**Text

Description automatically generated with low confidenceDocument type**

CONFIDENTIAL | AUTHORIZED HPE PARTNER USE ONLY

Multipurpose Template Headline text Arial Bold between 32-40 pt

Subtitle Arial regular 22 pt

**Add a photo (7.27” x 3.5”) from the brand-approved image library that is related to the document content. Or, please remove the FPO box and leave this section blank.**

Contents

[Chapter 1 Python Essentials 3](#_Toc92818810)

[1.1 Introduction: 3](#_Toc92818811)

[1.2 Python Essentials for Data Science & ML: 3](#_Toc92818812)

[1.3 Data Science 4](#_Toc92818813)

[1.4 Data Preparation 6](#_Toc92818814)

[1.5 Data Cleaning 13](#_Toc92818815)

[1.6 Dataset Generation and Import 16](#_Toc92818816)

[1.7 Data Visualization 17](#_Toc92818817)

[1.8 SPARK OVERVIEW 31](#_Toc92818818)

[Chapter 2 Machine Learning Lifecycle 35](#_Toc92818819)

[2.1 Introduction to Machine Learning 35](#_Toc92818820)

[2.2 Machine Learning task 45](#_Toc92818821)

[2.3 Sample Use Cases 65](#_Toc92818822)

[Chapter 3 Deep Neural Network 68](#_Toc92818823)

[3.1 Overview: Neural Networks 68](#_Toc92818824)

[3.2 Introduction to DNN 80](#_Toc92818825)

[3.3 CNN 82](#_Toc92818826)

[3.4 Recursive Neural Networks 87](#_Toc92818827)

[3.5 LSTM (Long Short-Term Memory) 89](#_Toc92818828)

[3.6 Transfer Learning 90](#_Toc92818829)

[3.7 Model Serving: Real-Time deployment 93](#_Toc92818830)

[3.8 Use Cases 94](#_Toc92818831)

[Chapter 4 Distributed Deep Neural Network 96](#_Toc92818832)

[4.1 Introduction 96](#_Toc92818833)

[4.2 Parallel and Distributed Techniques for Deep learning 96](#_Toc92818834)

[4.3 Architectures for distributed deep learning 97](#_Toc92818835)

[4.4 Training in distributed deep learning 100](#_Toc92818836)

[4.5 Use cases 100](#_Toc92818837)

[4.6 Aggregation of parameters in Distributed Deep learning 101](#_Toc92818838)

[4.7 Evaluation of distributed deep learning models 103](#_Toc92818839)

[4.8 Inferencing distributed deep learning models 104](#_Toc92818840)

[4.9 Advantages of Distributed Deep learning 105](#_Toc92818841)

[4.10 Applications of Distributed Deep learning 105](#_Toc92818842)

# Chapter 1 Python Essentials

## 1.1 Introduction:

Before Python, we had many other languages like C, C++, Java. These programming languages are used to develop, deploy, and maintain an application. These traditional languages consumed lots of users’ time as well as the length of code will be huge, and this created an issue regarding controlling such a big code. Users did not find it as a user-friendly language because it was hard to read and understand the syntax. Let’s suppose if any technical issues arise it was very difficult to figure out where that issue is and then very difficult to fix the issue. It’s harder for developers to understand as these things are new for that developer. So, looking at these issues Python came into the picture where it tried not to be simpler in terms of syntax but also it becomes more user-friendly and to solve a particular problem. We had to type a smaller number of codes which saved time and was very efficient. Because of these reasons, Python became more popular and is used worldwide.

## 1.2 Python Essentials for Data Science & ML:

As there are many programming languages that people use in their day-to-day life. Python turns out to be the best when it comes to Data Science. There are three best and crucial libraries for Data Science those are NumPy, Pandas, and Matplotlib.

There are many important libraries for doing Data Science stuffs, here we will name a few.

* NumPy — A library that makes a variety of mathematical and statistical operations easier; it is also the basis for many features of the Pandas library.
* Pandas — A Python library created specifically to facilitate working with data, this is the bread and butter of a lot of Python Data Science work.
* Matplotlib — A visualization library that makes it quick and easy to generate charts from your data.
* Scikit-learn — The most popular library for machine learning work in Python.
* NumPy and Pandas are great libraries for exploring and playing with data. Talking about Matplotlib it’s a data visualization library that generates beautiful graphs to understand the trends in data.

Let’s do things step by step:

1. Install Anaconda from here. It’s packed with many libraries to handle Data Science, Python, ML workloads.
2. Learn Python Fundamentals: Before using Python for Data Science, it’s better to start with the basics of Python which will help Data scientists to build custom logic solutions for a given problem.
3. Study Regular Expressions in Python: Regular Expressions help you to do lots of data cleaning, as data can be acquired from different sources specific to your business needs especially when data is in form of text.
4. Understand NumPy, SciPy, Matplotlib, and Pandas:

* NumPy - It will help you to do complex and different types of mathematical operations on arrays.
* SciPy – It is used to help detect patterns in big sets of data.
* Matplotlib – It will be helpful to visualize data.
* Pandas – It is for pulling, transforming data, and performing imputation of data.

1. Powerful Data Visualization: There are different and beautiful libraries you can use for data visualization like Plotly, Bokeh, Seaborn, etc.
2. Learn Scikit-learn and Machine Learning: It’s the widely used library on Python to do Machine learning.

## 1.3 Data Science

In today’s world, we generate lots of data every day and make this data into valuable or useful form Data Science knowledge is essential.

Data Science is a field that clubs different domains like Programming, Statistics, Mathematics to create meaningful data. It is a process of gathering, pre-processing, massaging, scrutinizing data so it can be used further to do analysis or train a Machine Learning model.

This is where Data Science comes into the picture helping us to transform data from zero to hero finding out patterns that will help to make the right business decision or creating AI-based applications like Siri, Alexa, Grammarly, and many more.

Data Science is such a vast field, and it touches every domain that generates data. It helps marketize products well as it helps to target the right people who could buy the company products. Health Care Industries are one such area where large data needs to be analyzed accurately to predict the early stage of a disease in a person and take appropriate measures to cure them. Another example where it’s massively used is the Automobile industry. It checks if a vehicle requires a service, provides autonomous driving. Now, it’s time for us to understand the steps that we take as Data scientists to build a solution. Let’s check that out in the below sections.

### 1.3.1 Data Acquisition

It is the process of collecting data that is generated by a process outside the organization or inside to help your business requirement.

It is a process of identifying, understanding, accessing, sourcing, and ingesting such data which is required to build your ML models.

There are 2 things that confuse people a lot which are Data Acquisition and Data Ingestion.

1. Data Acquisition: It is something that is generated by your organization rather than the data which you fetch from outside.
2. Ingestion: It is a process to capture data from outside the organization. It is not done widely compared to Data Acquisition. Almost Data Acquisition should be 70% and Ingestion should be 30%.

Finally, one of the most important features is that the end-to-end process is planned out doing many possible variations. It could be from open data sources or configuring tools to scan internet sources or hiring someone to aggregate the required data. Depends on Business-to-Business.

## 1.4 Data Preparation

It is a process of cleaning and transforming raw data prior to processing and analysis. It is an important step before reformatting data, doing corrections to data, and clubbing relevant datasets to enrich data.

Data Preparation is one of the time-consuming processes for data professionals or business users, but at the same time, it’s essential as it helps you to draw meaningful insights and discard the biases present in your data coming from low quality of data.

This process includes enriching source data, standardizing data formats, removing outliers, understanding data like what has to be added or removed so it is understood by a larger audience.

In this process to handle outliers, you need Python where Pandas, NumPy, matplotlib, seaborn come into the picture this Python library will help you to analyze, understand data and perform necessary actions on it.

Now, when you are done preparing the data, you can store it in a data warehouse or third-party application like a business intelligence tool.

Let’s talk about some key benefits of data preparation.

1. When you receive that data, you don’t have to invest time in doing cleaning or locating it. You just must analyze it and depend on your needs transform the data that’s it. It’s like cooking, tasty food served to you know it’s totally your choice how to eat it.
2. You maintain good quality standards to make reliable results for Machine Learning (ML), Business Intelligence (BI), Predictive Analysis.
3. Identify and fix data issues that were not detected previously.
4. It’s a cost-effective and efficient way to prepare data for analysis.
5. Can make more informed business decisions before executives have access to that data.
6. You can earn more business value or Return of Investment from BI and analytics initiatives.

Let’s take an example, we have a titanic dataset where we are trying to solve classification

problem predicting whether the person will survive or not based on past data.

FLOW OF PREDICTION PROCESS:

1. Load the dataset which is titanic dataset
2. Visualize the data.
3. Convert to their respective variable type.
4. Apply featurization

Code:



Calendar

Description automatically generated

Figure 1.1: Sample dataset

Now we will see what types of data the dataset holds

Code:



A picture containing text, person

Description automatically generated

Figure 1.2: Dataset Schema

It’s clearly visible that the pyspark SQL data frame has converted by default everything to string so we need to identify which all features need to be converted to Integers and float.

From the dataset, we can figure out that PassengerId, Survived, Pclass, Age, SbSp, Parch should be converted to Integer and Fare as float

Code:



A screenshot of a computer

Description automatically generated with low confidence

Figure 1.3: Dataset Schema: Conversion

Now if we observe Parch and SibSp can be combined as they can come under family. So we combined both the features to Family

Code:

## new column combining two columns into one

df4 = df4.withColumn('Family',df4.SibSp + df4.Parch)

df4.show(10)

Table

Description automatically generated

We go for Standardizing or Normalizing the dataset to bring them into the same scale if their measurement units are different. Generally, Standardization works well than Normalization.

Few scenarios where Standardizing or Normalizing data involves is

1. In finding Euclidian Distance
2. Calculating Gradient Descent in Deep Learning Techniques

Normalization (Min-Max Scalar): we try to scale down features between 0 to 1

A picture containing text

Description automatically generated

Standardization (Standard Scalar): All features are transformed in such a way that they will have properties of standard normal distribution where mean = 0 and standard deviation =1.

A picture containing text, clock

Description automatically generated

Where µ (mu) is mean, x represents datapoint and σ ( sigma) is standard deviation.Using the same titanic dataset, where we would just use two features to demonstrate Min-Max Scalar and Standard-Scalar.

Code:



A picture containing shape

Description automatically generated

Figure 1.4: Dataset schema column merge

Code



Table

Description automatically generated

Figure 1.5: Standard Scalar

Data Preparation can be advantageous in big data environments with data lakes often build around Hadoop clusters, which stores large amount of structured, semi-structed data in a raw form. In many big data applications, data preparation is largely an automated task: Machine learning algorithms can speed things up by examining data fields and automatically filling in blank values, fixing errors or renaming fields to ensure consistency when data sets are being joined.

## 1.5 Data Cleaning

It is a process of removing corrupted, incorrect, duplicate, or incomplete data from the dataset. Data Cleaning is done when you have clustered multiple data sources, there are times when you will see data that is irrelevant or missing or misplaced there could be times where you would find data is mislabeled.

There are no steps as such when it comes to Data Cleaning as it will vary from dataset to dataset. But it’s important to establish a template for your data cleaning process to check that you are doing the right things.

You do two things in Data Cleaning

1. Dropping: Drop the data that has missing values as it will mislead the machine learning models.
2. Imputing: if you know the missing values are less in number or you know that there should be no missing values in the dataset after speaking to business experts that are the time where you do imputations. Some of the common imputation techniques include Mean, Mode, Median Imputation also you can use ML Algorithm to predict those missing values to get better accuracy at a later point in time.

Let’s take the same titanic dataset which we used in an earlier section and try to check null values are present and do some feature engineering on it so as to get good accuracy from the machine learning model.

To check null values in spark sql data frame we are going to use functions like count(), isNull()

Code:



A screenshot of a computer

Description automatically generated with low confidence

Figure 1.6: Null Values

It’s clearly visible that we have many nan values for Age, Cabin, and Embarked column.

Let’s try to do some imputation on these features.

We will try to impute Age feature with its median value why because age cannot be a floating number, so we won’t go for mean here.

