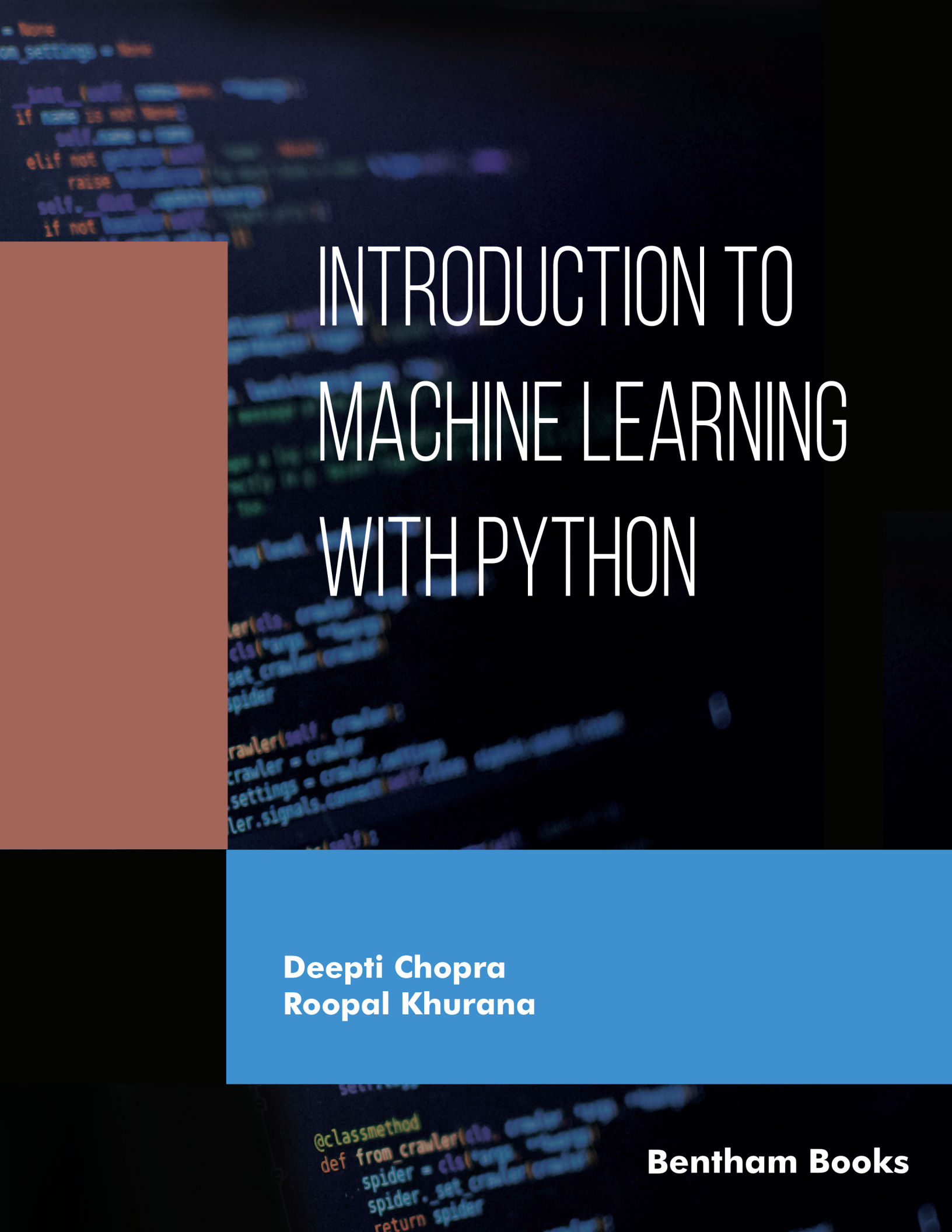

INTRODUCTION TO MACHINE LEARNING WITH PYTHON

Deepti Chopra
Roopal Khurana

Bentham Books

Introduction to Machine Learning with Python

Authored By

Deepti Chopra
Jagan Institute of Management Studies,
Sector 5, Rohini, Delhi-110085,
India

&

Roopal Khurana
Railtel Corporation of India Ltd,
IT Park, Shastri Park,
Delhi-110053,
India

Introduction to Machine Learning with Python

Authors: Deepti Chopra and Roopal Khurana

ISBN (Online): 978-981-5124-42-2

ISBN (Print): 978-981-5124-43-9

ISBN (Paperback): 978-981-5124-44-6

First published in 2023.

need for a court order if at any point you breach any terms of this License Agreement. In no event will any delay or failure by Bentham Science Publishers in enforcing your compliance with this License Agreement constitute a waiver of any of its rights.

3. You acknowledge that you have read this License Agreement, and agree to be bound by its terms and conditions. To the extent that any other terms and conditions presented on any website of Bentham Science Publishers conflict with, or are inconsistent with, the terms and conditions set out in this License Agreement, you acknowledge that the terms and conditions set out in this License Agreement shall prevail.

Bentham Science Publishers Pte. Ltd.
80 Robinson Road #02-00
Singapore 068898
Singapore
Email: subscriptions@benthamscience.net

CONTENTS

FOREWORD

I take the opportunity to congratulate the authors, Dr. Deepti Chopra and Mr. Roopal Khurana who have written this book titled, **"Introduction to Machine Learning With Python"**.

The advancement in technology in the past decade has been due to the introduction of Machine Learning. Today, machine learning has escalated Artificial Intelligence Revolution, be it in Fraud Detection and Prevention, Self-driving cars, Recommendation Systems, Facial Recognition technology, *etc.*

Machine Learning is one of the approaches of Artificial Intelligence in which Machines become capable of drawing intelligent decisions like humans by learning from their past experiences. In classical methods of Artificial Intelligence, step-by-step instructions are provided to the machines to solve a problem. Machine learning combines classical methods of Artificial Intelligence with the knowledge of the past to gain human-like intelligence.

The authors of this book have given explanations on Machine Learning with Python from the basics to the advanced level so as to assist beginners in building a strong foundation and developing practical understanding.

Beginners with zero or little knowledge about Machine Learning can gain insight into this subject from this book. This book explains Machine Learning concepts using real-life examples implemented in Python.

After learning from this book, one will be able to apply concepts of Machine Learning to real-life problems.

I am sure readers will benefit from this book and gain a lot in the field of machine learning.

Happy Reading!!

