Practical Machine Learning

Theory Notes

- Ajay Singh
PRN- 2403501250006

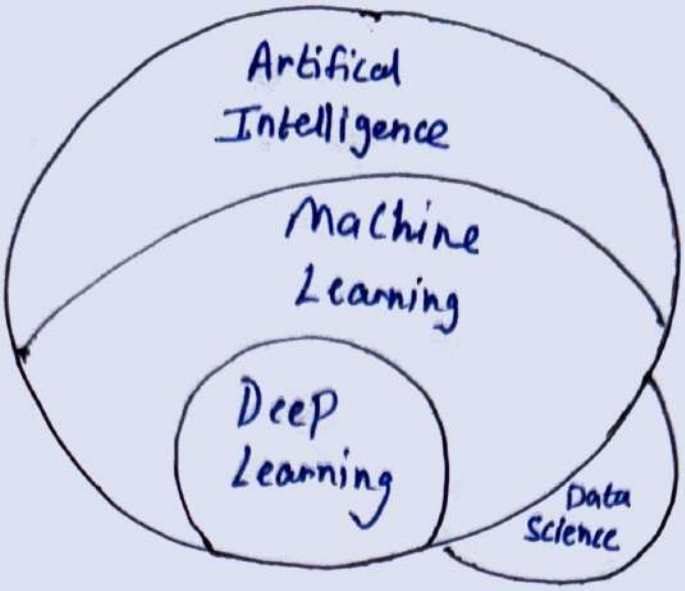
Introduction

* What is Machine learning?

→ Machine learning is a branch of artifical intelligence that devolops algorithms by learning the hidden patterns of the datasets used it to make predictions on new Similar type data, without being explicity programmed for each task.

A Computer Program which Learns from experience is called a machine learning program or simply a learning program.

Machine learning is used in many different applications, from image and speech recognition to natural language processing recommendation systems, fraud detection, portfolio optimization, automated risk and so on. Machine learning models are also used to power autonomous vechicles, drones and robots, making them more Intelligent and adatable to changing environments.



Classification of Machine Learning:

Machine Learning implementations are classification into four major categories, depending on the nature of the learning "signal" or "response" available to a learning system which are as follows:

Supervised learning:

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. The given data is labeled. Both classification and regression problems are Supervised learning problems.

Unsupervised learning :

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, Classification or categorization is not included in the observations.

Reinforcement learning :

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize rewards.

A learner is not told what actions to take as in most forms of machine learning but instead must discover which actions yield the most reward by trying them.

Semi-supervised learning :

where an incomplete training signal is given : a training set with some of the target outputs missing. There is a special case of this principle known as transduction where the entire set of problem instance is known at

learning time, except that part of the target are missing.

Machine Learning Lifecycle

1) Study the Problems:

The first step is to study the Problem. This step involves understanding the business problem and defining the objectives of the model.

2) Data Collection:

When the problem is well-defined, we can collect the relevant data required for the model. The data could come from various sources such as databases, APIs, or, web scraping.

3) Data Preparation:

When our problem-related data is collected then it is a good idea to check the data properly and make it in

the defined formats so that it can be used by the model to find the hidden Patterns.

4) Model Selection :

The next step is to select the appropriate machine learning algorithm that is suitable for our problem. This step requires knowledge of the strengths and weakness of different algorithms.

5) Model Evaluation :

Once the model is trained, it can be evaluated on the test dataset to determine its accuracy and performance.

6) Deployment :

Once the model is trained and tuned, it can be deployed in a Production environment to make Prediction on new data.

* Importance & Applications of machine learning

Machine learning is important because it allows computers to learn from data and improve their performance on specific tasks without being explicitly programmed. This ability to learn from data & adapt to new situations makes machine learning particularly useful for tasks that involves large amounts of data, complex decision-making and dynamic enviroments.

* Predictive Modelling:

→ Machine learning can be used to build predictive models that can be help businesses make better decisions

* Natural language Processing :-

→ In Computer, Machine learning is used to build systems that can understand and interpret human language.

* Computer vision:

→ Machine learning can be used to build system that can understand and interpret human language.