Code:



Table

Description automatically generated

Figure 1.7: Imputation

Now we will try to impute Embarked column with most frequent data

Code:

df4 = df3.na.fill({'Embarked': 'S'})

df4.select([count(when(col(c).isNull(), c)).alias(c) for c in df4.columns]).show()

Diagram

Description automatically generated with low confidence

Figure 1.8: Imputation: Frequent Data

This is how we do Imputation using pyspark.sql data frame.

After data preparation, we remove features that are not relevant to the train machine learning model. Just to build a strong relationship with the data points.

Code:



Table

Description automatically generated

Figure 1.9: Drop: Irrelevant feature Column

You won’t see the features that are dropped now.

This is how we do Data Cleaning generally. After cleaning and preparing the data, we train the model by inducing them to machine learning algorithms which we will come to know by the end of this book. For now, we are going to talk about how we visualize the data and figure out what trend a particular graph is trying to show us. Deciding on which graphical representation to be used for a particular dataset will be described in the following sessions.

## 1.6 Dataset Generation and Import

Usually, we get datasets from open-source or from cloud services. We can use this in the initial stages of our development by importing them to your coding environment. Sometimes, there can be situations where the existing data is not sufficient to derive conclusions. In these scenarios, we need to generate data based on the samples provided. Later this data set can be used for training Machine Learning models. More often we prefer using Jupyter Notebook or PyCharm. But nowadays, most organizations prefer working on cloud services such as Azure, GCP, AWS.

## 1.7 Data Visualization

Data Visualization is as the name tells, it is a pictorial representation that helps us to understand the trends or patterns in the data by using visualization elements like charts, graphs, pie-charts.

In today’s era, we are generating a huge amount of data to analyze huge amounts of data we use data visualization tools, and technologies and make data-driven decisions.

It is used in multiple areas such as medical conditions, Business growth, weather trends or it could be mathematical relationships. Data visualization is used mainly for data checking and cleaning and communicating results to stakeholders.

Now we are going to see in detail with the help of Python libraries and how it helps in a Data Science process flow.

1. Line Chart:

Line Chart is a simple data visualization in Python, which is accessible under Matplotlib.It is used to show the relation between two data X and Y on the respective axis. Let’s check that out with examples

Text

Description automatically generated

Chart, line chart

Description automatically generated

Figure 1.10: Matplotlib: Line chart

1. Histogram:

The Histogram is the graphical representation of a set of numerical data distribution across. It is a kind of bar plot with the X-axis and Y-axis representing the bin ranges and frequency respectively.



Chart, histogram

Description automatically generated

Figure 1.11: Matplotlib: Histogram

Features of Histogram

* measured on an interval scale of given numerical values with several data bins.
* It is used to get any abnormal observations in the given dataset.
* The X-axis represents data distributions
* The Y-axis represents the number of % of occurrences in the data.

1. Dis Plot: This is corresponding to the histogram in the graphical, but with additional features. And bringing Kernel Density Estimation (KDE).



Chart, histogram

Description automatically generated

Figure 1.12: Dis plots

1. Joint Plot: A combination of scattering and histogram.



Chart, scatter chart

Description automatically generated

Figure 1.13: Joint plot

1. Pie Chart:

This is a very well-known chart and representation statistical plot in the form of circular from a series of data. This is frequently used in many business presentations to represent Profit, Loss, Sales, Order, etc., It consists of a portion of data part in the collection of the same set and character-wise differentiation. Every slice of pie is called a wedge with values of different sizes.

This chart is widely used to represent the composition collection**.** Perfect for the categorical data type.



Chart, pie chart

Description automatically generated

Figure 1.14: Matplotlib: Pie Chart

To highlight a particular pie-chart:



**Chart, pie chart

Description automatically generated**

Figure 1.15: Matplotlib: Pie Chart

1. Area Plot:

This is very alike to a line chart with fencing surrounded by a boundary line of different colors. Plain representation of the evolution of a numeric variable.



Chart

Description automatically generated

Figure 1.16: Matplotlib: Pie Chart

1. Scatter Plots:

Scatter plots are used to plot data points across both axes (Vertical as well as Horizontal) and represent how each axis correlated with each other. Mostly in Data Science /Machine Learning implementation and before the EDA process, generally we should analyze what kind of trend it is trying to show. It could positive or Negative or sometimes be scattered across the graph. When it is scattered that means it is not showing any relation.

Linear graph: A graph that shows positive or negative trend in form of a straight line.



Chart, scatter chart

Description automatically generated

Figure 1.17: Linear - Scatter plot

Non-Linear Graph: Type of graph which shows no trend or pattern is termed as Non-Linear graph

Chart, scatter chart

Description automatically generated

Figure 1.18: Non Linear - Scatter plot

1. Hexbins Plots:

The objective of Hexbins is used to club the two sets of numeric values. Hexbins helps to improve the visualization of the scatter plots as in larger dataset, a scatter plot makes a confused smattering of points. We can improve this with Hexbins. It provides two modes of representations

1. List of Coordinates
2. Geospatial Object.



Background pattern

Description automatically generated

Figure 1.19: Hexbin Plot

1. Heatmap:

A heatmap is a set of variable correlations is represented by various shades of the same color. Lighter shades of the graph show lower correlations than the higher shades. This map help Data Scientists to check out how target variable is correlated with other dependent variables in the given data set. Less correlated variables can be removed for further analysis, we could say this helps us during the feature selection process. Later grouping them under X, Y as our target and followed by test and train split.



Chart

Description automatically generated

Figure 1.20: Heatmap

Box Plot:

A Boxplot is a type of chart repeatedly used in the Data Science life cycle, especially during Explanatory Data Analysis (EDA). Which represents the distribution of data in the form of quartiles or percentiles. Q1 represents the first quartile (25th percentile), Q2 is the second quartile (50th percentile/median), Q3 represents the third quartile (Q3) and Q4 represents the fourth quartile or the largest value.

Using this plot, we could find the outliers very quickly and easily. This is a very effective plot all among the plots. So, after the removal of outliers, the data set needs to undergo some sort of statistical test and fine-tune for further analysis.



Chart, box and whisker chart

Description automatically generated

Figure 1.21: Boxplot

1. Pair Plot:

A pair plot is another important plot in the Data Science life cycle during the EDA process, to analyze how features are related to each other, in the form of grid-based miniature graphical representation along the X and Y axis, either positively correlated or negatively correlated. So obviously we could eliminate the negatively correlated, by considering positively corrected pairs and moving for further analysis. This is very similar to Heat Map, but here we could see the relationship with our naked eyes. Again, this is best for doing the feature selection process.



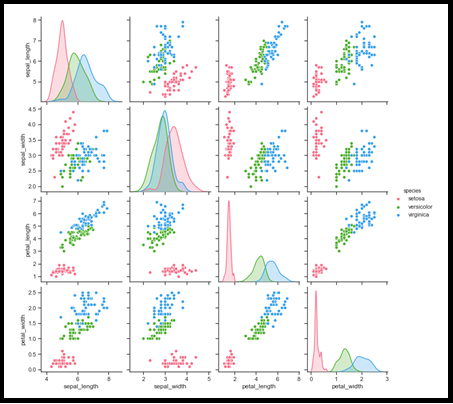


Figure 1.22: Pair Plot

Line Chart is always linear relation between X and Y axis, we observe that above picture

1. Bar Chart:

A bar chart is generally a very well-known chart to presents categorical data with rectangular bars. It can be plotted either way vertically or horizontally. This chart would show the impact of the gender over fare on the given titanic dataset. In the below chart “female” had higher fare rates than” male” which is clearly visible from bar graph. This would derive some observation on the dataset and focus on the problem statement.

**Chart, box and whisker chart

Description automatically generated**

Figure 1.23: Bar Chart

**Univariate – Bi-variate and Multivariate Analysis**

Analysis of variants in Data Science process, it could be Univariate (or) Bi-variate (or) Multivariate.

* Univariate: Considering only one variable at a time.
* Bi-variate: comparing two variables.
* Multivariate: comparing more than two variables.

## 1.8 SPARK OVERVIEW

Chart, shape

Description automatically generatedThe solution for big data analysis is Apache Spark. It was developed at University of California, Berkeley in 2009. It was developed for the faster processing of large data in terms of petabytes. They can include batch processing, stream processing, and interactive processing of data. The reason for faster computation is the use of the in-memory cluster computing feature. They run on memory which makes them 100 times faster when compared to Hadoop. Also, the cluster computing feature enables the Resilient Distributed Datasets (RDD) which makes it faster processing. RDD is nothing but logically dividing the datasets for parallel processing at different nodes of the cluster.

Diagram

Description automatically generated

Figure 1.24: Spark executes a job in the following way:

A picture containing text, sign, screenshot

Description automatically generatedThe dataset can be any format that obeys to Hadoop file format. The given dataset is divided into partitions of data. This is transformed and given as output when only requested for action by the driver program. The transformation is applied with reference to some base dataset and follows the same procedure. Each time an action is requested the RDD is recomputed. The RDD can be stored in memory or hard disk based on the rate of the query. The RDD is stored as a data frame with named columns.The DataFrame API developed by Spark enabled the usage of DF in Java, Scala and Python.The process behind this fast processing is the catalyst who just optimizes the data and selects a plan for RDD which the front end user is not aware off.

Figure 1.25: Components of Spark:

The core engine.

Spark SQL: A module for structured data processing.

Spark Streaming: This extends the core Spark API. It allows live data stream processing. Its strengths include scalability, high throughput, and fault tolerance.

MLib: The Spark machine learning library.

Graphical user interface

Description automatically generated with medium confidenceGraphX: Graphs and graph-parallel computation algorithms.

Apache Spark For Deep Learning:

As Deep Learning involves lots of computations and many layers with a feedback mechanism, Spark is one of the best methods for faster computations. Databricks have developed libraries called Deep Learning pipelines that provide API in python for Apache Spark. Since these libraries are created by the contributors of Apache Spark, it has an added advantage to be merged and released as official API. Also comparatively it provides ease of integration without compromising the performance in just few lines of code. Since it’s written in Python it easily gels with DL libraries such as Keras and TensorFlow.

Hence all the steps of Deep Learning is done in few line lines of code. It works fine for CNN, RNN and LSTM networks. The steps are as follows for reference

* Image loading
* Applying pre-trained models as transformers in a Spark ML pipeline
* Transfer learning
* Applying Deep Learning models at scale
* Distributed hyperparameter tuning
* Deploying models in Data Frames and SQL

How to use Apache Spark for Deep Learning:

* The command *spark.readImage* reads images in formats such as png, jpeg, etc. and stores them in Data Frame as rows in the predefined schema.
* Sample Ratio is a parameter that decides a ratio of images to be tested first before applying to the entire dataset.
* Deep Learning Pipelines offer transfer learning in a much faster way with its built-in utilities.
* The features are extracted using a small set of images and this is combined with the Deep Learning algorithm to train the model. For example, the feature extraction can be done with logistic expression and these features can be combined with the Inception V3 deep learning algorithm.

# Chapter 2 Machine Learning Lifecycle

## 2.1 Introduction to Machine Learning

Machine learning technique uses predefined algorithms to analyze provided data and to predict the outcome of unanswered data. In conventional programming we have to implement the logic to handle the incoming data based on the predefined rules. The programmer needs to explicitly include the rules for data handling. The machine learning approach is different, where the programmer need not worry about rules, instead collect as many data as possible to make the machine to learn the rules. For example, consider a situation where we need to detect hand sign language using a computer vision application and you are going to design the program. If you are using conventional approach you need to design the algorithm that how the system deal with the input image of hand sign, it may be first detect the edges, then compare edge with an edge image in the database, etc. But if you are using machine learning you need to collect as many images as possible, then you are going put the correct answer (we can call it as ‘label’) for each image, now we are ready with the labelled dataset, next we are going to choose a machine learning model based on our requirement. Next step is to use the labelled dataset to teach the model. The machine will process each data and update internal parameters according to the dataset. This process of updating the parameters based on dataset is known as training. To measure the level of learning of the machine we use different metrics. If we achieve metric value of our requirement, we can stop training and save the model. This saved model can then be used for deployment, and it can detect any hand signs. So major difference lies in the way it is processing data. The machine learning definitely have the adaptability, flexibility and scalability than the conventional approach. The learning of machine here is similar to human learning.