Best regards,

Rajesh Pokhriyal | Scientist 'D'
Indian Computer Emergency Response Team (CERT-In)
Ministry of Electronics & IT
Electronics Niketan 6 CGO Complex Lodhi Road
New Delhi 110003

PREFACE

Machine learning has become part and parcel of day-to-day private/non-profit/business and government operations because of its ability to grasp automatically through past experiences without being explicitly programmed. Today, machine learning has conquered the entire industry due to its numerous applications ranging from digital marketing to space research. Today, it governs the industry in terms of building high-tech products, ranking web searches, building speech recognition systems, recommendation systems, *etc.* However, we have not yet developed fully operational machines that give judgments on their own like humans but it is not far away to reach that level. From this book, we intend to re-discover the core concepts of Machine learning paradigms along with numerous architectures and algorithms used in different paradigms. The book elaborates on various topics related to the implementation side using Python with real-life examples. The book can kickstart your career in the field of Machine Learning. It also provides the basic knowledge of Python which is a prerequisite of this course. We can say that this book is meant for neophyte users who wish to get acquainted with the implementation of machine learning using Python. The reader will be able to read well-explained examples and exercises and it will be an ideal choice for Machine Learning enthusiasts. The book presents detailed practice exercises for offering a comprehensive introduction to machine learning techniques along with the basics of Python. The book leverages algorithms of machine learning in a unique way of describing real-life applications. Though not mandatory, some experience with subject knowledge will fasten the learning process.

CONSENT FOR PUBLICATION

Not applicable.

CONFLICT OF INTEREST

The author declares no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENT

Declared none.

Deepti Chopra
Jagan Institute of Management Studies
Sector 5, Rohini, Delhi-110085
India

&
Roopal Khurana
Railtel Corporation of India Ltd
IT Park, Shastri Park
Delhi-110053
India

CHAPTER 1

Introduction to Python

Abstract: Python is considered one of the most simple and efficient programming languages. Its object-oriented programming approach and elegant syntax make it a powerful programming language. Python is an interpreted language. Its dynamic typing and high level data structures make it an ideal language for application development in various areas and on multiple platforms. Today, Python is widely used in the areas of machine learning and data science. The following chapter discusses Python, the utility of Python in machine learning and data science, ecosystem of Python in machine learning and various libraries in Python required for machine learning.

Keywords: Data science, Jupyter, Machine learning, Matplotlib, Numpy, Python, Scikit learn, SciPy.

INTRODUCTION

Python was developed by Guido van Rossum in 1990s. The name of the language 'Python' was taken from "Monty Python's Flying Circus", which was one of the favorite TV shows of Guido van Rossum. Python has a simple syntax that was designed as a language that could be used easily by beginners yet proven to be one of the most powerful languages for advanced developers. Python is an object-oriented programming language that can be used on various platforms. The syntax used in Python is very simple as compared to other programming languages [1]. Today, Python is considered a very popular programming language among students, researchers, developers, *etc.* Python is extensively used by tech giants such as Netflix, Facebook, Google, *etc.* Python offers numerous applications [2], [3]. These include the following:

Web Development

Nowadays, Python is used widely in web development. Some of the frameworks for web development in Python are: Django, Pyramid, Flask, *etc.* These frameworks are known to incorporate characteristics such as scalability, flexibility, security, *etc.*

Deepti Chopra & Roopal Khurana

Game Development

PySoy and PyGame are two python libraries that are used for game development.

Artificial Intelligence and Machine Learning

There are a large number of open-source libraries which can be used while developing AI/ML applications.

Desktop GUI

Desktop GUI offers many toolkits and frameworks using which we can build desktop applications. PyQt, PyGtk, PyGUI are some of the GUI frameworks.

Today, Python is used extensively for doing research especially in the areas of bioinformatics, mathematics, biology, *etc.* It is a part of Computer Science curriculum for many universities.

It is not just companies that seek through python. Python is used in various fields such as Artificial Intelligence, Astronomy, Internet of Things and Social Science.

In this chapter, we will discuss Python, set up Python environment and the importance of using Python in Data Science. We will also discuss tools and libraries used in Python Programming.

SETTING UP PYTHON ENVIRONMENT

Python is available on different platforms such as Windows, Linux and Mac OS X. We can open Window terminal and type “python” ; this will return the version of python if it is already installed.

Current documentation, source code, news and updated version of Python are available at: https://www.python.org/

We may download documentation of python in different formats such as PDF, HTML and PostScript format from https://www.python.org/doc/.

For installing Python, we need to download the binary code according to our platform. If binary code for our platform is not available, then we need to compile the code on c compiler manually.

Steps involved in installing Python on Unix/Linux include the following:

Check if python is already installed on machine by going to terminal using Ctrl+Alt+T. For Python2, type python —version and For Python3.x, type

python3.x —version. In case, Python is already installed, then the version of Python installed is returned.

If Python is not installed then follow the following steps:

- Open the URL, https://www.python.org/downloads/.
- Download and extract files from zipped code available for Linux/Unix.
- Execute ./configure script
- Make, make install

The above steps install python libraries at /usr/local/lib/pythonYY. Here 'YY' represents the version of Python installed.

Steps Involved In Installing Python On Windows Include The Following:

- Open the URL, https://www.python.org/downloads/.
- Click on the link python-PQR.msi file and download it. Here, 'PQR' refers to the version of python we wish to install.
- Run the file and this installs python.

Steps involved in installing Python on Macintosh include the following

- Open the URL, https://www.python.org/downloads/.
- MacPython is used for older version of Mac; for Mac which are released before 2003.

Setting Up Path

The executable files and programs may be present in different directory locations. Path consists of a list of directories that comprise executable files that may be searched by the Operating System. Unix is case-sensitive and Windows is not case-sensitive. So, path is 'PATH' in Unix and 'path' in Windows.

Setting Up Path In The Unix/linux

Add python directory to the path using following ways:

- In csh shell, type set env PATH "$PATH:/usr/local/bin/python"
- In bash shell, type export PATH="$PATH:/usr/local/bin/python"
- In ksh shell, type PATH="$PATH:/usr/local/bin/python"

We can invoke python using different ways. One way to invoke python is by typing "python" at the shell command prompt. We may also type "help",

"credits","copyrights" and "license" to get more information about python. We can also open IDLE of Python from START. Python prompt is represented by three greater than sign (>>>). Python commands are written after '>>>' and return key is hit after each command in order to execute it. The 'print' command in python is used to print a statement. The print command prints the statement and adds a new line after statement.

We can terminate the python session on shell command prompt by typing ctrl-Z in Windows and ctrl-D on Unix.

The file extension of python file is .py. The first line in a python program is #!/usr/local/bin/python. Python consists of a similar structure like other programming languages. Python program may comprise of if/else/elif, while/for, try/except *etc.*

WHY PYTHON FOR DATA SCIENCE?