* Fraud Detection:

→ Machine learning can be used to detect fraundulent behaviour in fiancial transactions, online advertising and other areas.

* Recommendation system:

Machine learning can be used to build recommendation systems that suggest products; services, or content to users based on their past behaviour and prefences.

Various applications of machine learning

* Automation: Machine learning, which works entirely autonomously in any field without the need for any human.

Finance Industry: Machine learning is growing in popularity in the finance Industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraund.

Government Organization: The government makes use of ML to manage public Safety and utilities. Take the example of China with its massive face recognition. The government uses Artifical intelligence to prevent jaywalking.

Healthcare Industry: Healthcare was one of the first industries to use machine learning with image detection.

Transportation :- Machine learning is used in the transportation industry to optimize routes, reduce fuel Consumption, and improve the overall efficient of transportation systems.

* Machine Learning v/s Traditional Programming

Machine Learning

1) Machine learning is a subset of artificial intelligence (AI) that focus on learning from data to develop an algorithm that can be used to make a prediction.

2) Machine learning uses a data-driven approach it is typically trained on historical data and then used to make predictions on new data.

3) ML can find patterns and insights in large datasets that might be difficult for humans to discover.

Traditional Programming

1) In traditional, Programming rule based code is written by the developers depending on the problem statements.

2) Traditional programming is typically rule-based and deterministic. It hasn't self-learning features like machine learning and AI.

3) Traditional programming is totally dependent on the intelligence of developers. So, it has very limited capability.

Fundamentals of python

* Python Basics

→ Python is a versatile and widely-used programming language known for its simplicity & readability.

→ Hello, world!

let's start with a simple Python program that prints "Hello, world!" to the console. This is often the first program people write in any language to get familiar with its syntax.

Python

```
Print ("Hello, world!")
```

2) Indentation -

→ Python uses indentation to define blocks of code. It's crucial to maintain consistent intdentation for readability and to avoid synatx errors.

```
if True:
    print("This is indented correctly")
```

3) Comments :-

→ Comments are used to add explainations within your code. They start with a '#' symbol.

4) Variables & Data Types :-

→ Python supports various data types, including integers, floats, strings, lists, tuples, dictionaries & more variables are used to store data.

```
# variables
X = 5
name = "Alice"

# Data types
age = 25
height = 5.9
fruits = ["apple", "banana", "cherry"]
```

5) Operators :

→ Python provides operators for performing operations on variables & values. Common operator include +, -, *, /, ==, !=, <, >, <=, >= and more.

6) Input & Output :

→ you can take user input using the 'input()' & display output using 'print()'.

```
user_input = input ("Enter your name :")
Print ("Hello," + user_input)
```

7) Conditional Statements :

Python supports 'if', 'elif' and 'else' statements for conditional execution of code.

```
if x>0:
    print("x is positive")
elif x<0:
    print("x is negative")
else: print("x is zero")
```

8) Loops :

→ You can use 'For' & 'while' loops for iteration.

```
for i in range (5):
        print(i)
while x>0
    Print (x)
    x-=1
```

9) Functions :

→ Functions allow you to define reusable blocks of code. you can define functions using the 'def' keyword.

```
def greet (name):
        return "Hello," + name

message = greet ("Alice")
print (message)
```

10) Modules & libraries :

→ Python has a vast standard library and third-party packages. you can import modules using 'import' to access additional functionality.

```
import math
print (math.sqrt (25))
```

11) Lists & Indexing :

Lists are a versatile data structure that can hold multiple items. you can access elements in a list using indexing, where the index starts from 0.

```
my_list = [1, 3, 3, 4, 5]
Print (my_dist [0]) # Acces the first element
```

* Slicing :-

Slicing allows you to extract a portion of a list or string using a range of indices.

```
my_list = [1, 2, 3, 4, 5]
sliced_list = my_list [1:4]
```

Dictionaries :

Dictionaries are collections of key-values pairs and they provide a way to associate data with labels.

```
Person = { "name" : "Alice", "age" : 25, "city" : "Newyork" }

    print ( person ["name"]
```

list Comprehensions :

list Comprehensions often a concise way to create lists based on existing lists or ranges.

```
even_numbers = [x for x in range(10) if x % 2 == 0]
```

