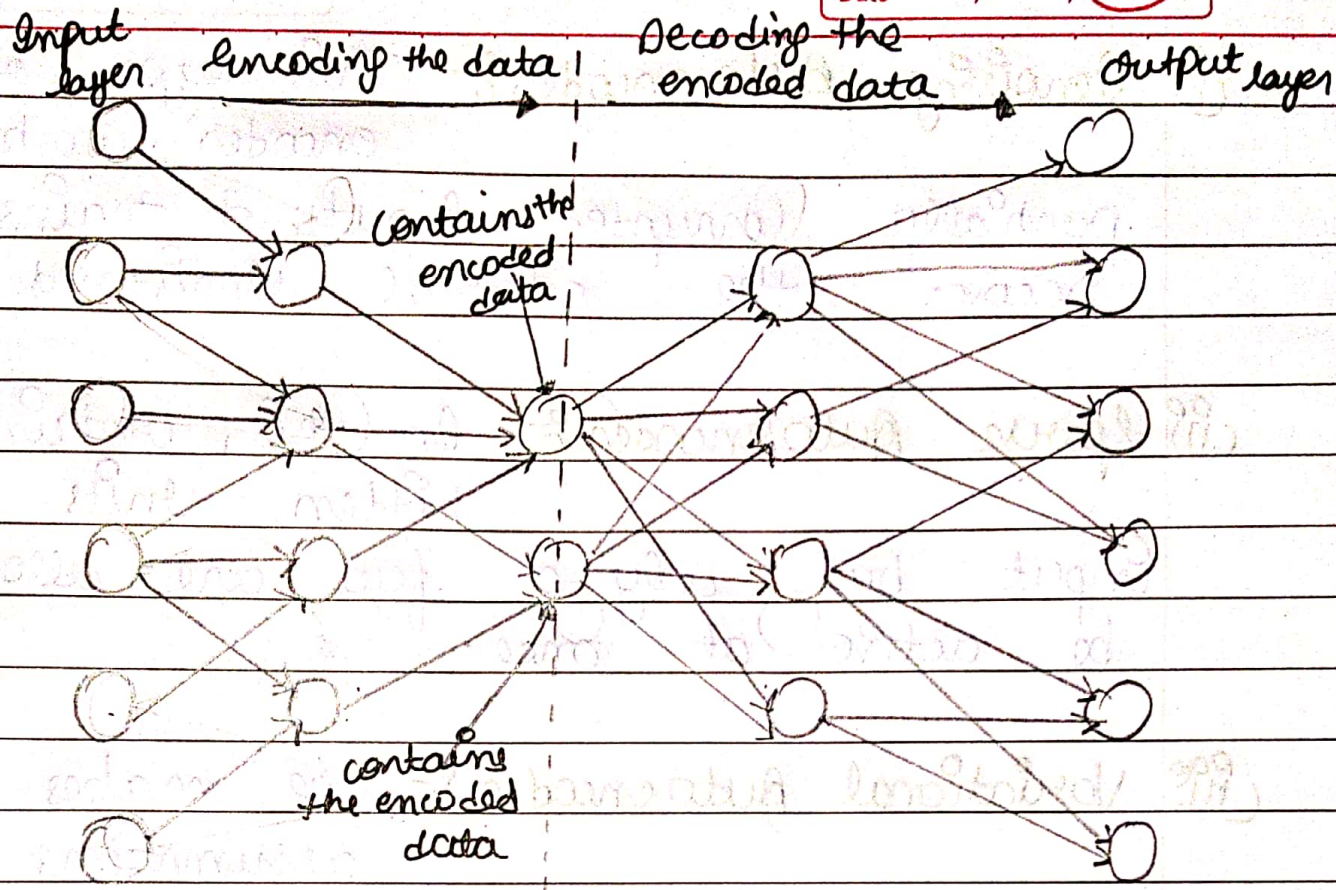
Training of Auto encoder for data compression

Step1 Encoding the input data.
The auto encoder first tries to encode the data using the initialized weights & biases.