Python is a high-level, interpreted and open source language that is based on object-oriented programming concepts. Python is a very popular language these days. Python offers different libraries that help in implementing different data science applications [4]. Data scientists use python for implementing different applications and projects related to Data Science [5]. Python has the ability to build projects related to statistics, and mathematics and also deal with the scientific function. Python comprises rich set of library that may be imported to build data science related application. The reason why Python is considered a widely used language for building research-based projects and scientific applications is its simple syntax and ease in use .

Python provides deep learning frameworks that are upgraded day by day as well as scientific packages. It is also used extensively in the field of Machine Learning, Natural language Processing, *etc.*

Following are the characteristics of Python:

- It has a simple syntax and is easy to use.
- It has a huge library package as well as community support.
- It is simple to test python codes, detect and correct errors.
- Modules written in other languages like C/C++ may be appended along with python language.
- Python may be used extensively in research and data science based applications.
- Python provides flexibility to the developers as the code may be run on any platform like Windows, Unix, Mac OS, Linux *etc.*

- It is freely available. There is no cost involved in downloading python and using it in different applications.
- It is scalable. It is used in many industries, across many applications and is applicable in multiple use cases.
- There are various python packages available such as NumPy, Pandas and SciPy. Scikit-learn is also one of the python packages that may be used in Machine Learning applications. Matplotlib is a python package that may be used for data visualisation ; for generating graphs and plots.
- Python has ever growing community in which members are volunteering and adding a new libraries to the existing ones.

According to the recent report given by Bureau of Labor Statistics, there is a lot of competition in the job market today. In order to seek for a stable job; one must make a career in Data Science. Many a times for a same job; multiple qualified candidates are competing among themselves. In order to stand out and get noticed by the recruiters; a certification in Python with Data Science will be added on and will add weightage to the resume as well.

ECOSYSTEM FOR PYTHON MACHINE LEARNING

Python is considered as a high level scripting language. If one wants to make a career in machine learning; then it is necessary to learn python as well due to the following reasons:

1. Python comprises many libraries.

2. Simple and easy to understand.

3. Python coders are in demand in Fortune companies.

4. Easy to learn and comprehend.

Python comprises many libraries that help in performing Machine learning tasks. Some of these libraries are: Numpy, Pandas, Theano, Tensorflow, Keras, Matplotlib, Scikit Learn, NLTK, *etc.*

Deeplearning is a multilayer neural network that may be generated using Tensorflow; which is a framework given by Google. The numerical computation may be performed by a Python library called Theano. Keras is also an api written in python for carrying out neural network based tasks. Matplotlib is used for data visualisation purpose. Scikit-Learn is a machine learning based library that can be used for implementing various machine learning algorithms on a given dataset. NLTK is a Natural Language Toolkit that can be used to perform various Natural Language Processing tasks on a given source text to obtain the desired target text.

ESSENTIAL TOOLS AND LIBRARIES

Python comprises various tools and libraries useful for carrying out various tasks in the field of Machine Learning, Deep Learning, Data Analytics, Data Science *etc.* This section will discuss some of the important tools and libraries in Python:

Jupyter Notebook

Jupyter Notebook is also referred to as ipython notebook. It provides an interactive environment for building Data Science related applications. The following are some of the characteristics of python notebook:

- Jupyter notebook performs analysis in a step by step manner by arranging the components in an orderly manner.
- Data Scientists are able to do documentation in a well organised manner.
- Results may be saved and may be shared with our peers as well.

Jupyter notebook may be installed separately or if Anaconda is installed, then no separate installation is needed [3]. We may type the following on Anaconda prompt:

C:\>jupyter notebook

After pressing enter, jupyter notebook will appear. This is shown in Fig. (**1**) .

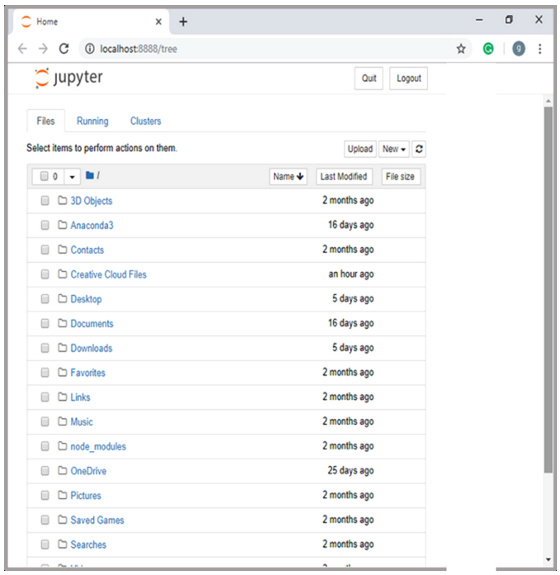

Fig. (1). Jupyter Notebook.

There is a New tab option. Select Python3 from drop down and new Jupyter notebook window will open where we can write our python code. This is shown in Fig. (**2**) below.

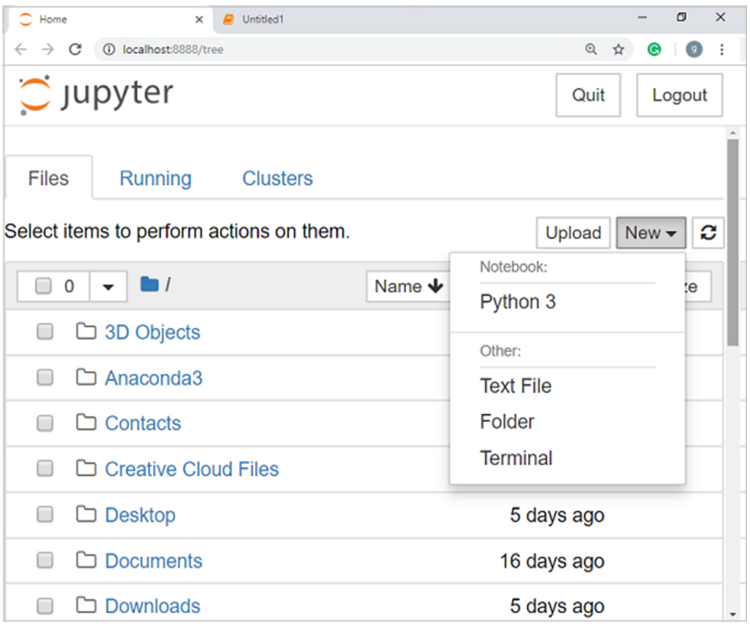

Fig. (2). Jupyter Notebook used for writing Python code.