* Exception Handling:

Python supports try-except blocks for handling exceptions and errors gracefully.

```
try:
    result = 10/0
except ZeroDivisionError use:
print("Error:",e)
```

* File handling:

You can read from and write to files in python using the 'open()' function

```
with open("file.txt","r") as file:
    content = file.read()
with open("output.txt","w") as output_file:
    output_file.write("Hello, world!")
```

* Object-oriented programming (OOP)

Python is an object-oriented language and you can define classes & object to create reusable code.

```
Class Person
    def __init__(self, name, age):
    self.name = name
    self.age = age
    def greet (self):
        return (f"Hello,my name is {self.name} and
                I'm {self.age} years old."
person = Person ("Alice", 25)
print ( Person.greet())
```

* Numpy & Pandas for data manipulation :

- **Numpy:**

→ Numpy (Numerical python) is a fundamental library for numerical computations in python. It provides support for working with arrays, matrics and mathematical functions, making it crucial tool for data manipulation and Scientific Computing. It revolves around the 'numpy' array which is similar to a python list but more powerful due to its homogeneity and the vast number of operations it supports.

Importing Numpy:

```
import numpy as np
```

Array Creation

Numpy arrays can be created from python lists or using built-in-functions like np.array.arrange(), 'np.zeros(), and 'np.ones()'. These arrays can have multiple dimensions.

```
# Creating arrays
arr1 = np.array([1,2,3,4,5])
arr2 = np.arrange(0,10,2)
```

Array Operations:-

Numpy allows you to perform elementwise operations, making it efficient for mathematical computations. you can add, subtract, multiply, divide and apply various functions to entire arrays without explicit loops.

Arithmetic Operations

```
result = arr1 + arr2
result = np.sqrt(arr1)
```

Indexing & Slicing:

Numpy arrays support advanced indexing & slicing. you can access individual elements or entire subsets of data quickly.

indexing & slicing

```
element = arr1[2]
Sub_array = arr1[1:4]
```

Shape Manipulation:

you can reshape arrays using 'reshape()' method or change their dimensions with functions like 'np.vstack()' and 'np.hstack()'.

Shape & reshaping

```
Shape = arr1.shape
reshaped.arr1. reshaped (3,3)
```

* Aggregation & statistics :

Numpy offers a range of functions to compute statistic like mean, median, Standard deviation & more. you can also perform aggregation operations along specified axes.

Aggregation

```
mean = np.mean (arr1)
max_value = np. max(arr1)
```

* Broad casting

Numpy enables broadcasting, which allows you to perform operations on arrays with different shapes, making code concise & efficient.

Pandas:

Pandas is designed for data manipulated and analysis especially when dealing with structured data like spread-sheets or SQL tables.

Data structure:

The primary data structure in Pandas are the series and Dataframe. A series is like a labelled one-dimensional array, while a dataframe is a two-dimensional table with the labelled columns.

Importing Pandas:

```
import Pandas as Pd.
```

Data Structures:

The primary data structures in Pandas are the series & dataframe.

Data import & Export:

Pandas Provides functions to read data from various file formats (CSV, Excel, SQL databases) and export data in these formats. This makes it easy to work with external data sources.

Data Cleaning:

Pandas allows you to handle missing data, duplicate records & outliers effectively. you can remove or fill missing values & drop duplicates.

Data Selection:

Pandas provides powerful tools for selecting indexing & filtering data based on conditions. you can slice data, select columns & apply complex filters.

Data Aggregation & Grouping:

you can group data based on one or more columns & apply aggregation functions.

Creating a dataframe :

```
# creating a dataframe
data = {'Name': ['Alice', 'Bob', 'charlie'],
        'Age': [25, 30, 35]}
df = pd.Dataframe (data)
```

Basic Dataframe operations :

```
# Basic operations
head = df.head()
Shape = df.shape()
```

Index & selection

```
# indexing & selection
age_column = df['Age']
subset = df[df['Age']>30]
```

Data cleaning :-

```
#data cleaning
df.dropna()
df.fillna(0)
```

Data Aggregation & Grouping

```
group_by_age = df.groupby('Age')
mean_age = group_by_age.Mean()
```

Matplotlib & Seaborn for Data Visualization:

Matplotlib :-

Matplotlib is a widely-used library for creating static, animated or interactive plots in python. Seaborn is built on top of Matplotlib and offers a high-level interface for creating attractive statistical graphics.