Diagram

Description automatically generated

Figure 2.1: Learning vs. designing

### 2.1.1 Types of Machine Learning Algorithms

The machine learning algorithms can be classified based on the way machine is learning. If machine is using a labelled dataset for training means the answers are there for the ‘supervision’. Other kind of learning strategy is without labels is known as unsupervised learning. Based on the kind of problem also we can classify the machine learning problems into different kinds. If you are dealing with a problem where the model needs to predict a categorical value as a result. For example, if we need to check the incoming emails to our mailbox and predict the mail is spam or not. Here the output is limited to two categories, spam, and not spam. This type of machine learning workloads is known as classification problems. If you are dealing with the problems like predicting the average life expectancy of a person based on the key health parameters like blood pressure, blood lipid profile, etc. Then the result is a continuous value. This kind of scenario where machine learning is used for prediction of continuous output is known as regression. Usually both classification and regression models are using supervised learning techniques. The other kind of problems where we need to categorize the data into groups, we have to use unsupervised learning. Clustering is the type of machine learning where we use these techniques. Usually, clustering is used for grouping the data like customers with similar buying patterns can be grouped together to recommend new products to them. One of the major applications is together with classification. Using clustering the data that need to be classified will be categorized and labelled accordingly then the dataset will be applied to classification model.

Diagram

Description automatically generated

Figure 2.2: Machine Learning: types

#### 2.1.1.1 Regression

Regression models analyze the relationship between dependent and independent variables. It is powerful when the target of an independent variable shows a linear or nonlinear relationship between them. Regression algorithms usually determine the possible curve fitting between variables. This genre of algorithms is generally suitable when the target variable is continuous. Predict the fuel efficiency based on the vehicle condition indicating a man's age based on his features, predicting the time required for a taxi trip, etc are examples for regression analysis. Various regression algorithms proposed in the literature can be listed as,

1. Linear regression
2. Decision tree
3. Support vector regression
4. Random forest regression
5. Lasso regression
6. **Linear Regression**

The simple linear regression model has a fixed number of parameters depending on the input features, and it outputs a numeric prediction, such as the house price, insurance premium, or loan amount.

The linear regression algorithm is a process of fitting the data to a line. A general equation for linear regression is,

Here is the value we are predicting, n is the number of features of our data points, is the value of the th feature.are the parameters of the model and is the bias. In order to predict the premium of a health insurance service using linear regression, we need to determine the age and the weight of the applicant in the formula below.

Using this equation of linear regression, we can predict the premium for an unknown applicant. For that the first step is to train the model with the labelled dataset to determine the regression line. Training the model consists of the following steps: We must first choose a metric that tells us how well our model is performing by comparing the model's predictions made for premium in the training set with actual premiums. Metrics such as Mean Square Error (MSE) or Root Mean Square Error (RMSE) can be used. The model's parameters () are initialized usually randomly and the error is calculated over the entire training data set. In order to minimize this error, these parameters are modified iteratively. Gradient descent is one of the algorithms used for this. In a wide range of problems, Gradient Descent is an efficient optimization method. Basically, this method consists of iteratively tweaking the parameters of a model until an optimal set of parameters values is achieved that minimizes the prediction error made by the model. The parameter value varies in the direction of descending gradient. Algorithm will continue to do this until the parameter value reach the optimal value at the bottom of the curve. In linear regression the data will be fitted to the line that having this optimum parameter determined by training.

Chart, scatter chart

Description automatically generated

Figure 2.3: Datapoint: Training Dataset

The datapoints shown in the above figure is the training data. Linear regression algorithm determines the best fit line that have minimal error from the datapoints as shown in figure below. The further prediction will be carried out based on the regression line thus formed.

Chart, scatter chart

Description automatically generated

Figure 2.4: Linear Regression Line

#### 2.1.1.2 Classification

Classification algorithms are usually supervised machine learning algorithms to predict the class of the output variable. Hence the output of the classification problem will always be a categorical value. Based on the number of categories, the classification algorithms are binary classification, multi-class classification, etc. Examples of classification problems are predicting whether the email received is spam or not. Prediction of images into cat and dog, prediction of patient data into cancerous or not are examples for binary classification. Predicting the kind of wheat seed, predicting categories of food items are examples for multi class classification. There are numerous classification algorithms proposed in the literature.

1. Logistic Regression
2. K nearest neighbor
3. Decision trees
4. Random forest algorithm
5. Naïve Bayes Algorithm
6. Support vector machine

etc.

**Logistic Regression**

Logistic Regression is regression algorithm with the target variable as categorical variable. A Sigmoid curve, or S-shaped curve, is fitting to our observations in Logistic Regression rather than fitting a straight line like in linear regression. In essence, Logistic Regression models are binary classification models with the ability to determine whether a person is overweight based on their weight, or whether a house is large or small based on its size. Therefore, our data contains two kinds of observations (Category 1 and Category 2 observations), as we can observe in the figure below.

Chart

Description automatically generated

Figure 2.5: Logistic Regression: Sigmoid Curve Line

We can observe the sigmoid curve that is been used for fitting our data points. A 0 to 1 scale is used for the Y-axis. The reason is that the sigmoid function always takes these two values as maxima and minima, and this is how we classify samples into two categories. A probability of an observation belonging to one of the two categories can be calculated by computing the sigmoid function of X. Like linear regression, it is derived from the weighted sum of the input features. The sigmoid function

Consider the example where we are using logistic regression for predicting patient is diabetic or not. we would compute a weighted sum of a person's features before using sigmoid function to determine whether they are diabetic or not.

Training of Logistic regression is fitting of the sigmoid curve to the data points. The parameter values of the model (the weights) can either be calculated iteratively using Gradient Descent or by using more probabilistic methods such as Maximum Likelihood.

The steps for predicting a patient are diabetic or not based on single feature weight can be listed as

Collect patients’ data with and without diabetes diagnoses, along with their corresponding features.

To train our model, we would fit our S shape line to the data and obtain the parameters associated with it such as slope and intercept value together with hypothesis.

Parameters

Equation

1. If we have two patients’ data, each weighing 130 kg, and 50 Kg we can make some predictions. We can check out how these numbers affect the model by plugging them in.

Patient 1 Weight = 50 Kg

Patient 2 Weight = 100 Kg

We can see that the first patient (60 kg) has a very low probability of being diabetic, while the second one (120 kg) has a very high likelihood.

The outcome of this prediction can be of four kinds as explained in figure below

Chart, histogram

Description automatically generated

Figure 2.6: Diabetic Prediction

#### 2.1.1.3 Clustering

Clustering is an unsupervised machine learning algorithm used to group data points having similar characteristics. Clustering is of two types; first one is hard decision clustering. The clustering decision output will specify only one cluster. In soft decision clustering, the result provides the likelihood of data points to be in each set. A few examples for clustering problems are image segmentation, recommendation systems, analysis of social networks, customer trend analysis, etc. the available clustering algorithms are,

1. K means clustering
2. Agglomerative hierarchical clustering
3. Divisive hierarchical clustering
4. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
5. OPTICS (Ordering Points To Identify Clustering Structure)
6. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

**K Means Clustering**

Algorithms using unsupervised techniques make inferences about datasets without making use of known outcomes. The foundation of K-means clustering is discovery of underlying patterns by gathering similar data points. In order to achieve this objective, K-means seeks out a fixed number (k) of clusters. The central point of the cluster is the centroid. In K-means clustering K number of centroids will be chosen. The data points are allotted to each cluster by reducing the squares within each cluster. The algorithm starts with randomly generated centroids. Then, by reducing the sum of squares inside each cluster, every data point is assigned to a cluster and optimized according to the results of the iterative calculation. Cluster creation and optimization are halted when either the Centroids have stabilized, their values have not changed since the clustering succeeded. Or we have reached the maximum number of iterations.

Chart, scatter chart

Description automatically generated

Figure 2.7: K means Clustering

## 2.2 Machine Learning task

All enterprises are keen to collect the data and use Machine Learning algorithms to leverage data to have insights about their activities. Understanding the life cycle of the ML model will enable you to know where you stand in the process and manage resources more effectively. Proper care must be provided in each phase of the ML life cycle to operationalize ML algorithms. The following are steps of a typical ML life cycle

### 2.2.1 Data Ingestion

The process of crawling into various data sources and transferring them into the destination where storage and analysis are much more manageable. The target data destinations can be databases, data warehouses, data mart, or a document store

**Types of Data Ingestion**

Data ingestion can be classified into three based on when we are crawling the data.

1. **Real-time Data Ingestion**

Data is sourced, processed, and loaded as soon as it is created. Therefore, real-time data ingestion is highly sensitive to time. For example, the data ingestion from a power grid needs to be processed in real-time for anomalies.

1. **Batch Data Ingestion**

The most widely used data ingestion techniques are batch processing. The ingestion process regularly collects the data from the source and transfer it to the target system. Batch processing can be done based on any logical ordering. For example, triggered by a condition, or a scheduler can initiate the batch process.

1. **Lambda Architecture**

This method has the advantage of batch and real-time data ingestion techniques—batch ingestion for broad data views and real-time ingestion for time-sensitive information.

### 2.2.2 Hypothesis Generation

After data ingestion, the next step is hypothesis generation. It generates educated guesses of various features affecting the problem that needs to be solved using Machine Learning. Without proper hypothesis generation, the chances for machine learning projects' failure increase exponentially. Hypothesis generation helps to face the problem in a structured way and increases the domain knowledge of the data scientist.

Consider an example problem of city taxi trip duration prediction the hypothesis generation face will consider various factors that affect the duration of a taxi trip. the possible hypothesis generated can be listed as follows

1. More the trip distance the duration will increase
2. Drop points increase duration increases
3. As the speed rises, trip duration decreases
4. For small cars, the trip duration will be less
5. Pulling trips can increase the trip duration
6. More the age of the driver higher the trip duration
7. Less experienced drivers contribute higher trip duration
8. Drivers with medical conditions can contribute higher trip duration
9. Passengers with medical conditions caused the higher trip duration
10. Passengers with emergency cause lower trip duration
11. Weekends high traffic contributes higher trip duration
12. On festive days the trip duration will be increased due to high traffic
13. Early morning trips require less trip duration
14. Rainy weather will increase the trip duration

### 2.2.3 Model Evaluation

After we formulate, the hypothesis needs to be tested using a machine learning model. Before selecting a model, careful investigation and evaluation must be carried out to choose the suitable model. Before evaluating the model, the data need to be split into train tests and validation sets, and the training is done using the training data set. The new test data need to be used for evaluation.

For classification problem, the possible outcomes will be either of following four, consider a machine learning model for predicting whether a patient is diabetic or not.

1. **True Positive (TP)**

The model predicts an observation belongs to a class and it belongs to that class. In our example TP means our system predicts a patient is diabetic and he actually is.

1. **True Negative (TN)**

Model prediction observations that do not belong to a class but it does not belong to that class. In our example TN means that our system predicts a patient is not diabetic and he actually doesn’t have it.

1. **False Positive (FP)**

When you predict an observation belongs to a class, and it does not belong. FP in our example indicates the model predicts a person is diabetic and actually he is not.

1. **False Negative (FN)**

The model predicts that observation does not belong to a class, but it belongs to that class. The model predicts a person is not diabetic but actually the person has diabetes.

#### 2.2.3.1 Evaluation Metrics for Classification Models

1. **Confusion Matrix**

The confusion matrix depicts true positive, true negative, false positive, and false negative values graphically. hence it is easier to understand these values and makes easier to analyze the quality of the output.

A picture containing timeline

Description automatically generated

Figure 2.8: Confusion Matrix

Chart

Description automatically generated with medium confidence

Figure 2.9: Example: Confusion Matrix

In this example TP value is 750, TN is 1025, FP is 38, and FN is 27.

1. **Accuracy**

Accuracy is defined as the ratio of accurate prediction; it is suitable when classes are well balanced.

In the example shown above the accuracy value will be

1. **Precision**

The fraction of actually positive predictions among the total positive predictions is known as precision.

For our example the precision value will be

1. **Recall**

The fraction of the positive class prediction, which is truly the positive class

In the diabetes example

1. **F1 score**

F1 score is also known as the harmonic mean of precision and recall. F1 score is a measure of overall correctness that our model has achieved in a positive prediction environment.