If anaconda is not installed, then Jupyter notebook may be installed using pip as follows:

Pip Install Jupiter

There are 3 types of cells in Jupyter Notebook namely- Raw cells, Code cells and Markdown cells. The raw cells are those in which the text written is not altered or modified. The outcome of the computation process in the form of text, images, html tags *etc.* is stored in Markdown cells. Code cells are used for writing the codes. The code that is written is sent to the Jupyter kernel for processing.

NumPy

It is also referred to as Numerical Python. It comprises multidimensional array objects. Following operations may be performed using NumPy:

- Operations on Array
- Operations related to linear algebra
- Fourier transform

NumPy works along with Matplotlib and Scipy to obtain results; this is similar to Matlab.

If we have Anaconda installed, then we need not to install NumPy again. Instead, we can use NumPy in our python script and call it using import.

Import NumPy as np

If we have not installed Anaconda, then NumPy can be installed using pip as follows:

Pip install NumPy

Pandas

Pandas is one of the python libraries that help in data wrangling, manipulation and analysis. Pandas library was introduced by Wes McKinney in 2008. Following data processing tasks may be performed using Pandas:

- Model
- Load
- Manipulate
- Prepare
- Analyse

Pandas consists of data structures such as: Series, Data frame and Panel. Series is used to represent one dimensional array having homogeneous data. Frames are usually used to represent two dimensional data structure having heterogenous data. The panel is used to represent 3 dimensional data structure having heterogeneous data. It is difficult to represent panel graphically. It may be represented as a container of frame.

If Anaconda is already installed; then we need not to install pandas separately instead we can import pandas package in the python script as follows:

Import pandas as pd

If Anaconda is not installed, then we can install Pandas using pip as follows:

pip install pandas

After installing pandas, we can use pandas in python script by importing pandas package as follows:

```
Import pandas as pd
```

Scikit-learn

Scikit-learn is a very useful library in python that performs the task related to Data Science and Machine Learning. Scikit-learn comprises features of SciPy, NumPy and Matplotlib. It is an open source library. Codes written in Scikit-learn may be reused under BSD license.

If Anaconda is already installed, there is no need to install Scikit-learn; instead we can import Scikit-learn directly in python script.

If Anaconda is not installed, then we can install Scikit-learn using the following command:

```
Pip install -U scikit-learn
```

We can import dataset of breast cancer patients as follows:

```
From sklearn.datasets import load_breast_cancer
```

SciPy

SciPy is a python library that is useful in writing codes pertaining to Data Science as well as Scientific computing. SciPy comprises modules pertaining to linear algebra, interpolation, optimization, FFT, integration, Statmodel, signal and image processing.

Consider the following code in python that imports scipy and finds the cubic root of numbers:

```
Import sciPy

From sciPy import cart

Import NumPy

cuberoot=NumPy.cbrt([125,216])

print(cuberoot)
```

```
[5. 6.]
```

Matplotlib

Matplotlib is also one of the helpful python libraries used for performing Data Visualization. For a given organization, visualising and analysing the data are very important. Matplotlib provides an effective way of visualising data. Matplotlib helps in creating charts such as pie chart, bar graph, histograms, line chart, flow chart, *etc.*

Mglearn

Mglearn is a python based repository that comprises codes related to book titled "Introduction to Machine Learning with Python" written by Andreas Mueller and Sarah Guido. Mglearn package may be installed on the system using the command pip install mglearn.

PYTHON CODES

1. The following code is an example of using NumPy in python:

```
import numpy as np

a = np.arange(27).reshape(3, 9)

a

array([[ 0, 1, 2, 3, 4, 5, 6, 7, 8],

[ 9, 10, 11, 12, 13, 14, 15, 16, 17 ] ,

[18, 19, 20, 21, 22, 23, 24, 25, 26]])

a.shape

(3, 9)
```

2. The following is an example of creating a series using Pandas

```
import pandas as pd

import numpy as np t=np.array(['January', 'February', 'March','April','May', 'June','July', 'August','September', 'October','November','December'])

p=pd.Series(t)

print(p)
```

0 January

1 February

2 March

3 April

4 May

5 June

6 July

7 August

8 September

9 October

10 November

11 December

3. The following is the example of creating Data Frame using Pandas:

```
import pandas as pd

t = [['January',1],['February',2],['March',3],['April',4],['May',5],['June',6],['July',7],['August',8],['September',9],['October',10],['November',11],['December',12]]

dataframe = pd.DataFrame(t,columns=['Month','MonthNo'])

print(dataframe)
```

Month MonthNo

0 January 1

1 February 2

2 March 3

3 April 4

4 May 5

5 June 6

6 July 7

7 August 8

8 September 9

9 October 10

10 November 11

11 December 12

4. Consider the following python code that is used to draw a plot using matplotlib:

```
import matplotlib.pyplot as plt
x = [ 1,5,7]
y=[10,40,60]
plt.plot(x,y)
[<matplotlib.lines.Line2D object at 0x0A06AE70>]
plt.xlabel('xaxis')
Text(0.5, 0, 'xaxis')
plt.ylabel('yaxis')
Text(0, 0.5, 'yaxis')
plt.title('Value of y corresponding to x')
Text(0.5, 1.0, 'Value of y corresponding to x')
plt.show()
```

The output of the above code is shown in Fig. (**3**) .

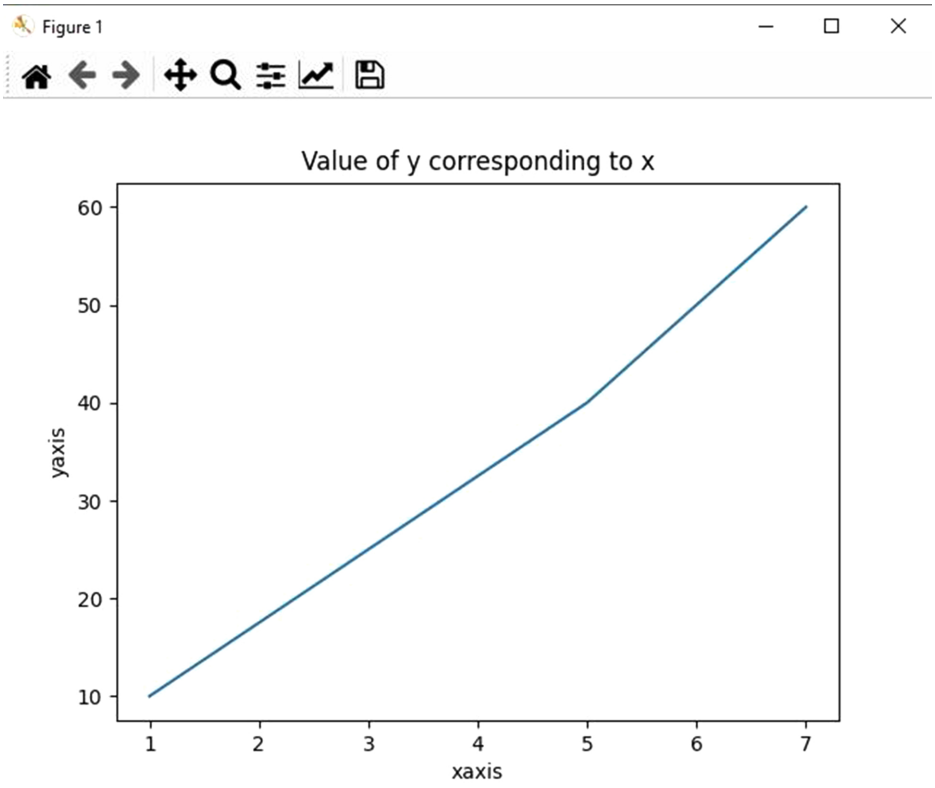

Fig. (3). Plot using matplotlib.