Importing Matplotlib & Seaborn :

```
Import matplotlib.pyplot as plt
Import seaborn as sns
```

Creating basic plots with matplotlib

```
# Creating a simple line plot
x = np.arrange (0, 10, 0, 1)
y = np. sin (x)
plt.plot (x, y)
plt. xlabel ('x-axis')
plt. ylabel ('y-axis')
plt. title ('sine wave')
plt. show ()
```

Creating Seaborn plots:

```
# creating a scatter plots with seaborn sns.scatterplot
(data = df, x = 'Age', y = 'Income')
plt.show()
```

Customizing Plots :-

matplotlib and seaborn allow extensive customization of plots, including colors, labels, legends & more.

Data Preprocessing

Data Cleaning is a crucial step in the data Processing phase. It involves identifying and rectyfying issues, such as errors, inconsistencies and missing values in the data set. Missing values can arise due to various reasons, including data errors, sensor malfunctions, or simply because certain information wasn't collected. Handling missing data is essential because many machine learning algorithms can't work with it directly.

One Common approach to handling missing values is imputation, which means filling in the missing values with estimated or calculated values. For numerical data, you can use measures like mean, median or mode to replace missing values.

However, it's crucial to consider the nature of the missing data. If data is missing at random, simple imputation methods like mean or median replacement may work well. Still, if there's a pattern to the missing data, more advanced techniques like regression implutation or using machine learning models to Predict missing values may be necesary.

Additionally, in some cases, it might be appropriate to remove rows with missing values, but this should be done Cautionsly, as it can lead to a loss of valuable information.

Features scaling & Normalization

Feature scaling & Normalization are crucial preprocessing steps, especially when working with machine learning algorithms that are sensitive to the scale of input features. These techniques ensure that all features contribute equally to the model's Performance, Preventing one feature from

dominating the other's due to its longer magnitude.

Scaling typically involves transforming features to have a common scale, such as between 0 to 1, or -1 and 1. This can be achieved using methods like min-max Scaling or z-score. Standardization.

Normalization, on otherhand, aims to scale features to a standard distribution often a Gaussian distribution with mean 0 & standard diviation 1. This transformation can be particularly useful for algorithms like Principal component Analysis (PCA) that assume normally distributed data.

Example:

Imagine you have a dataset of product reviews Each review has two features the length of the review & the number of positive words used. If you want to not normalize these features, the length of the review, which can be large, might dominate the impact on the model compared to the number of positive words. By applying normalization, you ensure

that both features contribute equally to the model's decision-making process.

Handling Categorical data :

Categorical data represents discrete categories or labels rather then contiguous numberical values. Machine learning algorithms often require numerical Input, So handling Categorical data is crucial. There are two primary approaches for encoding categorical data is crucial. There are two primary approaches for encoding Categorical data :

1) label encoding
2) One-hot encoding

1) label encoding :

label encoding assigns a unique integer to each category. For example. If you have a 'size' column with categories 'small', 'medium' and 'large'. you can encode them as 1.2, 3 resp.

2) One-hot encoding

→ One-hot encoding, on other hand, creates binary columns for each category and assigns a 1 or 0 indicates the presence or absence of that category.

Considered a dataset of car attributes, including a 'car type' column with values like 'SUV', 'Sedan' and 'Convertible' one-hot encoding would creates seprate binary columns for each car type, making it easier for machine learning models to work with this categorical data.

Supervised Learning

Introduction to supervised learning :

Supervised learning is a fundamental machine learning Paradigm where a model learns to make prediction or classifications based on labelled data. In this approach, the algorithm is provided with input data and corresponding targets labels, enabling to generalize and make predictions on new, unseen data.