Step2 Decoding the input data.
The auto encoder tries to reconstruct the original I/P from the encoded data to test the reliability of encoding.

Step3 Backpropagating the error.
After the reconstruction, the loss function is computed to determine the reliability of the encoding. The error generated is backpropagated.

Complete steps diagram



Backpropagating the error

The different ways to constrain the N/W are :-

- Keep small hidden layers
- Regularization
- Denoising
- Tuning the activation functions

The different Variations of Auto-encoders are :-

(i) Denoising Auto encoders:- This type of auto-encoder works on a partially corrupted input & trains to recover the original undistorted image.

(ii) Sparse Auto encoders:- In this, contains more hidden units than the input but only a few are allowed to be active at once.

(iii) Variational Auto encoders:- we makes strong assumptions about the distribution of latent variables & uses the stochastic gradient variational Bayes estimator in the training process.

(iv) convolutional Auto encoders:- These type of auto-encoder that use CNN as their building blocks. The encoder consists of multiple layers that take a image or a grid as input & pass it through different convolution layers thus forming a compressed representation of input.

Similarities Between Bagging and Boosting

Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

Differences Between Bagging and Boosting

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Difference between L1 & L2 regularization

L1 Regularization	L2 Regularization
The penalty term is based on the absolute values of the model's parameters.	The penalty term is based on the squares of the model's parameters.
Produces sparse solutions (some parameters are shrunk towards zero).	Produces non-sparse solutions (all parameters are used by the model).
Sensitive to outliers.	Robust to outliers.
Selects a subset of the most important features.	All features are used by the model.
Optimization is non-convex.	Optimization is convex.
The penalty term is less sensitive to correlated features.	The penalty term is more sensitive to correlated features.
Useful when dealing with high-dimensional data with many correlated features.	Useful when dealing with high-dimensional data with many correlated features and when the goal is to have a less complex model.
Also known as Lasso regularization.	Also known as Ridge regularization.

Comparison of DFNNs and DRNNs

Feature	Deep Feedforward Neural Networks (DFNNs)	Deep Recurrent Neural Networks (DRNNs)
Structure	Sequential layers of interconnected neurons	Layers with feedback loops and hidden memory
Strengths	Classification, regression, feature extraction	Capturing temporal dependencies, sequential data processing
Applications	Image recognition, natural language processing, forecasting	Machine translation, speech recognition, time series forecasting
Data Type	Independent data points	Sequential data, time series data

Export to Sheets

Feature	Forward propagation	Backward propagation
Direction	Input to output	Output to input
Purpose	Calculate the output of the neural network given an input	Calculate the gradients of the loss function with respect to the weights and biases of the neural network
Steps	1. Compute the weighted sum of the inputs to each neuron. 2. Apply an activation function to the weighted sum. 3. Repeat steps 1 and 2 for each layer of the neural network.	1. Calculate the error between the predicted output and the actual output. 2. Propagate the error backwards through the network, layer by layer, calculating the gradients of the loss function with respect to the weights and biases of each layer.
Use	Used to make predictions on new data	Used to train the neural network

Summation of all three networks in single table:

	ANN	*CNN*	*RNN*
Type of Data	Tabular Data, Text Data	Image Data	Sequence data
Parameter Sharing	No	Yes	Yes
Fixed Length input	Yes	Yes	No
Recurrent Connections	No	No	Yes
Vanishing and Exploding Gradient	Yes	Yes	Yes
Spatial Relationship	No	Yes	No
Performance	ANN is considered to be less powerful than CNN, RNN.	CNN is considered to be more powerful than ANN, RNN.	RNN includes less feature compatibility when compared to CNN.
Application	Facial recognition and Computer vision.	Facial recognition, text digitization and Natural language processing.	Text-to-speech conversions.
Main advantages	Having fault tolerance, Ability to work with incomplete knowledge.	High accuracy in image recognition problems, Weight sharing.	Remembers each and every information, Time series prediction.
Disadvantages	Hardware dependence, Unexplained behavior of the network.	Large training data needed, don't encode the position and orientation of object.	Gradient vanishing, exploding gradient.