CONCLUSION

Python is a conducive programming language in the areas of Data Science, Machine Learning, Scientific Computing, *etc.* In this chapter, we have discussed Python, its characteristics, installation and what makes it the most suitable language in the area of Data Science. Further, this chapter also discusses the various libraries such as NumPy, Pandas, Matplotlib, SciPy, Scikit-learn and mglearn and tools such as Jupyter notebook which is used in Python Programming. In the next chapter, we will discuss the basics of Machine Learning, the steps involved in the Machine Learning process, In-Sample and Out-of-sample Error and Applications of Machine Learning.

EXERCISES

1. Explain in detail the following libraries in python:

a) Matplotlib

b) mglearn

c) Scikit-learn

d) NumPy

e) Pandas

f) SciPy

2. Explain why Python is considered as most preferable language in Data Science.

3. What features of Python make it best suitable for building machine learning based applications?

REFERENCES

[1] G. Van Rossum, *An Introduction to Python.,* F.L. Drake, Ed., Network Theory Ltd.: Bristol, 2003.

[2] A. Rawat, "A Review on Python Programming", *International Journal of Research in Engineering, Science and Management,* vol. 3, no. 12, pp. 8-11, 2020.

[3] A. Kadiyala, and A. Kumar, "Applications of Python to evaluate environmental data science problems", *Environ. Prog. Sustain. Energy,* vol. 36, no. 6, pp. 1580-1586, 2017. [http://dx.doi.org/10.1002/ep.12786]

[4] K. Soumya, G. Ramanathan, and G. Clinton, "An Assessment on Classification in Python Using Data Science", *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA),* 2021pp. 551-555

[5] J.S.S. Lowndes, B.D. Best, C. Scarborough, J.C. Afflerbach, M.R. Frazier, C.C. O'Hara, N. Jiang, and B.S. Halpern, "Our path to better science in less time using open data science tools", *Nat. Ecol. Evol.,* vol. 1, no. 6, p. 0160, 2017. [http://dx.doi.org/10.1038/s41559-017-0160] [PMID: 28812630]

CHAPTER 2

Introduction To Machine Learning

Abstract: Machine Learning is referred to as the subset of Artificial Intelligence. It involves a machine being learned without programming it explicitly. In Machine Learning, machines try to improve their performance with the help of past experiences or by using training examples. The following chapter discusses Machine Learning, Selection of training set, Selection of target function, Selection of Function Approximation Algorithm, Perspectives and issues involved while building a machine learning system,In-sample and Out-of-sample error and Applications of Machine Learning.

Keywords: Data science, Jupyter, Machine learning, Matplotlib, Numpy, Python, Scikit learn, SciPy.

INTRODUCTION

Machine Learning (ML) is a subset of Artificial Intelligence. It is a field in computational intelligence that involves the analysis and interpretation of structures and patterns present in the data and the machine is able to do learning, decision making and reasoning with these patterns and be able to draw intelligent solutions without human intervention. Using ML, a user can input huge amount of data, perform training, give test data and get the automatic result. If there are any errors identified in the result, then machine learning algorithm can make changes, correct errors and use this information to do better decision making in future.

Machine learning comprises 3 components:

- The computational algorithm, expert in doing Machine Learning task.
- Variables and features based on which decisions may be taken.
- Actual or human expected output to determine the accuracy of the system.

Initially, machine learning model is provided with training data for which the result is known. Machine Learning algorithm is executed and adjustments are accomplished till the result of ML model is exactly similar to the actual model. Once, the results of ML model is as desired, then the ML model is said to be

trained and testing data is fed as input to this trained ML model to get human like intelligent results.

Machine Learning basically comprises pattern matching as well as data exploration in order to obtain automatic result with least human intervention.

For any business or corporate world, data is a lifeline. Data driven decisions made by any organisation decide its fate in terms of organisation's growth, progress, sale, *etc.* Machine learning driven decisions if applied correctly, can escalate organisation's growth and enable it to excel compared to its competitive counterparts.

Machine Learning applications are developed in various areas such as computer vision, natural language processing, pattern recognition and image processing. Machine Learning applications are being built in various industries and sectors such as healthcare, automotive, finance, innovation, travel, utilities, energy, hospitality, life science, feedstock *etc.*

Machine learning applications are advancing with time and they have become indispensable in organisations for taking quick and intelligent decisions with no or least human intervention. Machine learning systems are used in: Recommendation systems in e-commerce system, social networking sites or websites as chatbot, displaying advertisement on the basis of user-liked content, in healthcare, medical diagnosis, self-driving cars, *etc.*

DESIGN A LEARNING SYSTEM

Machine Learning is a process in which a machine gets trained from the training data fed into it and the machine is able to improve its performance from past experiences and can give human-like intelligent output [1]. When training data is fed to a Machine Learning model, the Machine learning algorithm will develop a mathematical model. Using this mathematical model, when test data is sent to the machine learning model, it provides efficient results. For example, in designing a driverless car, training of the car is done on how to drive in different road conditions, what should be the speed and how to stop at a signal or when there is an obstacle. Also, the more the training is performed on Machine learning systems, the more accurate and desirable the results are obtained.

Steps involved in designing a learning system include the following:

- Selection of Training Set
- Selection of Target Function
- Selection of Function Approximation Algorithm.

We will discuss all the above-mentioned steps in detail in sub-sections.

Selection Of Training Set

The first step in designing a learning system is to design a training set. This training set will be used for training the system. The success of a machine learning model in terms of its accuracy depends on the quality and quantity of training set that is sent as an input to this model. Apart from training data, a machine learning model which is an artificial intelligence system must also learn from its experiences and use this knowledge to improve the quality of output in the future. For example, in a chess game, if a particular move leads to the losing situation, then a machine learning system must learn from this experience and suggest an alternative move that would lead to the winning situation. When a machine learning system has many training examples as well as feedback or result of each game; then a machine learning system will be able to train itself better and be able to depict good performance in the consecutive games. Thus, the performance of a machine learning system will escalate only when we input numerous examples considering all situations or cases and experiences on a machine learning system.