The primary objective is to develop a model that can genelize from this labeled data to make accurate predictions or classifications on new, unseen data. Supervised earning is widely employed in various domains, such as NLP, Computer vision, healthcare, finance & more. It forms the foundation of many predictive & decision-making systems. The process typically involves selecting an appropriate algorithm, training the model on a labeled dataset, & then evaluating its performance using metrics like

accuracy, precision, or mean squared error.

Linear Regression:

Linear Regression is a fundamental supervised learning algorithm used for solving regression problems, where the goal is to predict a continuous target variable. The core idea behind linear regression is to model the relationship between the depedent variable (for target) & one or more independent variables (or features) as a linear equation.

The linear equation for simple linear regression is typically represented as:

$$y = mx + b$$

where,

y is the dependent variable (the one you want to predict)

x is the independent variable

m is the slope of the line

b is the y-intercept, indicating the value of y when $x = 0$

Example:

Suppose you want to predict a person's weight (y) based on their height (x). Using linear regression, you collect data on the heights & weights of several individuals. The linear regression model learns to fit a line to this data that best represents the relationship between height & weight. It finds the slope (m) & y-intercept (b) values for the eqn $y = mx + b$. This can then be used to predict a person's weight based on their height.

Logistic Regression:

Logistic regression is a supervised learning algorithm used for binary & multi-class classification problems, as opposed to linear regression, which is used for regression tasks. logistic regression models the uses the logistic function (also known as the sigmoid function) to make predictions. The logistic function takes any real-valued no. & maps it to a value between 0 & 1, which can be interpreted as a probability.

The equation for logistic regression is:

$$P(Y=1) = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+\cdots b_nx_n)}}$$

where,

$P(X=1)$ is the probability that the dependent variable Y belongs to class 1. $X_1, X_2, \ldots X_n$ are independent variables. b_0 is the intercept term & $b_1, b_2 \ldots b_n$ are the coefficients of the independent variables.

Example:

Consider an email classification task, where you want to determine whether an incoming email is spam (Class 1) or not spam (Class 0). Features such as the emails subject, sender & content can be used as independent variables. After training a logistic regression on a dataset of labeled emails, the model can predict the probability that a new email is spam. If the predicted probability is greater than 0.5, it's classified as spam; otherwise it's classified as not spam.

Decision trees & Random forests :

Decision tree :

Decision tree is a versatile supervised learning algorithm used for both classification & regression tasks. It's a graphical representation of a decision-making process where the data is split into smaller subsets based on features. The goal is to create a tree structure where each internal node represents a features, each branch represents a decision rule, & each leaf node represents a class label or numerical value.

Random Forests :

Random forests are an ensemble learning method that improves the accuracy & robustness of decision trees. Instead of relying on a single decision tree, random forests build multiple decision trees, each using a different subset of the training data & a random subsets of the features. The final prediction is made by aggregating the results of these trees, typically using majority voting for classification or averaging for regression. Random forest

reduce overfitting & increase the model's predictive power.

Super vector machines (SVM) :

Suppor vector machines (SVM) are a powerful and versatile supervised learning algorithm primarily used for binary & multi-classification tasks.

The main idea behind SVM is to find the optimal hyperplane in a high-dimensional space that best separates data points belonging to different classes. This hyperplane is chosen to maximize the margin which is the distance between the hyperplane and the nearest data points of each class.

1) Hyperplane
2) Support vectors
3) margin
4) Kernel Trick

K-Nearest Neighbors (K-NN) :

K-Nearest Neighbors, often abbreivated as K-NN is a simple yet effective supervised algorithm used

For both classification & regression tasks. Unlike many other algorithms, K-NN doesn't create explicit models during the training Phase. Instead, it Stores the entire dataset in memory and makes Predictions based on the similarity between the new dataset in memory & makes predictions based on the similarity between the new data Points & existing data points.

Unsupervised Learning

Introduction to unsupervised learning:

unsupervised learning is a branch of machine learning where the primary goal is to find patterns & structures in data without the use of labeled target outcomes. Unlike supervised learning where algorithms are trained on labeled data to make predictions, unsupervised

learning algorithms work with unlabelled data to discover inherent structures & relationships. Unsupervised learning particulary useful when dealing with large datasets, data exploration, and uncovering hidden insights.