In the diabetes example

1. **Area under the ROC (Receiver Operating Characteristics) curve**

ROC curve is a probability curve that plots the TPR versus the FPR at different thresholds. The area under the ROC curve is its performance measurement for the classification problem. In addition, it is the measure of distinguishability among the classes. Positive and negative classes are more easily distinguished when the AUC is high.

Chart

Description automatically generated

Figure 2.10: AUC=1

If AUC = 1, then the classifier is capable of correctly separating all Positive and Negative class points.

Chart, line chart

Description automatically generated

Figure 2.11: AUC=0.5

The classifier is unable to distinguished Positive versus Negative class points when AUC=0.5. In other words, either the classifier predicts a constant class or a random class for all the data points.

Diagram

Description automatically generated

Figure 2.12: AUC>=0.5

In the case of 0.5<AUC<1, a high probability exists that the classifier will be able to identify positive and negative class values. Thus, the number of TP and TN values will be very larger as compared to FP and FN values.

### 2.2.4 Evaluation Metrics for Regression Models

Regression is the type of machine learning that helps find the relationship between the dependent and independent variables. For analysis of regression models, the metrics required will be different from that of the classification model.

Residuals

Residuals correspond to the distance vertically between a point and its regression line. In simplest terms, it is the difference between a predicted value and the observed value.

Where, is the actual outcome and is the prediction. The greater the deviation between the real and predicted outcome, the less accurate the model is; the closer the values, the better the system's performance. Consider an example model for predicting the premium of the medical insurance based on the health features of an applicant. If the predicted premium value is 450$ and actual value is 500$. Then the residual value will be 50$.

1. **Mean Squared Error (MSE)**

It is the squared difference between the actual value and predicted value.

For the example insurance premium prediction problem, if the summation of squared error is 1000$ and 100 predictions are used, then MSE will be 0. The Mean Square Error is not robust to outliers, which is a major disadvantage. The squaring will increase the residual value of outlier significantly.

1. **Root Mean Squared Error (RMSE)**

It is the square root of the mean squared error. In RMSE, errors are measured in the same units as the target variables.

For instance, if our target and is the insurance premium in dollars, RMSE shows the error in dollars, while MSE gives the error in dollars squared, which is more difficult to interpret.

1. **Mean Absolute Error (MAE)**

It is defined as the absolute error between the actual and predicted values. In calculating the MAE, we don't care about the difference between actual and estimated values, whether estimated > actual or vice versa, but rather the absolute distance between them. While MSE penalizes more for bigger errors, MAE does not

If we have 100 predictions and summation of residuals of these 100 predictions is 250$ in our example regression problem. Then MAE will be .

1. **R – Squared (R2) / Coefficient of Determination**

**Chart, scatter chart

Description automatically generated**

Figure 2.13: Coeff. of determination

The figure shows the relation between two variables, insurance premium and Body Mass Index (BMI), in our example regression problem. The blue dots are the data points, orange line indicates the mean line which is the mean values of all the data points and the purple line indicates the regression line. The residuals for each of the lines are measured and the difference between unity and the ratio of residuals of regression line and squared sum of the residuals of the mean line. to the then the R – Squared is the measure of how the regression line is better than the mean line.

indicates the square of the () between the variables. The difference between and is that, gives more interpretable values than . For example, if we choose two variables such that the , and , we can not quantify how much better the first as compared to the second. If we choose ,

For

For

From value we can say the first one has double the correlation than second, which can be easily interpret from values.

### 2.2.5 Evaluation Metrics for Clustering Models

* **Silhouette Score:-** It is the measure of separateness between the clusters. Silhouette plot is a visualization of cluster separations. It is a value between -1 and +1 which indicates the how similar different points within the cluster and how different it is from neighboring clusters. For mean intra cluster distance (i) and mean nearest cluster distance (n), the silhouette score is,
* **Rand Index**:- This is the measure of similarity for every pair in our model to each pair in optimal or ground truth cluster. This measure has less practical application than Silhouette Score. The value is between 0 and 1, for a perfect fit the RI will be 1.
* **Adjusted Rand Index:-** It overcomes the unpredictability in the expected value of rand index between two random clusters. It is comparative similarity measurement where the randomness distribution is fixed.

* **Davies-Bouldin Index**:-Davies-Bouldin Index measures the similarity between clusters based on the ratio of distances within and between clusters. In contrast to most performance metrics, the lower the score the better the clustering performance. Clusters that are farther apart and less dispersed will lead to a higher score.

### 2.2.6 Testing and Validation

The data set for the machine learning pipeline be split into training data set validation data set and testing data set.

#### 2.2.6.1 Training Dataset

The training data set is the data set that we are using to train the model. The significant chunk of the data will be split as the training data set. If we are using the training data set for supervised learning then it will be having labels that we're going to predict. The training data set must have the data collected from diverse sources for perfect generalization of the model.

#### 2.2.6.2 Validation Datasets

The validation data set infuses new data during the training process to get some helpful information to optimize hyperparameters. The validation process is usually an optional process that is useful for fine-tuning the model.

#### 2.2.6.3 Test Datasets

Test data set is usually unlabeled data set to check the generalization capability of our model. it can give an idea about how well the model is fit to the training data set.

Testing Dataset

Validation dataset

Training Dataset

Figure 2.14: Dataset Split

The split ratio is the measure of the relative quantity of each of these datasets. For example, suppose the divided ratio is 0.7:0.1:0.2. In that case, 70 percent of the data will be split as training data set 10 percent as validation data set reminding 20% will be the testing data set. Usually, people first split their data into train and test data sets. Then they keep aside the test data set. Finally, X percentage of the training data set is chosen for training (100 – X) percentage of the training data set will be used for validation purposes.

### 2.2.7 Model Tuning

Model architecture is a choice you will have to make when creating a machine learning model. Sometimes, we do not know exactly what the best model architecture is for a given model, so we would like to be able to experiment with a range of possibilities. Hyperparameter tuning is the process of identifying the ideal architecture by tuning the parameters that define the model architecture. Hyperparameters are not model parameters, and they cannot be directly learned from the data. Hyper Parameters are some variables that affect the characteristics of training process, but these parameters are not trainable parameters itself. Unlike model parameters, hyperparameters define the structure of our model. The model parameters specify how the input data is transformed into the output. The learning rate, regression rate and batch size are few examples for hyper parameters. Careful tuning of hyperparameters is required for optimum performance. The fine-tuning of hyperparameters can maximize the model performance. The hyperparameters selection is a crucial step in the machine learning pipeline. there are many methods for hyperparameter selection are exist. In tuning hyperparameters, we look for the optimum values by looking at possible model architecture candidates, also known as "searching" the hyperparameter space for them.

#### 2.2.7.1 Hyperparameter Tuning

Generally, there are many terminologies associated with a machine learning model such as model parameters, model hyperparameters, etc. that are often confusing to understand. Therefore, we briefly discuss the difference between the terminologies.

1. **Model Parameters:-** It is a set of variables internal to the model. Few examples of the model parameters are shown in FIG. 1a. The values of the model parameters are determined by the training model using the historical data i.e., the training dataset and an optimization algorithm (say, gradient descent, adagrad, etc.). Therefore, the values of the model parameters are dependent on the training dataset. Further, the model parameters are saved along with the trained model. The saved model parameters are utilized in making the predictions for the new data during the inferencing. The model parameters decide the performance of the machine learning model.

Weights of a neural network - W0, W1, … Wn.

Support vectors.

Coefficients of a regression model.

Centroids of a clustering model.

Filter values of a CNN model.

Figure 2.15: Examples of model parameters

1. **Model Hyperparameters:-** The word “hyper” indicates a “top-level”, i.e., the hyper-parameters are at a higher level as compared to model parameters. The model hyperparameters are external to the machine learning model. Few examples of the model hyperparameters are shown in FIG. 1b. The values of the model hyperparameters cannot be determined from the training data and must be set using hyperparameter tuning or determined manually. The values of the model hyperparameters must be set before starting the training of the machine learning model. The model hyperparameters are in-turn used to determine values of the model parameters. Therefore, we regard the model hyperparameters as the control attributes for training a machine learning model. The model hyperparameters decides the efficiency of trained machine learning model. Further, the model hyperparameters are not used during inferencing.

Learning rate.

Number of clusters.

Number of hidden layers in a neural network.

Choice of activation function, penalty function, loss function, etc.

Value of regularization variable.

Choice of train test split.

Figure 2.16: Examples of model hyperparameters

#### 2.2.7.2 Use Case for hyperparameter tuning

Consider a scenario where a classification model such as logistic regression is to be trained. The logistic regression model comprises the co-efficients as the model parameters and a data scientist or a machine learning engineer needs to determine the value of the model hyperparameters say, the regularization parameter (“C”) and the maximum iterations (max\_iter). One way of determining the best set of values for the hyperparameters is to manually set the and try (i.e., trial and error approach) different combination of values for the model hyperparameters. For example, {C = 1, max\_iter = 500}, {C = 0.01, max\_iter = 500}, {C = 0.01, max\_iter = 1000} etc. The manual process can be laborious and time consuming. Therefore, hyperparameter tuning (discussed in the subsequent sections) is used to automate the process determining the best set of values for the model hyperparameters.

Tuning **,**It is a process of training multiple models with same algorithm and same training dataset, but for different combinations of the values associated with the model hyperparameters. Further, the trained multiple models are tested and validated to determine the performance of the models using a suitable metric such as accuracy, precision, F1-score etc. The model with the best set of performance is selected for inferencing.Before, we start with the process of hyperparameter tuning, the dataset needs to be split into training and test datasets. One approach is to split the dataset into two parts i.e., training dataset and testing dataset using TrainValidationSplit function in the Spark engine as shown in FIG. 2a. Generally, 70% of the data is used as training data and 30% is used as testing data.

Dataset

Testing Dataset

Training Dataset

Figure 2.17: Train Validation Split

Another approach is to split the dataset into two parts i.e., training dataset and testing dataset. Further, the training dataset is in-turn divided into “k” subsets as shown in FIG. 2b. For each set of hyperparameter values, the machine learning model is trained “k” times (also known as k-fold training). In each training step, “k-1” subsets are used for training and “1” subset is used for validation. This is achieved using the CrossValidator function in the spark engine. Further, the performance of the machine learning model is an average of the performance over the k-folds.

Dataset

Testing Dataset

Training Dataset

Testing Dataset

K = 1

Validate

Train

Train

Testing Dataset

Train

Validate

Train

K = 2

Validate

Testing Dataset

Train

Train

K = 3

Figure 2.18: Cross validation split

#### 2.2.7.3 Hyperparameter Search Space and Sampling

Before, we start with the tuning of the hyperparameters, the set of hyperparameters to be tuned should be identified corresponding to the machine learning model. Further, the search space and sampling technique for the hyperparameter values should be defined.

1. **Select the hyperparameters:-** A machine learning model may include many hyperparameters to be set. In order to determine the list of hyperparameters associated with a model, model\_name.explainParams() function from the spark can be used.
2. **Search Space:-** The search space includes a set of values associated with a hyperparameter variable that should be tried during the training of the multiple models. For example, max\_iter = [20, 30, 50] indicates the set of 3 values to be tried for a machine learning model during hyperparameter tuning and identify the best model based on the performance. The set of values to be tried depends on the type of hyperparameter. There are two main types of hyperparameters namely, Discrete and Continuous. As the name indicates, in case of the discrete hyperparameters, the model may be trained by selecting the hyperparameter values from a finite set of possible values. The discrete hyperparameters includes binary values (0/False, 1/True), integer values such as 5, 14, 180, etc. The discrete values for the hyperparameters may be defined as a list of values or can be drawn from a discrete distribution such as qnormal, quniform, qlognomal, qloguniform.

Examples of search space with discrete hyperparameters:

[10, 12, 18, 24, 36, 45]

quniform(a, b) = quniform(5, 10)

5 6 7 8 9 10

Figure 2.19: quniform

In case of continuous parameters, the model may be trained by selecting any value in the provided range or scale as the hyperparameter value. Generally, the continuous hyperparameters include floating point numbers such as 0.1, 0.001, 2.56, 10.87, etc. The continuous values of the hyperparameters may be drawn from the continuous distributions such as normal, uniform, loguniform, and lognormal.