Selection Of Target Function

The next step in designing a machine learning system is to select a target function. The selection of a target function means that using a machine learning algorithm, a machine learning system is well equipped to take the next course of action to be performed. For example, in a game of chess, a machine learning system would be able to take decision to make the next intelligent move that would lead to a winning situation out of numerous available moves.

Selection Of A Function Approximation Algorithm

Choosing a training data will not decide the optimised moves taken by the machine learning systems. There needs to be the presence of numerous examples while training any system, of the outcome of this training whether success or failure is fed to the machine learning system as feedback. The success rate of the move helps the machine learning algorithm in deciding next optimised move to be taken and improve its performance with the least human intervention. For example. Deep Blue is the first supercomputer made by IBM that is based on Machine Learning concept and it won the chess game played against world chess champion Garry Kasparov in 1997.

PERSPECTIVE AND ISSUES IN MACHINE LEARNING

Designing an effective machine learning system is a complex task involving a lot of effort. The following steps must be taken in order to fine tune a machine learning system:

1. Developing a data pipeline as early as possible.

2. Determining high variance or high bias and taking immediate steps accor dingly.

3. Analysing manually missed records or patterns.

An efficient machine learning system must also undergo several tests and should learn from experiences so as to effectively improve its performance.

To develop a data pipeline, following steps must be followed:

• Elimination of Outliers or noisy data.

• Dealing with the null values can be done either by removing that complete row in which some of the null values are present or replacing the null value field by some average value or default value.

• Normalising the numerical values.

• Encoding the categorical attributes.

• Splitting the data into 3 parts: 70% training set, 15% testing data and 15% cross validation data.

• Fit and perform predictions using a suitable Machine Learning model.

Bayes Error means the error rate for a particular program is unreachable or optimal. High variance is a situation when Bayes error and train error are closer to each other but cross validation error is worst as compared to both. In High bias situation, cross validation error is closer to the train error but bayes error is much better than both.

In case of High bias, following measures must be taken:

• In case of logistic and linear regression, add polynomial features.
• Elevate the number of hidden layers or number of units present in a layer in a Neural Network.

• Increase in the number of iterations in a gradient descent.

In case of High variance, following measures should be taken:

• In a Neural Network, insert regularisation (L1 norm).
• Increase training data.

For designing a Machine Learning system, we need to design the interface, algorithm, hardware and infrastructure of data in such a manner that specific requirements such as scalability, adaptability, reliability and maintainability are met by the system. These essential requirements required for a Machine Learning system are discussed below:

a. Scalability- If there is an increase in the complexity of a Machine Learning system in terms of traffic or data volume, then a Machine Learning system must be able to handle such scaling growth efficiently. For example on a certain prime day if some feature in a particular e-commerce app stops working, then it may lead to heavy revenue loss in terms of lack of orders and may also raise question on the credibility of the app.
b. Reliability- Machine Learning systems should perform consistently in all conditions. We can check the reliability of a given Machine learning system by providing a tricky data for testing corresponding to simple training data. In case of failure, the machine learning system will not give an error instead it will product an absurd value as an output.
c. Adaptability- Machine Learning systems should be able to add new data or make changes to the existing data as the need arises without deteriorating its performance.
d. Maintainability- As the time passes, there may be a need to update the data distribution in a ML system; this may affect the performance of a ML system. We should be able to retrain or refresh the Machine learning system and make necessary upgradations without affecting its throughput.

Issues In Machine Learning

There are numerous issues faced by ML expert while developing a ML based application.

Quality of Data

The output of a Machine Learning system depends on the quality of data that is sent as an input to the ML system. The presence of noisy data may cause ML algorithm to give incorrect result. We need to perform preprocessing on data

which involves eliminating outliers and dealing with the missing values. This needs to be done prior to sending the data for the training.

Improve the Quality of Training

ML system will produce a quality output only when training is done properly. The following things help in enhancing the quality of training:

- Increase the training time
- Include complexity in the model by adding more features to the data.

Overfitting the Training Data

Overfitting means when training is performed with enormous data that affects its performance negatively. We can solve overfitting of the data by:

- Deep analysis of data
- Reduce the number of features
- Eliminate outliers
- Perform data augmentation

Underfitting is the opposite of Overfitting. Underfitting takes place when the ML model is too simple. We need to add certain new features to this model.

Machine Learning Involves A Complex Process

Machine Learning is a complex process comprising data analysis, data preprocessing, training the data, performing complex mathematical computations, *etc.*

Insufficient training data

Machine Learning model needs to be trained with an enormous amount of data in order to obtain accurate outcomes.

Feasibility of Learning An Unknown Target Function

Machine Learning involves a complex process. For building a machine learning model, the following steps are involved:

Collection of Data

It involves finding the data on the basis of the machine learning project that we desire to make. Data may be gathered from various sources such as files, sensors, databases, *etc.*

Pre-processing of Data

The data collected from different sources for building the machine learning model cannot be directly used for analysis purpose, as it may contain a large amount of noisy data, unorganized text, missing values, large values, or irrelevant text. All such unwanted data may be eliminated in order to obtain clean data. While developing a machine learning model, we must follow the 80/20 rule. According to this rule, we must spend 80% of time in pre- processing of the data and 20% of time in analysis. Data may be classified into the following categories:

a. **Numerical**: For *e.g.*, age, salary, *etc.*
b. **Categorical**: For *e.g.*, nationality, gender, *etc.*
c. **Ordinal**: For *e.g.*, high, medium, low, *etc.*

Data pre-processing may be performed in the following ways:

Dealing with Null Values

We can solve the problem of null values by either deleting the rows and columns that comprise null values or by using imputation, which is a process of substituting the missing values with some substituted values.

Standardization

It is a process that involves the manipulation of values so that the mean of all the values is 0 and the standard deviation is 1.

Dealing with Categorical Variables

Categorical variables are those which are discrete and not continuous.

Feature Scaling

It is a technique in which we make the values of all the features same by scaling down the features that are insignificant and have a large range of values.

Splitting the Data

In machine learning, we usually split the data in 70:30, meaning 70% of the data is used for training and 30% of the data is used for testing.

Finding The Model That Will Be Best For The Data

Find the machine learning model which is best suited for our problem.