There are two primary types of unsupervised learning techniques :-

Clustering:

Clustering algorithms group similar data points together based on some measure of clarity. The goal is to identify natural groupings in the data. Common clustering algorithms include K-means, Hierarchical clustering & DBSCAN. For example, clustering algorithms can be used in computer segmentation, where data points representing customers are grouped into segments based on their purchasing behaviour.

Dimensionality Reduction :-

Dimensionality Reduction techniques aim to reduce the complexity of data while preserving its essential structure. Principal component Analysis (PCA) &

t-distributed stochastic Neighbour Embedding (t-SNE) are popular methods for this purpose. Dimensionality reduction can help with data visualization, feature selection & simplifying complex datasets for further analysis.

Examples of unsupervised learning applications.

Anomaly detection:

Unsupervised learning can be used to identify anamolies or outliers in a dataset, which can be crucial in fraud detection, network security & quality control.

Natural language processing:

Topic Modelling, a type of unsupervised learning, is used to identify topics within a collection of documents. It can be applied in areas like document clustering, content recommendation.

Clustering Algorithms (K-means, Hierarchical clustering, DBSCAN)

Clustering Algorithms :

Clustering Algorithms are a subset of unsupervised learning techniques used to group similar data points together based on some clustering algorithms, each with its own characteristics and use cases.

K-means Clustering :-

How it works:
K-means aims to partition data into K-clusters where k is a user-defined parameter. It works by iteratively assigning data points to the nearest cluster centroid and the recalculating the centroids as the mean of the points in each cluster. The process continues until convergence.

Hierarchical Clustering :

How it works:
Hierarchical clustering creates a tree-like structure (dendrogram) of data points by successively merging or splitting clustering based on similarity. It does not

require specifying the number of clusters in advance, making it a versatile method.

3) DBSCAN (Density - Based Spatial clustering of Application with Noise)

How it works:-

DBSCAN identifies clusters as regions of high data Points density seperated by regions of low density. It doesn't require specifying the number of clusters and can find arbitary - shaped clusters. It defines core points, which are data points with sufficient number of neighbours within a defined distance and then expands clusters by connecting core points.

Use cases:

DBSCAN is useful in anomaly function identifying spatial clusters in geographical data and grouping customers by geometric proximity.

Dimensionality Reduction (PCA):

Dimensionality Reduction (PCA - Principal component Analysis).

Dimensionality reduction techniques are on essential part of unsupervised learning, which aim to reduce the number of features or variables or in a dataset while preserving as much of the relevant important information as possible.

PCA works on:

1) Standardization
2) Covariance Matrix
3) Eigen value decomposition
4) Principal component.

ML in Real time:

Machine learning in real-time processing data & makes decisions instantly, essential for applications like online fraud detection, personalized recommendations, and autonomous driving challenges include maintaing low latency. Scalability data freshness and efficient resource management. The techniques involves learning (continuous model updates) batch processing (periodic updates) and steam processing frameworks (Apache Kafka, Apache Flink, Apache spark streaming).

ROC & AUC

The Receiver operating characteristic (ROC) curve plots true positive rate against false positive rate, showing a model's diagostic ability. The Area under the ROC curve (AUC) quantifies overall performance with, higher AUC indicating better discrimination between classes. ROC & AUC aid comparing models & selecting optimal thersholds.

Confusion Matrix :

A Confusion matrix evaluates classification performance, Showing true positive, true negatives false positive and false negatives. It allows calculation of metrics like accuracy, precision, recall, and F1 score. helping identify model strengths & areas for improvement.

MSE & MAE

Mean Squared Error (MSE) & Mean Absolute Error (MAE) evaluate regression models. MSE measures average squared prediction errors, sensitive to outliers. MAE measures average absolute prediction errors, less sensitive to outliers. Both metrics assess model accuracy & guide improvements.

Deep Learning

Introduction to Deep Learning:

Deep learning is a subset of machine learning that uses neural networks with many layers (deep neural networks) to model complex patterns in data. It excels in tasks such as image & speech recognition, natural language processing, & autonomous driving. Deep learning algorithms require large amounts of data & significant computational power but can achieve high accuracy by automatically learning features from the raw data.