Examples of search space with continuous hyperparameters:

normal(mean, standard deviation) : normal(5, 2).

σ = 2

µ = 5

Figure 2.20: normal

1. uniform(a, b) : uniform( 10, 50)

10

50

Figure 2.21: uniform

1. **Configure Sampling technique:-** The sampling technique defines one or more approaches used to select the values for the hyperparameters from the search space. There are 3 main approaches for sampling the values of the hyperparameters namely, Grid Sampling, Random Sampling, and Bayesian Sampling.

* **Grid Sampling:-** In this approach, all the possible combinations of the hyperparameters values in the search space are tried and finally, the best set of values is provided based on the performance metric.

For example, if HP1 = [2, 5], and HP2 = [60, 85, 90], then the 6 models trained for the two hyperparameters specified comprises {[2,60], [2,85], [2,90], [5,60], [5,85], [5,90]}.

* **Random Sampling:-** In this approach, some of the hyperparameter values are randomly selected from the specified values or range and models are trained. The best of values is provided based on the performance metric among the tried values.

For example, if HP1 = [2, 5, 8, 10], and HP2 = uniform(150, 156), then the models trained for the two hyperparameters specified may comprise {[8,155.6], [10,151.94], [8,154.82]}.

* **Bayesian Sampling:-** In this approach, the value of the hyperparameters to be tried is decided using the Bayesian optimization algorithm. After every iteration, the algorithm tries to provide a combination of hyperparameter values such that the performance of the model increases as compared with the previous combination of the hyperparameter values.

For example, if HP1 = [2, 5, 10], and HP2 = normal(1, 7), then at iteration ‘i’ if the values [2, 1.9] resulted in a performance metric value of ‘X’. For the iteration ‘i+1’ the algorithm provided the hyperparameter values (say [2, 1.1]) such that the model performance is better than ‘X’.

Each of the sampling techniques have their own advantages and disadvantages. The selection of the sampling technique is done based in the search space and the type of hyperparameters. For example, grid sampling is time consuming, random sampling works the best when the range of continuous hyperparameters are very huge, etc.

#### 2.2.7.4 Early Stopping Policy

The process of searching the best value of the hyperparameters is time consuming when the number of hyperparameters and/or the number of values in the search space is large. Therefore, we evaluate the trained models at specified intervals to check if the desired performance has been reached. If the performance metric has reached the desired value, then the process of hyperparameter can be terminated because the best set of hyperparameter values have been determined. This reduces the time for hyperparameter tuning since all the values of the search space have not been tried. There are three main techniques in early stopping policy, namely Bandit Policy, median stopping policy, truncation selection policy.

1. **Bandit Policy:-** In this policy a margin value for the performance metric used to evaluate the trained models is specified. Further, an evaluation interval is also provided, that indicates the periodic interval for checking. After the completion of the evaluation interval, the current performance metric is compared with the (best performance metric value – margin value). If the current performance is lesser than the (best performance metric value – margin value), then the hyperparameter tuning is stopped. Further, evaluating the policy can be delayed until a certain number of iterations have been completed by providing a delay value.

For example, consider a margin value of 0.2 and evaluation interval is 5. Let the best performance value obtained be ‘Y’. The evaluation policy, after every 5 iterations compares the current performance metrics (say ‘X’) with (Y - 0.2). If the value of ‘X’ is lesser than (Y – 0.2), then the hyperparameter tuning is stopped.

1. **Median stopping policy:-** Median is the middle value in a set of sorted values. In this approach, the hyperparameter tuning is stopped when the current value of the performance metric is worse than the median of the performance metrics obtained for the previous combinations of the hyperparameter values. Further, the delay value and evaluation interval can also be specified.

For example, let the values of the performance metric (say accuracy) after training the model for 5 hyperparameter values be [ 0.62, 0.68, 0.71, 0.75, 0.78]. At the 6th iteration if the performance metric value is less than the median value i.e., 0.71, then the hyperparameter tuning is stopped.

1. **Truncation selection policy:-** In this approach, a truncation percentage value (‘X’) is provided. At the end of the evaluation interval, the least performing ‘X’ combinations of the hyperparameter values are removed.

For example, if the value of ‘X’ = 20, then after the evaluation interval, two models with the least performance is removed. Therefore, when no more models can be removed, the hyperparameter tuning is stopped.

Hyperparameter tuning is an efficient method for optimizing the models and to determine the best model from a group of models. The hyperparameter tuning maximizes the performance of the model on the desired performance metric. Further, the combination of the search space with the sampling technique and early stopping technique reduces the time taken to determine the optimized model for a given training dataset.

### 2.2.8 Model Selection

* **Grid Selection**

Grid selection is also known as parameter sweeping, in which all the combinations of the hyperparameter set provided by the user will be tested for selection. For each possible combination of the hyperparameter values provided, a model is constructed, evaluated, and selected based on the results.

For our regression model if we have two hyperparameters such as

Learning\_Rate = [0.1, 0.01]

Batch\_Size = [2, 4]

would yield the following models.

LinearRegression(Learning\_Rate =0.1, Batch\_Size =2)

LinearRegression(Learning\_Rate =0.1, Batch\_Size =4)

LinearRegression(Learning\_Rate =0.01, Batch\_Size =2)

LinearRegression(Learning\_Rate =0.01, Batch\_Size =4)

λ

* **Random search**

Every possible combination will not be tested here, but the model randomly selects a certain number of hyperparameter pairs. We no longer provide a discrete set of values for each hyperparameter to explore with random search, but instead a statistical distribution with a random sample for each hyperparameter. This method is less resource and time consuming as compared to the grid selection

* **Bayesian optimization**

Bayesian optimization belongs to a class of sequential model-based optimization algorithms (SMBO). Where, we can use the information from one experiment to improve the next. Let the first model with hyper parameter λ having an evaluation metric . Next, we compute a posterior expectation of the hyperparameter space based on the previously evaluated hyperparameter values. Our next step is to select the optimal hyperparameter values based on the posterior expectation, and then repeat this process repeatedly until an optimum is reached.

### 2.2.9 Model fitting

Model fitting is a measure of how well the machine learning model generalizes the training data. A good model fitting enables the machine learning model to predict output for the unseen inputs correctly. Fitting is a process of making trainable parameter adjustments in the model to improve accuracy. The model tried to predict the outcome and the predicted data is compared with the label, the difference in these values is the error, which can be utilized to update system parameters. In new epoch again the process repeats with a smaller error. At the end of multiple epochs, the difference between actual and predicted output decreases below a threshold.

* **Over Fitting**

When the model learns too much about the training data, it fails to generalize. The situation in which the model was unable to generalize is known as overfitting. In this case, the model is useless for dealing with new data even though it can perform well with the training data.

* **Under Fitting**

The inverse of overfitting is underfitting, here also the model cannot predict unknown data. This problem arises due to a lack of proper training.

### 2.2.10 Model Generalization

When the model fits perfectly, then the model is said to be generalized. Reduced ability to react to new data. A well-generalized model can predict any unforeseen situation. Hence generalization is the basis for a successful machine learning model.

## 2.3 Sample Use Cases

The applications of machine learning are very vast, from automated cars to sophisticated modern medical diagnosis systems to use the virtues of machine learning algorithms. Let us consider a few practical machine learning use cases.

* **Medical Insurance Premium Prediction based on Client details**

Based on the Clients medical details and habits the insurance premium can be predicted using a machine learning model. The problem of this kind comes under regression problem. The linear regression model can be selected for training. The dataset needs to be split into testing and training dataset. The training dataset can be used for training. After training process the trained model can be utilized for predicting the premium values for the test dataset. The results of prediction can then be compared with the actual label can produce the required evaluation metrics.

* **Prediction of diabetes**

The machine learning model can predict diabetes from the patients' data such as glucose, blood pressure, BMI, diabetes pedigree, age, etc. It is a simple classification algorithm that can predict diabetes by training with the patient data. The primary concern while designing such a model is that

1. Selection of data set
2. Feature selection based on correlation
3. Data preparation
4. Training and evaluation of machine learning model
5. Saving the model
6. Making predictions using the model

* **Classification of Tumors**

Cancer is a genomic disease that makes uncontrolled cell growth known as tumors. However, cancer can be treated and cured if it is detected in the early stages. Machine learning models could help to detect cancer cells from normal cells if we trained the model well. there are many implementations of tumor detection by various organizations

1. **collaborative cancer cloud**

The Knight Cancer Institute at Oregon health and science university implemented this tumor detection algorithm

1. **Veye chest**

Amsterdam Based company called Aidance developed this machine learning system to help radiologists detect, quantify and report lung cancers

1. **Cancer LinQ**

American Society for clinical oncology (ASCO) create cancer LinQ which is big data analytics to gather all cancer data

# Chapter 3 Deep Neural Network

## 3.1 Overview: Neural Networks

We have seen the traditional machine learning methods and use cases in the previous chapters. Machine learning algorithms learns patterns from the data and makes decisions/ predictions. We have seen various types of supervised and unsupervised algorithms in regression and classification-based problems. Now, let us see a subset of machine learning known as Deep learning. In Deep learning, the learning takes place based on a type of networks which simulates the functions of human brain, called Neural Networks or Artificial Neural Networks. In traditional machine learning, the learning and decision making is limited to the capabilities of the algorithms used whereas in deep learning, the neural network can take better decisions, like the decision-making process of human brain. Neural network’s massively parallel distributed architecture which contributes to exceptional computing power and its ability to generalize the learning obtained, for inputs which are not provided during learning phase, makes it unique and accurate in real-world problem-solving scenarios. Traditional machine learning has simple structure whereas Neural Networks have layered structure to solve complex problems in logical fashion. Traditional algorithms require more human intervention whereas Neural Networks require very less human intervention. Deep learning is used when there are millions of datapoints available for analysis compared to the traditional machine learning techniques which can process thousands of data.

Neural Networks can be used in Classification, Clustering and Predicting the outcomes. It is used in segmentation of customers which allows businesses to target on their marketing techniques, thereby helping them improve the business in best possible way. A 30-layer Neural Network was used in improving Google’s search engine functionality by identifying complex searches involving shapes and colors, thereby reducing error rate from 23% to 8%. It has also been widely used in credit card fraud detection, developing personalized treatment plans in medical industry, Accurate Insurance provision based on customer needs, Self -driving cars, map applications etc. Near-human-level image classification​, speech recognition, text to speech conversion etc. are already achieved with the help of Artificial Neural Networks.

Let us now see the basics of ANN, Deep learning, and Deep Neural Networks in this chapter.

### 3.1.1 McCulloch Pith Model

The most fundamental unit of neural networks is called an *artificial neuron/perceptron*. McCulloch and Pitts created first artificial neural network by mimicking biological neuron functionality in 1943.

Diagram

Description automatically generated

Figure 3.1: Structure of a biological neuron

Source: www.wikipedia.org​

The neuron consists of Dendrites, soma, Axon and Synapse, which carry out the following functions.

* **Dendrite**: Receives signals from other neurons
* **Soma**: Processes the information received.
* **Axon**: Transmits the output of the processing.
* **Synapse**: Point of connection to other neurons.

So, basically a single neuron receives information through Dendrites, processes the information with the help of soma and transmits the output through Axons, which in turn can pass onto other neurons through connection links called synapses. In our brain, billions of such interconnected neurons process information to arrive at a decision on how to respond and react to many situations which happen in our day today life. Now let us see next, how this model is recreated artificially.

Diagram

Description automatically generated

Figure 3.2: McCulloh Pit Model-Artificial Neuron

The inputs are fed to g. It performs an aggregation (sum of binary inputs) and f takes an decision based on the aggregation. The inputs and output are Boolean. If the sum exceeds a threshold value, the output is 1 else 0. This model works on linearly separable data.

### 3.1.2 Perceptron Model

In 1957, Frank Rosenblatt modified the above artificial neuron model known as perceptron. This model can work on non-Boolean values as well. Perceptron model takes weights depending upon the inputs provided.