Training and Testing Of The Developed Model Evaluation

Data preparation refers to data pre-processing and prediction refers to the testing phase of the developed machine learning model. The feasibility of machine learning may also be expressed using Hoeffding's Inequality. According to Hoeffding's Inequality, for a given sample size M and ϵ as tolerance, the probability of the difference of in-sample estimation (μ) and the expected outcome (v) is smaller than a constant quantity.

It is defined as follows:

$$P(|v-\mu|>\epsilon)<=2exp^{-2\epsilon 2M}$$

In a given binary classification problem, the output is 1 or -1. Here, μ represents out-of-sample error $E_{out}(h)$, which is an expected error as a result of the preferred hypothesis h and v represents the in-sample error $E_{in}(h)$, which is an error arising as a result of the classification of data points by hypothesis.

It is represented as follows:

$$P(|E_{in}(h) - E_{out}(h)|> \epsilon) <= 2exp^{-2\epsilon 2}$$

In Sample Error and Out of Sample Error

The data points used for building a model are referred to as in-sample data. The In-sample error may be defined as the error that is obtained on the same data set that is used for building the machine learning model. In-sample error is also referred to as a resubstitution error.

The data points that are new and do not belong to any of the training data samples are referred to as out-of-sample data. Out-of-sample error may be defined as the error that is obtained on the new data set. It is also referred to as the generalization error.

Generally, the In-sample error is always less than the Out-of-sample error. The In-sample error occurs as a result of Overfitting, as we overtrain the machine learning model with datasets. If the In-sample error or the error due to the already known data occurs; then the reason is obvious. We should care about eliminating the Out-of-sample error that occurs due to new datasets *i.e.* as a result of Underfitting.

APPLICATIONS OF MACHINE LEARNING

Today, Machine learning has become very popular. It is a buzzword these days. There are enormous applications of machine learning [2]. Some of the applications are:

Virtual Personal Assistants

The virtual personal assistants popularly used today include Alexa, Siri, and Google. These virtual personal assistants assist in getting information, whenever asked over voice. Using these virtual personal assistants, we can ask questions like "*Which are the flights from India to Germany?*", "*What are the tasks that have to be performed today?*" For answering such queries, virtual personal assistants collect information or search previously asked queries or collect information from phone apps. Machine learning is considered as an integral part of virtual personal assistants as they collect information and refine it based on the previous information which is then used to generate results based on the given preferences. Virtual personal assistants are integrated to various platforms such as mobile apps (for example, Google Allo), smartphones (for example, Samsung Bixby on Samsung S8), smart speakers (for example, Amazon Echo, Google Home), *etc.* Virtual personal assistants are small, portable devices.

Traffic Prediction

For managing traffic, GPS navigation devices are used. GPS devices help in tracking the current location and velocity of a vehicle, and store the information in the central server. This information is used for generating the current traffic report. This reduces traffic jams and helps in congestion analysis. So, machine learning is used for estimating the areas where congestion can be found on the basis of daily GPS reports.

Online Transportation Networks

While booking a cab using an app, it determines the price of the ride. Machine learning is used to minimize the detour. It plays a very important role in predicting the travel time, the price of the ride, and in reducing detour.

Video Surveillance System

A single person cannot monitor multiple video cameras. A video surveillance system uses machine learning at its back end to detect unusual behavior in people, like napping, stumbling, standing, *etc.* The video surveillance system on detecting unusual behavior will alert a human attendant and prevent any mishap from taking place.

Social Media Services

Machine learning is vastly used in social media platforms for personalizing news feed and targeting better ads. Other applications of machine learning in social media services include the following:

People you May Know

Machine learning is based on the concept of gaining knowledge with experience. Facebook notices people that we connect with, the profiles that we often visit, our workplace, groups that we share, our interest, *etc.* On the basis of all this information, Facebook gives a suggestion of the list of people we would like to become friends with.

Face Recognition

When we upload a photograph of ours along with a friend, Facebook can immediately recognize that friend. This is because Facebook identifies unique features, poses, and projections and matches them with the photographs of the people in our friend list. It is a very complicated application at the back end as it considers the precision factor, but at the front end, it is a very simple application of machine learning. Facebook uses DeepFace, which is a deep learning project responsible for the identification of a person's face in images. Facebook also provides a feature of alternative tags for an image that is already uploaded on Facebook.

Similar Pins

Computer vision involves extraction of useful information from videos and images. Machine learning is one of the applications of computer vision. Pinterest makes use of computer vision to detect pins or objects in an image and recommend pins which are similar to it.

Sentiment Analysis

Sentiment analysis is one of the applications of machine learning. It is a process of determining emotion or the opinion of a person from the given text. It is also referred to as opinion mining or sentiment classification. The application of sentiment analysis is found in decision-making applications, review-based websites, *etc.*

Email Spam and Malware Filtering

Email clients use numerous spam filtering techniques. In order to ensure that these approaches are continuously updated, machine learning is used. A rule-based spam filtering technique does not track the latest tricks adopted by the spammers. Examples of a spam filtering technique that is influenced by machine learning are multilayer perceptron, C 4.5 Decision Tree Induction, *etc.* Every day, nearly 325,000 malwares are detected and about 90-98% of the code is similar to its previous version. The system security programs that are based on machine learning are able to the coding pattern. Hence, they are able to find malware with a variation of 2-10% and also provide protection against them.

Online Customer Support

Many websites today provide the facility to chat with the company's customer support representatives while the user is scrolling through the website. But not all the websites provide live executives to answer the queries. In some websites, the user talks to a chatbot. These chatbots extract useful information from the website and present it to the customer as a response to their query. Chatbots have advanced with time. They are based on a machine learning algorithm that gains knowledge from past experiences [3]. Chatbots try to understand the user queries and then serve them with better responses every time.

Result Refinement of a Search Engine

Google and many other search engines make use of machine learning in order to improve the searching capability. Whenever we perform a search, an algorithm is run at the back end to see how we respond to the results provided by a search engine. If we open the top-most result and stay on the page for a very long time, then the search engine assumes that the result displayed is appropriate in accordance with the query. Also, if we reach the second or the third page of the search results but do not open any of the pages, then the search engine assumes that the result displayed is not in accordance with the query. In this way, the algorithm running at the back end tries to improve the performance of the search results.

Product Recommendations

When we shop an online product from a website, we notice that we start receiving emails containing shopping suggestions. Also, the app or the shopping website recommends items that match our choices. Product recommendations are displayed on the basis of past purchases, items browsed or added to the cart, or brand preferences. This magical shopping experience is due to machine learning.

Online Fraud Detection

With the help of machine learning, we can track online financial frauds. For example, to prevent money laundering, Paypal uses the machine learning approach. Using certain tools, Paypal can compare millions of transactions occurring between buyers and sellers, and be able to distinguish between legal and illegal transactions.