Introduction to TensorFlow & Keras

TensorFlow is an open-source deep learning framework developed by Google, designed for numerical computation and large-scale machine learning. Keras is a high-level

neural networks API, written in python and capable of running on top of tensorflow. Keras simplifies the process of building and training deep learning models, offering user-friendly APIs & extensive documentation, making it a popular choice for beginners & researchers alike.

Neural Networks & their Applications:

Neural networks are computational models inspired by the human brain, consisting of interconnected nodes (neurons) that process data in layers. They are used for a variety of applications, including image & speech recognition, language translation, & predictive analysis. Neural networks learn to perform tasks by adjusting the weights of the connections based on the data they are trained on.

Activation Functions : Sigmoid, Hyperbolic Tangent, ReLU

Activation Functions introduce non-linearity into neural networks, enabling them to learn complex patterns. The sigmoid function outputs values between 0 & 1, useful for binary classification. The hyperbolic tangent (tanh) function output values between -1 & 1, often used in hidden layers to center data. ReLU (Rectified Linear Unit) outputs the input directly if positive, otherwise, zero is widely used for its simplicity and effectiveness in deep networks.

Training Methods for Neural Networks

(High-Level overviews)

1) Update of weights with single Training Set Element (Stochastic Gradient Descent - SGD):

weight are updated after each training example. This method can converge quickly but is often noisy.

2) Batch training: weights are updated after processing the entire training dataset. This method provides stable convergence but can be slow & memory-intensive.

3) Mini-batch training: A compromise SGD and batch training, weights are updated after processing a small subset of the training data. This method balances convergence speed & stability.

L1 & L2 Regularization

L1 & L2 regularization are techniques used to prevent overfitting by adding a penalty to the loss function.

L1 Regularization:

Adds the absolute value of the weights to the loss functions, promoting sparsity by driving many weights to zero, which can lead to simpler models.

L2 Regularization:

Adds the squared value of the weights to the loss functions, discouraging large weights & promoting weight decay, which helps improve the model's generalization.

PyTorch Framework

Pytorch developed by Facebook's AI Research lab, is an open-source deep learning framework. It features a dynamic computational graph, ideal for research and prototyping. Known for its ease of use, Python integration and strong community support, PyTorch includes the Autograd package for automatic differentation and the torchvision library for image processing tasks.

Convolutional Concept:

Convolutional Neural Network (CNNs) process grid-like data, such as images. Convolutional layers use sliding filters to produce feature maps, capturing pattern like edges & textures. Pooling layers reduce spatial dimensions controlling overfitting & computational load.

Transfer learning:

Transfer learning uses a pre-trained model on a large dataset, fine-tuned for a specific smaller dataset. This approach is useful with limited new data, reducing training time & improving performance by leveraging learned features from the pre-trained model.

Data Augmentation:

Data Augmentation increase training data diversity by applying transformation like rotations, filps & color changes to existing data. This improve model robustness & generalization, preventing overfitting by exposing the model to varied inputs.

Object Detection:

Object detection identifies and locates objects in an image, providing bounding boxes and labels for each object. Unlike image classification, it detects multiple objects per image. Popular architectures include R-CNN, Fast R-CNN, Faster R-CNN, and YOLO.

YOLO Algorithm:

YOLO (you only Look once) is a fast and accurate object detection algorithm. It treats detection as a single regression problem, predicting bounding boxes & class probabilities from entire images in one evaluation. YOLO's real-time detection speed suits applications needing quick & accurate object recognition.

RNN Concept:

Recurrent Neural Network (RNNs) are designed for sequential data. They maintain a hidden state that

Captures information from previous inputs, modelling temporal dependencies. RNNs are used in tasks like language modeling, machine translation and time series prediction. They process inputs of variable length by sharing parameters across time steps.

Types of RNNs:

1) Standard RNNs: Use the same network structure at each time step & share parameters accross the sequence, suitable for simple sequential tasks.

2) Bidirectional RNNs (Bi – RNNs): Process data in both forward & backward directions, capturing content from both past & furture states. Effective for tasks requiring content for from both sides, such as natural processing language.