1

1

W0

∑

mm

2

W1

W2

Wm

Input

Weights

Net input

Function

Activation function

Output, yk

bk

*vk*

Figure 3.3: Structure of a Perceptron

As shown in figure 6.1, a node combines data input (1, ) with a set of coefficients or weights ) that either increase or decrease that input, thereby assigning importance to each input depending upon the task that the algorithm is learning. This product is added with bias , which is used to shift the activation function. Bias is like intercept added to the linear equation, which makes the ANN model fit best to the given data. The added result, , is passed onto the activation function Ꝕ(.), which converts the output of the combiner to the required voltage levels. The input features are assigned significance using adjustable weights.

From fig 6.1, the equation of linear combiner’s output can be written as eq [6.1] below.

eq (6.1)

Where are the input signals, are the synaptic weights of neuron k, is the output of linear combiner, which is the sum of products of the input signals and the corresponding weights.

The equation of output can be expressed as eq [6.2]

eq (6.2)

Where,

: The bias

= : The activation potential.

Ꝕ(.): The activation function.

yk :  The output signal of the neuron.

The use of bias **bk** has the effect of applying an affine transformation to the output **uk** of the linear combiner in the model of the neuron.

Perceptron’s play an important role in binary classification. They are also referred to as binary classifiers.

### 3.1.3 Activation Functions

Some of the activation functions are sigmoid, Relu, Softmax and tanh.

* **Sigmoid**

The sigmoid activation function is a logistic function which transforms the input values to a value between 0 and 1. The input values which are more than 1 is converted to 1 and the values which are smaller than 0 is converted to 0.

The sigmoid function can be expressed as (3.1)

The function can be depicted as shown in figure 6.2 below

Diagram

Description automatically generated with medium confidence

Figure 3.4: Sigmoid Function

* **Rectified linear activation unit (ReLU)**

It has the formula:

The function is represented in figure 3 below, with f(x) = y, as the output and x as the input. According to equation 4 and figure 3, the function outputs the input if it is positive and the function outputs ‘0’ if the input is a negative number. So the output will always be either ‘0’ or a positive number. This function is very commonly used in many neural network models as it is simple and easy to train and produces good outcome.

Chart, line chart, scatter chart

Description automatically generated

Figure 3.5: RELU activation function

* **Softmax**

Softmax function converts numbers into probabilities. It converts a vector of numbers into a vector of probabilities. These probabilities of each value will be in proportion with the scale of values in the vector.

Here, represents the input vector and stands for ith element in the vector space. represents standard exponential of input vector, K number of classes in the classification.

* **Tanh function**

This function can have values ranging from (-1,1) and fig.4 below depicts the function.

Diagram

Description automatically generated

Figure 3.6: Tanh function

**Example** : Implementation of AND gate using neural networks

Consider x1, x2 Ɛ {0,1} and have y = x1 AND x2, w1=30, w2=20 and w3=20. The neural network can be implemented as shown

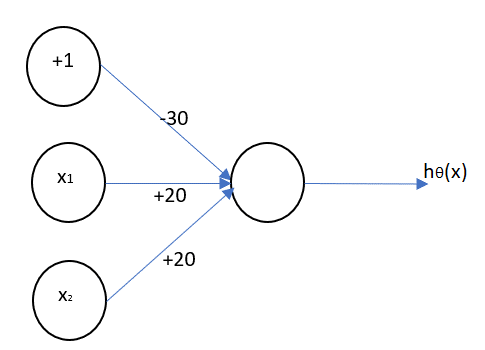


Figure 3.7: Neural Network: AND Gate

|  |  |  |
| --- | --- | --- |
| x1 | x2 | hθ (x) |
| 0 | 0 | f (-30) ≈ 0 |
| 0 | 1 | f (-10) ≈ 0 |
| 1 | 0 | f (-10) ≈ 0 |
| 1 | 1 | f (10) ≈ 1 |

which is like logical AND function. In similar way various gates can be implemented using neural networks. However, perceptron’s can classify, if the data is linearly separable. If the classes are non-linearly separable, for example XOR function cannot be solved by perceptron.

The primary data structure in ANN is layers. Layers are combinations of multiple neurons. Layers can be classified into three based on the position of each layer: input layer, an output layer, and hidden layers. The input layer will be the first layer of an ANN which will accept the user input. The output layer reduces the required output from the artificial neural network. Finally, the hidden layers produce the data transformations. There can be any number of hidden layers are possible, if the number of hidden layers is high such ANNs are known as Deep Neural Networks (DNNs).

Based on the number of layers, ANN is classified into single layer and multilayer Neural Network. The perceptron shown in figure depicts a single layer structure. Only linear decision surfaces can be predicted by perceptron’s. In order to classify non-linear decisions multilayer networks are used.

### 3.1.4 Multilayer Neural Networks

Neural Network with more than one layer is known as multilayer network.Fig.4.7 below, consists of input layer, hidden layer, and output layer. Output from each layer is connected to the inputs of next layer. Input layer receives the data that is considered for the algorithm. There can be multiple hidden layers also.

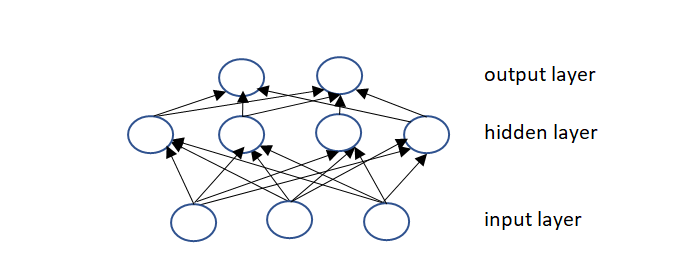


Figure 3.8: Multilayer ANN

Let us discuss the main algorithm used in learning and reducing errors in ANN next.

1. **Feed forward networks**

In case of feedforward networks, there is no feedback path present as shown in figure 4.8. The weights map the input to a set of guesses. The three main steps followed by the network are scoring input, calculating loss and trying an update to the model.

A picture containing sky

Description automatically generated

Figure 3.9: Feed forward network

Input \* weight = guess

The ground truth about the data is taken and is compared with the guesses

Ground truth - guess = error

The distinction between ground truth and networks guess is known as error. Depending upon the error, the network tries to adjust the weights to reduce the error.

Error \* contribution of weights to error = adjustment.

Feedback Neural networks on the other hand will have feedback paths present in it which propagates the output back to the input side.

Advantages:

* It can model complex and non-linear functions easily.
* Easy to maintain deep networks as well.

Disadvantages:

* Prone to overfitting.
* As there are a greater number of parameters, it may take more time to converge and more inference time and model size.

1. **Backpropagation Algorithm**

Backpropagation means “backward propagation of errors”. The backpropagation algorithm does the learning of ANN. It is a supervised learning algorithm. Here all the weights are initially assigned randomly; after each input, the air in and predicts the output. The difference between expected output and actual output is calculated as an error. This error is propagated in the backward direction to the previous layer. Then, the weights are updated based on this propagated error in such a way that it reduces the resulting error. This process is repeated until the error decreases below the threshold value. The backpropagation and the weight updating are the core of the training process; after training, the ANN model is ready to work with new inputs. The most common method used in minimizing the error is depicted below.

Diagram, schematic

Description automatically generated

Figure 3.10: Backpropagation algorithm

Following are the steps for backpropagation algorithm

(a). Inputs enter through the input layer.

(b). Weights are selected randomly and inputs are modeled using the weights.

(c). Output is calculated for each neuron passing through the hidden layer.

(d). Error is calculated which is the difference between actual output and desired output.

(e). Again propagate in the backward direction to adjust the weights to reduce error.

(f). The processes are repeated until desired output is obtained.

Advantages of backpropagation:

* Simple, fast and easy to program
* It does not require networks prior knowledge.
* Other than inputs, no other parameters need to be tunes.
* Features of the function need not be specified to be learned.

Disadvantages of backpropagation:

* The performance is dependent on the input data.
* In some applications, algorithm is sensitive to noisy data.

1. **Gradient Descent**

To optimize the cost function or the error of the model, gradient descent is used. Gradient descent (GD) is an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning(DL) to minimize a cost/loss function. The error in the model must be reduced quickly without much resource wastage. A ball rolling down a hill is like gradient descent. The ball will roll to the point that is lowest on the hill. Similarly, when the error is the least, model is optimized.

Chart, line chart

Description automatically generated

Figure 3.11: function indicating its minima

For gradient descend method to work and find the minima of a function, the function must be differentiable and convex. As per the name, Gradient represent the rate of inclination/ declination of a slope and Descent indicates the instance of descending to reach the minima of the function. Hence gradient descent indicates the step-by-step process of reducing the slope to reach the minimum point, which is used to minimize the error/loss function. When the gradient descend can’t reduce the error any further, it is said to be converged. As it is an iterative process towards reducing the error, the number of iterations required to minimize the error varies depending on the error function.

Different types of gradient descent algorithms are Batch Gradient Descent, Stochastic Gradient Descent and Mini Batch Gradient Descent.

* **Batch Gradient Descent:** In this method, the error is calculated for each sample within the training dataset, but the model gets updated only after the entire training samples have been evaluated.
* **Stochastic Gradient Descent:** In this method the error is calculated for each training sample and parameters are updated then and there itself. So, the model gets updated each time the error is evaluated.
* **Mini Batch Gradient Descent:** This method is a combination of the first two methods. Here, training samples are divided into small batches and the model gets updated after each batch is updated on the error.

1. **Cost Function**

Cost function is a single value which gives a measure of performance of the neural network with respect to the prediction. It is the difference between the output predicted and the actual output. This parameter is like a correctional function that helps to understand when the model is the most accurate. Consider a robot segregating boxes. Initially if it bumps to any other obstacle, it learns that it has not taken right decision. Next time when it comes across any obstacles other than box, it learns to take the right decision. Different variables are used to better fit the data. The outcome helps to optimize the variable which acts as a cost function to better fit the model. A cost function gives you an idea of how badly model is predicting.

**-**

In neural network each layer will have a cost function. So the minimum value out of all values are considered which is known as global minima. Each layers minima is known as local minima.

Chart, line chart

Description automatically generated

Figure 3.12: Gradient descent for neural networks

## 3.2 Introduction to DNN

Deep artificial neural networks or deep learning is a subset of machine learning. Deep learning is widely popular in applications like sound recognition, recommender systems, image recognition and natural language processing. In early 2016, DeepMind’s AlphaGo algorithm used deep learning to beat former world champion Lee Sedol at GO and champion Ke Jie in 2017.Deep neural networks (DNN) are Artificial Neural Networks (ANN) which has more than one hidden layer in it. More hidden layers make it possible to pass input data features through more mathematical operations making it computationally intensive to train. Simple features in input data recombine from one layer to the next forming complex features, thus hidden layers help to learn the features of the data in a form of feature hierarchy. Deep learning provides high accuracy and performs well on unstructured machine perception tasks as well. Because of the high complexity involved requires much better hardware’s like Graphical processing units and takes hours to train the model. Neural networks with one or two hidden layers are known as shallow networks and more than three are known as deep neural networks.

The DNN performs the required mathematical operations using deep learning techniques to convert inputs into the output, irrespective of the relationship be linear or not. It is also known as Universal approximator, as it can approximate any function f(a)=b for any input ‘a’ and any output ‘b’. Neural network finds the right way of transforming ‘a’ to ‘b’ for any function like, f(a) = 4b-0.9 or f(a) = 7b-0.6.



Figure 3.13: Deep Neural Networks

### 3.2.1 Training a deep neural network

* Multiple iterations known as *epochs* are carried out to train a deep neural network.
* Random initialization values are assigned to weights(w) and bias(b) values are assigned for the first epoch.
* After that the process is as follows.
  1. Features of data observations with known labels are generally grouped into batches. (also referred as mini-batches).
  2. The neurons with activation functions applied are sent to the next layer until output are generated.
  3. The error is calculated.
  4. The revised weights are calculated, and adjustments are done to the neuron weights.
  5. The next epoch repeats the forward pass with revised weights and bias values to improvise the model accuracy.

The most popular networks in deep neural networks are convolutional neural networks. Now let us have a brief overview of CNN’s.

## 3.3 CNN

Convolutional Neural Network (CNN) Artificial Intelligence are quite popular in applications like image recognition, medical image analysis. Major advances in computer vision (CV) are powered by CNNs which has applications for robotics, drones, security, medical diagnosis and for visually impaired treatments. Convolutional Neural Networks (CNNs) are Deep Learning methodologies that take an input image and give importance (weights and biases) to distinct entities in the image, allowing them to be distinguished.