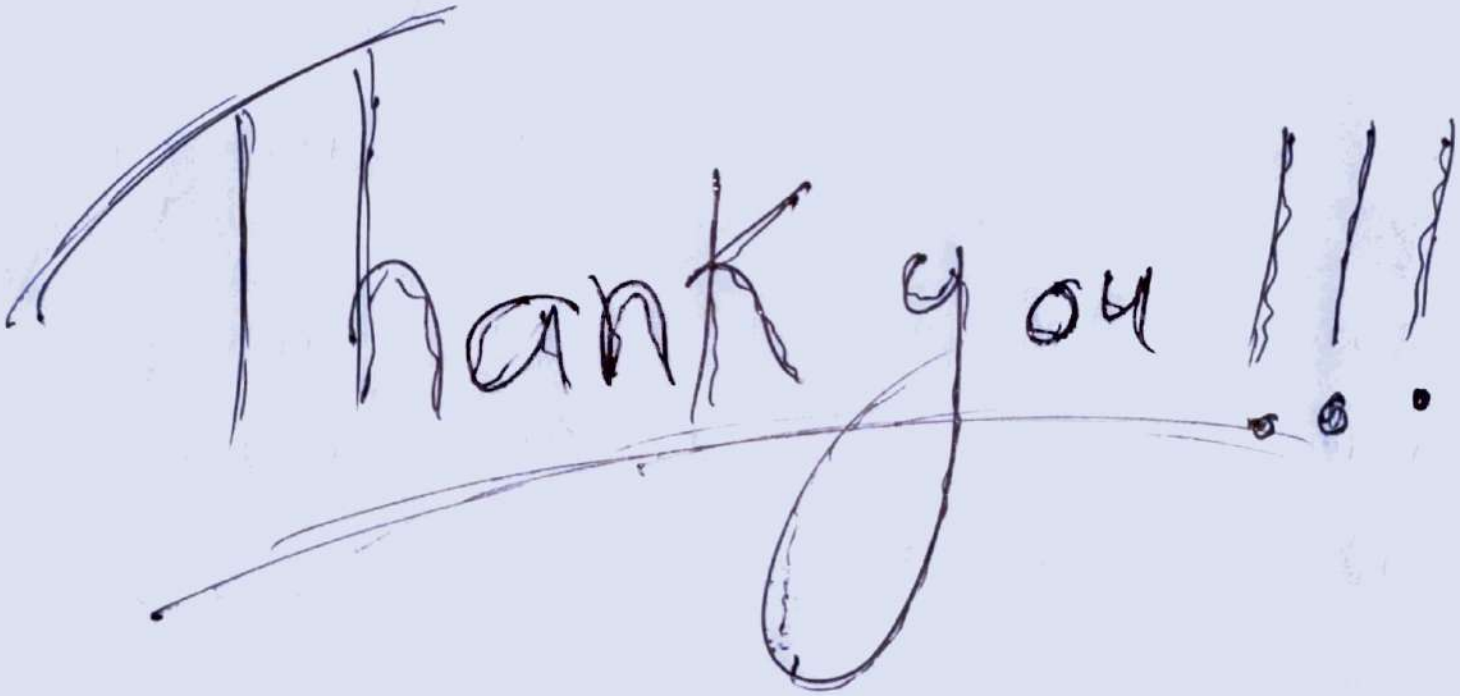
3) Deep RNNs: Stack multiple RNN layers to learn complex representations, capturing hierarchical patterns in sequential data.

Vanishing Gradients with RNNs:

The vanishing gradient problems occurs when gradients become very small, causing slow or halted learning. In RNNs, repeated multiplication of gradients through time steps exacerbates this, making it difficult to learn long-term dependencies.

Long Short-Term Memory (LSTM):

Long Short-term Memory (LSTM) Networkes are RNNs designed to mitigate the vanishing gradient Problem. They have a cell state and three gates: input gate, forget gate, & output gate. The cell state carries long-term information, while the gates control the flow of information into and out of the cell state. This architecture enables LSTMs to learn & retain long-term dependencies, making them effective for tasks like language modeling and time Series Prediction.


Thank you !!!