The architecture of a CNN is based on the organization of the Visual Cortex and is akin to the neuron connection pattern in the human brain. Individual neurons only respond to stimuli in the Receptive Field, which is a tiny subset of the visual field. An aggregate of similar fields covers the full visual area via overlapping.

A CNN can successfully capture the Spatial and Temporal correlations in a picture by applying filters. The design delivers a better fit to the picture dataset due to the reduced number of parameters and reusability of weights. As a result, a network may be trained to better understand the image's sophistication.

To understand this algorithm let’s consider an example of a three-color input image (red, green, and blue) (RGB)

A picture containing diagram

Description automatically generated

Figure 3.14: Image represented as a matrix

Images are ingested and processed as tensors by CNN’s. Matrices of numbers with additional dimensions are tensors. A scalar for example ‘6’ is one dimensional whereas a vector is two-dimensional. A three-dimensional cube is a stack of matrices. The three dimensions are width, height, and depth. Arrays nested within arrays and infinitely nested arrays accounts for dimensions greater than what we can visualize. Order is tensor’s dimensionality; therefore, fourth order tensor would have four dimensions. Red-Green-Blue (RGB) encoding produces three-layer deep image. Each layer is called a channel. Grayscale, RGB, HSV, CMYK, and more image formats are available in CNN.

CNN reduces the images to a simpler format keeping all the key characteristics for producing an accurate forecast. This is important when creating an architecture that can learn features while still being scalable to large datasets.

### 3.3.1 CONVOLUTION LAYER-KERNEL

Convolving is the process of rolling over. Mathematically convolution is measuring how two functions overlap in an integral way as they pass over. The integral is the area under the curve.

This layer carries out a dot product between two matrices. One matrix is a kernel, and the other is the data portion overlapping with the kernel. A kernel helps to extract features from the image. The kernel matrix will usually be smaller than the image matrix. Hence the kernel will slide over the image to produce the output of the convolution operation. The sliding size of the kernel is called a stride. These features are represented as feature maps. The Kernel K is the element in the first half of a Convolutional Layer that implements the convolution operation. The image, I, is a 5x5x1 RGB, whereas this kernel, K, is a 3x3x1 RGB in yellow.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 3.15: Convolution operation with kernel

Consider Kernel/Filter, given by

If Stride Length= 1, the Kernel shifts 9 times, each time performing a matrix multiplication operation between K and the image part P through which kernel progress. The filter is a concatenation of kernels with each kernel assigned to each channel. For a CNN layer with kernel dimensions h\*w and input channels k, the filter dimensions are k\*h\*w. In 2D convolutions, the kernel will be a 2D matrix and filter will be a 3D matrix.

When the stride value is set, the filter flows to the right until it occupies the entire width. This process is continued again from image’s beginning (left) until the entire image has been explored. From a given input the Convolution Operation collects high-level information like edges. There can be many numbers of Convolutional Layers in a CNN. The first layer is often responsible of acquiring Low-Level data such as gradient orientation, edges, colour, and so on. When more layers are added, the framework adapts to the Elevated features, resulting in a network that understands all the images in the dataset. The output is passed through an activation function like RELU.

### 3.3.2 Pooling /Sub-Sampling

Its point is to limit the number of parameters and calculations in the network by steadily shrinking the spatial dimension of the representation. The pooling layer replaces the values of the locations with the statistical value derived from the neighboring values—the number of adjacent values dependents on the stride size of the pooling layer. There are two types of pooling: maximum pooling average pooling and minimum pooling. Max Pooling extracts the maximum value from the portion of the image covered by the Kernel. For example, if we use max-pooling with a 2x2 filter, the filter will choose four neighboring values and the maximum value out of the four values.

A picture containing text, clock

Description automatically generated

Figure 3.16. Pooling

Average Pooling returns the average of all the values from the image's Kernel section as shown in figure. Similarly minimum pooling extracts the minimum values from the image. After pooling, flattening is done to convert the matrix to a 1-D array as shown in figure.

Table

Description automatically generated with medium confidence

Figure 3.17: Flattening

### 3.3.3 FULLY CONNECTED LAYERS

The fully connected layer is the layer that connects all the neurons with all other neurons in the next layer. By adding a fully connected layer, image data will be converted to a multilayer perceptron scheme. The fully connected layer (FC) has weights and biases used to connect the neurons between two layers. Flattened input is fed to FC layer. FC layers are frequently found right at the end of CNN architecture and classification process begins to take place.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

A picture containing indoor, white, mammal, dog

Description automatically generated

Cat

Dog

Fish

Horse

Mouse

Monkey

Feature Maps Feature Maps

Pooling

Convolution

+ RELU

Output

Prediction

Fully Connected Layer

Flatten

Figure 3.18: Convolution Neural Networks

After Fully connected layer, the next parameter is the activation function. This helps to learn and approximate any continuous and complex relationship between variables of the network. The right information is forwarded to the next layer. Various activation functions are ReLU, Softmax, sigmoid and tanH functions. Sigmoid and SoftMax are preferred for binary classification and for multi-class classification usually SoftMax is used.

Applications:

* Speech recognition
* Image classification
* Facial Recognition
* Climate predictions
* Natural Language Processing

## 3.4 Recursive Neural Networks

The language translator uses recursive neural networks. When sequential data like video, speech must be processed, RNN is a better option. In RNN, there is a path from the output towards the input layer of the system, whereby the input gets influenced partly by the feedback signal. The category of artificial neural network which uses feedback path in the architecture is known as Feedback Neural Network, which falls into a category known as Recurrent Neural Network. A Recurrent Neural Network will have minimum one feedback loop present in it. It can be single layer or multilayer.

Diagram, schematic

Description automatically generated

Figure 3.19: Recursive Neural Networks

Feed-forward networks doesn’t have memory as they just move in the forward direction. But in recursive neural networks, as there is feedback it takes a decision considering current input and the learning from the previous inputs. Imagine that we want to process word “network”. Applying the “network” as input to neural network and it tries to process character by character. As feed forward networks doesn’t have memory it’s not possible for it to predict characters. As recurrent neural networks have memory, they can remember characters. The output generated is copied back as input into the network.

There are various types of applications where RNN can be used. They are

1. One to one: If input is fed to a hidden layer and produces a single output like image classification scores that implies one to one network.
2. One to Many: But machine learning should provide the flexibility to handle various kinds of data, for example Image captioning where for an input image, captions should be generated. In this case one to many can be used.



Figure 3.20: Types of RNN

1. Many to one: Input could be a sequence of text and generate sentiments i.e sentiment classification.
2. Many to many: Input and output could be variable in length like Machine translation. In machine translation the input could a sentence in English of variable length and must translate it to an output of variable length in another language. Also, the length of input and output can be variable as well. Also, to process video sequences RNN’s can be used.

## 3.5 LSTM (Long Short-Term Memory)

Generally, RNN can remember things for a short duration of time. Because of the presence of feedback, RNN can remember things or taking decisions based on past inputs. But if we want the information to be remembered for a long period of time and get it reproduced, then RNN cannot be used. If we need to use the old context of an information to predict the next outcome, then RNN fails to do it with good accuracy. So, if we consider a scenario for long term dependencies, a slightly modified version of RNN, known as LSTM is used. In LSTM, a memory structure is introduced, which is controlled by a gate that decides the things to remember, things to forget and the things to output.  LSTM is very productive for time series forecasting based problems. They are used in applications like speech recognition, language modeling, Image captioning, Sentiment analysis etc.

LSTM handles selective filtering of data based on gates. It has 3 types of gates known as

* Forget gate
* Input gate and
* Output gate.

Forget gate decides what information we are going to throw away from the cell. Input gate decides which information is allowed to pass inside the cell layer and output gate decides what we are going to output.

Based on the state of the cell, a filtered version of it will be passed on as output.

A picture containing text, sky, map

Description automatically generated

Figure 3.21: LSTM and GRU

There are different variants of LSTM based on the way in which these gates are used. One variant used coupled forget and input gates. In this model it works together and the forget gate decides to forget only if a new value is given as input and input gate takes in new data. only if it forgets old data.

In another version, it replaces the forget and input gates with an update gate, which combines both the types. This variant is known as Gated recurrent units or GRU. GRU has only two gates known as reset gate and update gate. GRU’s are computationally faster than normal LSTM with three gates. Adding more neurons to LSTM layers help in solving complex problems.

## 3.6 Transfer Learning

In transfer learning, the knowledge acquired by the machine learning model training can be utilized for a different but related problem. For example, a simple classifier model trained to detect sunglasses can be used for detecting face masks. Training a machine learning model means we're updating the weights and biases according to the data set. In transfer learning, the updated weights are reused for a new task to improve the generalization. It's interesting to know-how transfer learning is working. Consider an example of a computer vision model. Usually, the earlier layers of a computer vision model detect edges of the image, layers in the middle they picked shapes, the final layers will identify problem-specific features. For transfer learning out of these layers, the early and middle layers will be utilized. And the task is specific output layers will be replaced according to the new problem that the model needs to address. Benefits of transfer learning are less training time and better performance. The various models are VGG16, RESNET 50, Inception, Xception.

### 3.6.1 VGG16

VGG16 is a 16-layer multilayer network of 13 convolutional layers, each with a 3x3 filter size and fully linked layers. It was developed by Visual Graphics group. Karen Simonyan and Andrew Zisserman proposed this network. It’s a predefined model and is trained on ImageNet dataset.VGG-16 is around 533MB in size due to its depth and density of entirely connected nodes. Color images need to be resized to 224x224 to input to this model.



Figure 3.22: VGG16

### 3.6.2 ResNet 50

ResNet is an abbreviation for residual networks. It uses residual learning instead of learning from features. Residual is the subtraction of feature learned from input of that layer. The future layer is formed by combining the preceding layers. This network makes use of shortcut connections.

Diagram

Description automatically generated

Figure 3.23: ResNet 50 source: <http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006>



Figure 3.24: Skip Connections in ResNet

Skipping one or more layers as shown in figure results in skip connections.Skip connections do identity mapping and their outputs are added to stacked layers.Resnets optimizes easily compared to other networks where training error increases as depth increases.Also greatly increased depth results in increased accuracy compared to traditional networks.

## 3.7 Model Serving: Real-Time deployment

After the model training, the model serving is a critical process to use or leverage by the end-users. Together with the ML model, data scientists need to think about many other aspects

### 3.7.1 Data Wrangling

Data wrangling have a significant role during the real-time deployment of the model. Since the data scientist may have done many transformations to the data for training like one-hot encoding, the same transformation needs to be provided to the real-time data from various sources

1. Storage

The storage space the model requires is significant in the model serving phase. However, since storage constraint deployment like containerizations and usage in mobile devices, we need to reduce the storage size of the model without much impact on the performance.

1. API

API design is also an essential aspect in model serving since the user window to the model is API. Therefore, according to the service structure of the ML model, the API structure needs to be carefully chosen.

1. Scheduling

Scheduling of training and retraining is the primary function of a scheduler. It can be scheduled periodically, or it can be triggered when the performance goes beyond a threshold

1. Logging and Monitoring

Proper logging of data and monitoring of health and conditions of deployment is crucial for making the model deployment fault-tolerant.

1. Security

ML models are usually dealing with critical data, and ML models take many crucial decisions. Hence the security breach of the deployment can cause serious harm.

1. Visualizations and User Interfaces

The visualizations and UIs to be designed according to the target users

## 3.8 Use Cases

### 3.8.1 Medical Image Analysis

The convolutional neural network is used popularly to analyze medical images as its accuracy is better compared to other techniques. A convolutional neural network can identify edges, shapes, and even more sophisticated features over an image. Steps involved in medical image analysis using the convolutional neural network are following:

* Selection of data set, medical image data set is available from various sources. Also, many open data sets are available.
* Preparation of data, Image augmentation process, can be used to expand the size of the data set.
* Selection of CNN model. the filter sizes, pooling layer, and stride lengths must be chosen according to the requirements.
* Model iteration
* Evaluation of the model.

Convolutional neural networks and transfer learning techniques are used for detection of cancer, pneumonia, covid 19 using chest X ray and CT scan as input images.

### 3.8.2 CNNs for Skin Cancer Diagnosis

Skin cancer or Melanoma is a deadly disease. The diagnosis of skin cancer is carried out on the dermatoscopic images taken with a high-resolution magnifying camera. Therefore, as a good tool for image detection, CNN can be utilized for skin cancer diagnosis.

### 3.8.3 Visual Data Processing for Tumor Detection

The tumor images can be classified based on the visual features of the cancer images. The images and its masks can be visualized, and classification model can be trained using transfer learning modules such as ResNet-50. ResNet-50 is a convolutional neural network that is 50 layers deep. We can load a pretrained version of the network, trained on millions of images from the ImageNet database. This network has the capability to classify images into various categories, which can be utilized in identifying if the image has tumor or not. In addition, computer vision algorithms can also be used for the detection of tumor cells.

# Chapter 4 Distributed Deep Neural Network

## 4.1 Introduction

Big data and Deep learning are two trending names in the field of Data Science. The two names are inter-related and inter-dependent due to capability of solving challenging problems related to computer vision, speech and audio, natural language processing, etc. By 2021, the volume of data was predicted to reach 30 zettabytes (ZB) to 35ZB (zetta = 1021 bytes) according to a survey conducted by International Data Corporation (IDC).

Generally, processing of Bigdata includes performing operations on high volume of data (for example, in terabytes 1012 bytes), streaming huge amount of across internet within a timespan of milliseconds to seconds. Further, handling the variety of data such as texts, multimedia, structured and unstructured data. In addition, processing of Bigdata comprises handling the veracity in data such as data inconsistency, ambiguities, incompleteness, etc.

The Deep learning models trained using the Bigdata have shown an exponential increase in the performance as compared to training with small datasets. Further, the models tend mitigate the risk of overfitting with the increase in the data used to train the model. However, training the deep learning models on Bigdata is time consuming. Even with the usage of GPUs the deep learning models take days or months for completing the training. In addition, the higher dimensional data, the number of features increase the training time. Therefore, there is a need to reduce the training duration of the deep learning models.

In view of the above stated problems, the parallel and/or distributed computation techniques have evolved to reduce the training time, that are discussed in the subsequent sections.

## 4.2 Parallel and Distributed Techniques for Deep learning

The parallel and distributed techniques are used to share the computations involved in the deep learning models for reducing the training time. Here we briefly introduce some methods used to achieve faster training times.

* **Local Training:** In this technique, the data and model is stored on a single machine with multiple cores. This technique can be regarded as a parallel technique. The multiple cores are used to train multiple mini batches of the model in parallel. The data used for training is present in a shared memory accessible by all the cores. Further, the GPUs present in machine can be shared for computationally intensive tasks such as matrix multiplication.
* **Distributed Training:** In this technique the data and the model are stored or distributed across multiple machines.

There are two main approaches in the distributed training i) Data Parallelism and ii) Model Parallelism.

* **Data Parallelism:** In this approach the data used to train the deep learning models are distributed and stored across multiple machines.
* **Model Parallelism:** In this approach the model is split and stored across multiple machines.

Regarding to the architecture and the training process using distributed learning are discussed in the further sections.

## 4.3 Architectures for distributed deep learning

### 4.3.1 Model Parallelism

As shown in FIG.1, model parallelism is achieved using a cluster of machines. The cluster of machines may be connected via a communication network. Each machine can be treated as a sever comprising multiple CPUs and GPUs. The deep learning model to be trained is split and distributed across multiple machines. For example, consider a deep learning model with 4 layers as shown in FIG.1.

In one instance, each layer of the deep learning model is stored and trained in one machine as shown in FIG.1. Generally, model parallelism is preferred when the deep learning model is too huge to fit in a single machine. The model is segmented such that each portion of the model can run concurrently. Further, each portion of the model operates on the same data in different machines.

Figure 4.1: Model Parallelism

### 4.3.2 Data Parallelism

A cluster of machines or servers is used to achieve data parallelism. The cluster of machines may be connected via a communication network. Each machine can be treated as a sever comprising multiple CPUs and GPUs. The deep learning model to be trained is replicated in each of the servers or machines. The training data set is divided to smaller subsets. The number of subsets is equal to the number of machines in the cluster. Further, each machine or server trains the copy of the deep learning model stored in the memory with the subset of the training data. Later, the trained parameters of the deep learning model are aggregated. Two types of architecture shown and described below

1. Decentralized architecture

as shown in figure below, a decentralized architecture where each node trains the deep learning model with a mini batch from the subsets and shares the trained parameters with the other machines or servers for aggregation.

DS

DS

DS

DS

Figure 4.2: Decentralized architecture

1. Centralized architecture

as shown in figure below, a centralized architecture where the master or the parameter server receives the updated or trained weights from each of the machine or the server and aggregates the weights. Further, the aggregated weights are distributed to each of the machine or server.

DS

DS

DS

DS

Figure 4.3: Centralized architecture

In addition to model parallelism and data parallelism, a hybrid distributed architecture that is a combination of model and data parallelism is also realizable. For example, using model parallelism the deep learning model may be split and distributed across GPUs, and data may be distributed across machines or servers.

## 4.4 Training in distributed deep learning

During the training of the deep learning models, minimizing the loss function (L) is generally performed using the iterative approaches based on the concept of stochastic gradient descent. The parameter update rules used in a mini batch of training is also based in the computation of the gradient. However, the gradient descent approach is computationally intensive and the time-consuming task in the process of training the deep learning models. Therefore, distributed deep learning approach aims to distribute the computationally intensive task such as the gradient computation among the cluster of machines or severs. This results in the reduction of training times.

## 4.5 Use cases

* 1. Consider a CNN model as shown in FIG.4a. The model parallelism corresponding to FIG. 4a is achieved as shown in FIG. 4b. All the 3 machines execute a mini batch concurrently and update the trained parameters. Further, only the dependent parameters are shared among the machines.

Machine -2

Machine -3

Machine -1

Data

Layer - 1

Layer – 2a

Layer – 2b

Prediction - 1

Prediction - 2

- 1

Data

Layer - 1

Layer – 2a

Layer – 2b

Prediction - 1

Prediction - 2

Figure 4.4: figure 4a Figure 4.5: figure 4b

* 1. Consider a deep learning model such as CNN or RNN, to be trained with 106 images. Using data parallelism with 3 machines, we can divide the 106 images into 3 subsets based on the class weightage as shown in FIG. 5. Further the deep learning model is replicated in all the 3 machines with random initializations. Each of the deep learning model is trained with the corresponding subsets concurrently for each mini batch as shown in FIG. 5. The trained parameters are combined and distributed to the 3 machines for training the subsequent mini batch.

Machine -1

Machine -2

Machine -3

DS

DS

DS

Figure 4.6: figure 5

## 4.6 Aggregation of parameters in Distributed Deep learning

With distributed approaches, the common challenges faced includes synchronizing the model parameters between the machines or servers and the combining the results (i.e., trained parameters) obtained from the machines in the cluster. The aggregation problem is addressed using two approaches namely, synchronous update and asynchronous update.

* **Synchronous Update:** This aggregation technique is suitable for centralized data parallelism architecture and model parallelism architecture. In this technique, Referring to FIG. 6, random initialization of the deep learning model parameters is performed in the parameter server. Further, the parameter server distributes a copy of the parameters and the deep learning model to each of the machine in the cluster. The training dataset is divided into subsets and distributed to each of the machine in the cluster.

The machines in the cluster, train on the subset of the data (mini batch) and share the trained parameters to the parameter server. The parameter server waits until it has received all the trained parameters from all the machines. Further, the parameter server determines the average of the trained parameters obtained from all the machines in the cluster. Finally, a copy of the averaged parameters is distributed to each of the machines for further training the mini batch.

Pseudo Code for synchronous update at the Parameter server:

1. Distribute a copy of global parameters (Wi) to the machines.
2. Wait till all the trained parameters (Δw1, Δw2, … Δwn) are received from the machines.
3. Compute the average of the trained parameters (1/n \* Σ Δwi).
4. Update the global parameters Wi as Wi+1 = Wi + α \* (1/n \* Σ Δwi).
5. Distribute a copy of Wi+1 to all the machines for the next iteration.

Parameter server - Wi+1 = Wi + α \* (1/n \* Σ Δwi)

Server - 1

Server - 2

Server - n

Wi

Wi

Wi

Δw1

Δw2

Δwn

Figure 4.7**:** Synchronous Update

* **Asynchronous Update:** This aggregation technique is suitable for de-centralized data parallelism architecture and model parallelism architecture. However, it can be used with the centralized data parallelism architecture as well. As compared to synchronous update, there is no need of parameter server and in case of different computation capabilities of the machines or severs there is no need to wait for the trained parameters from all the machines. Thereby, increasing the training process as compared to synchronous update.

In this technique, referring to FIG. 7, the global set of parameters are stored in a shared memory. Each machine obtains a copy of the parameters from the shared memory and updates the parameters by training on a mini batch of data. Further, the trained parameters are added or aggregated to the global parameters asynchronously by each of the machines.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Global Parameters stored in a shared memory | | | | | | | | | |
| W100 | W101 | W102 | W103 | W104 | W105 | … | … | … | W100+n |

Server -1. Obtains W100 performs mini batch update and adds to W104. W105 = W104 + α Δw100

Server -2. Obtains W100 performs mini batch update and adds to W101. W101 = W100 + α Δw100

Server -3. Obtains W102 performs mini batch update and adds to W103. W104 = W103 + α Δw102

Figure 4.8:Asynchronous Update

## 4.7 Evaluation of distributed deep learning models

The classical evaluation metrics used to evaluate the deep learning models are unsuitable for the distributed deep learning models because of the new aspects such as data distribution, communication overhead between the machines, aggregation techniques, etc.

Some of the suggested evaluation criteria for distributed deep learning are given below:

* Training Time vs Test Error
* Training Time vs Training Error
* Dataset size vs Training Time
* Dataset size vs Test Error

In addition, to the above said criteria, Jaccard-index, or Jaccard similarity coefficient can be used. The Jaccard similarity coefficient is defined as the cardinality of the intersection divided by the cardinality of the union of the sets A and B. For example, the sets A and B may represent features determined by the model 1 and features determined by the model 2 respectively.

Further, a plot of Feature size vs Dataset size vs F-score may be used to analyze the evolution of the F-score with respect to the features and dataset size. Similarly, a plot Feature size vs Sample size vs Jaccard-index and/or a plot of Feature size vs Sample size vs Training time may be used to evaluate the distributed deep learning model.

For example, consider the evaluation plot of test error vs training time corresponding to two machines (say machine -1 (red color) and machine – 2 (blue color)) as shown in FIG. 8. Based on the plot it can inferred that error of the model trained in the machine – 2 has reached required minimum test error at an earlier time as compared to the model 1.

## 4.8 Inferencing distributed deep learning models

The models trained using Data parallelism can be combined during training as detailed in the aggregation section of the document. Alternatively, the models trained can be combined post-training during inferencing with the help of ensemble learning approach.

Training Time

Test error

Min Test error

Figure 4.9:Training Time vs Test Error

Where the predicts of all the models are combined to determine the overall prediction for the given inputs. In such situations, the distributed learning is completely a parallel process, and no communication is required during the training.

In another instance, the size of the deep learning model trained using the model parallelism and/or data parallelism is reduced using knowledge distillation techniques. For example, a single model can be trained to mimic the output of the original deep learning model or ensemble of models. The single model is used for inferencing.

In general, the trained models are used for inferencing by deploying the models as a web service. The REST API may be used for communication between the web service and the client applications.

## 4.9 Advantages of Distributed Deep learning

1. Distributed learning is scalable.
2. Overcomes the problems related to centralized storage with respect to data and model.
3. Enables large-scale learning with complex algorithms and memory limitations.
4. Train several classifiers from distributed data sets, thereby achieving higher accuracy.
5. Integration compensates the bias of the classifiers.
6. Derive independent classifiers from different partitions of the data.

## 4.10 Applications of Distributed Deep learning

* Used in identifying the best model architecture using the Sequential Model-Based Optimization (SMBO), Reinforcement Learning (RL), and Evolutionary Algorithms (EA).
* Used in searching the best hyper parameters for the model.