Online Gaming

Machine learning is used in online gaming. For example, in Chess, it uses machine learning. On the basis of previous moves that gave winning situations, the algorithm running at the back end tries to improve its performance and make similar moves that it considers the best. In this way, machine learning tries to imbibe human-like intelligence in computer that can play like any human chess champion.

Financial Services

According to the experts, online credit card fraud has risen to $32 billion in 2020. Companies that are related to the financial sector can see the flow of financial data and prevent financial fraud. This is possible using machine learning. Machine learning helps in the identification of opportunities in trading and investment. With the help of cyber surveillance, we can identify those institutions and individuals that are at financial risk and take timely action to prevent any cyber fraud.

Healthcare

There has been an advancement in technology in the field of medical healthcare by the incorporation of machine learning [4]. Wearable sensors and devices can provide real-time overall health conditions of a person, such as his heartbeat, blood pressure, *etc.* A doctor can use this information for analyzing the health condition of an individual, drawing patterns using patient history, and predicting

any sort of ailment in the future. A machine may also be trained to have human-like intelligence. A machine may behave as a doctor and predict a disease or suggest medicine on the basis of past health condition records.

Oil and Gas

This industry requires the need of machine learning the most. Machine learning applications in oil and gas industry are vast. It not only includes streaming oil distribution but also finding new sources of energy and analyzing underground minerals.

Self-driving Cars

A self-driving car is one of the latest and most exciting applications of machine learning. Tesla is a famous car manufacturing firm, which is working on developing self-driving cars. It is building a self-driving car using an unsupervised machine learning algorithm that is able to detect objects and people while driving.

Automatic Text Translation

Today, if you visit any new place, the language is not a barrier to understand the thoughts of the locals or to share your thoughts with them. With the help of machine learning, we can perform translation from one language into another, and also perform the conversion of text to speech and *vice versa*. GNMT (Google NeuralMachine Translation) is based on neural machine learning that performs the translation of text from one language into another language. GNMT is also referred to as an automatic translation system.

Dynamic Pricing

Dynamic pricing refers to the pricing strategy that wholly depends on the objective thought. For example, plane ticket, movie ticket, cab fare, *etc.* are dynamically priced. Using machine learning, buying trends can be found out and the dynamic prices of products can be determined. Uber uses a machine learning model called *Geosurge* to perform the dynamic pricing of individual rides.

Classification of News

Machine learning helps in the classification of vast news into different categories like National News, International News, Sports News, Entertainment, *etc.* These help the readers to choose the news from their desired category. Machine learning methods used for the classification of news are: Support Vector Machine, Naive Bayes, K-Nearest Neighbor, *etc.*

Information Retrieval

It is one of the significant applications of machine learning as it involves the extraction of knowledge or structured data from unstructured data. Information retrieval plays a crucial role in the big data sector. In the machine learning approach, unstructured data is taken as input and knowledge or structured data as output.

Robot Control

One of the applications of machine learning is robot control. Recently, research was carried out to obtain control over helicopter aerobatics and flight. In a robot control-based competition sponsored by Darpa, a robot that drove in a desert for one hundred miles won over a robot that could notice distant objects.

CONCLUSION

Machine Learning is a new and a growing field. This chapter discusses in detail about steps involved in building a Machine Learning system and Machine Learning Process, perspectives and issues involved while building a machine learning system, In-sample and Out-of-sample error and Applications of Machine Learning.

EXERCISES

1. Explain steps involved in designing a Machine Learning system.

2. Explain the feasibility of learning an unknown target function with an example.

3. Explain differences between In-sample Error and Out-of-sample Error.

4. Explain with example Application of Machine Learning in the Healthcare sector.

REFERENCES

[1] J.G. Carbonell, R.S. Michalski, and T.M. Mitchell, "An overview of machine learning", *Mach. Learn.,* vol. 1, pp. 3-23, 1983.

[2] F. Recknagel, "Applications of machine learning to ecological modelling", *Ecol. Modell.,* vol. 146, no. 1-3, pp. 303-310, 2001.
[http://dx.doi.org/10.1016/S0304-3800(01)00316-7]

[3] P. Langley, and H.A. Simon, "Applications of machine learning and rule induction", *Commun. ACM,* vol. 38, no. 11, pp. 54-64, 1995.
[http://dx.doi.org/10.1145/219717.219768]

[4] D.S. Char, N.H. Shah, and D. Magnus, "Implementing machine learning in health care-addressing ethical challenges", *N. Engl. J. Med.,* vol. 378, no. 11, pp. 981-983, 2018. [http://dx.doi.org/10.1056/NEJMp1714229] [PMID: 29539284]

CHAPTER 3

Linear Regression and Logistic Regression

Abstract: Supervised learning is a machine learning task of mapping the input to the output on the basis of labeled input-output example pairs. Supervised learning may be of two types: classification and regression. In this chapter, we will discuss linear regression in one variable, linear regression in multiple variables, gradient descent, and polynomial regression.

Keywords: Data science, Jupyter, Machine learning, Matplotlib, Numpy, Python, Scikit learn, SciPy.

INTRODUCTION

Supervised Learning involves training a machine using a labelled dataset so that the machine exhibits human-like intelligent behaviour. The training data in supervised learning comprises input data and corresponding output data. A Supervised Learning Algorithm is capable of performing mapping of a given input variable with the corresponding or related output variable. It involves two approaches: regression and classification. Regression is a supervised learning approach in which output in the form of real values is predicted. Classification is a supervised learning approach in which output in the form of discrete values is predicted.

LINEAR REGRESSION

Linear regression is a technique to depict the relationship between an independent variable x and a dependent variable y. Linear regression states that the relationship that exists between one or more input features and the relative output or target vector is approximately linear in nature. Linear regression finds the weighted sum of the input features along with the constant referred to as bias term or intercept [1]. Linear regression has numerous real-life applications. These applications fall into two categories:

If the application comprises forecasting, prediction, or error reduction, then linear regression may be applied to the data set values and make predictions in response.

When there is a need for variations in the response, then it may be attributed to the presence of other explanatory variables. Linear regression is used to find possible relationships between the variables in the field of behavioral, biological, and social sciences. In linear regression with one variable, the hypothesis is defined as:

$$h_\theta (x) = \theta_0+\theta_1*x$$

Here, *x* is referred to as an independent variable on which our hypothesis depends. For example, '*Rainfall*' measured in mm could be *x* and '*The Number of Umbrellas sold*' could be the hypothesis that we are trying to predict. θ_0 and θ_1 are referred to as the bias variable and weight variable, respectively and they together constitute the weight matrix.

Cost function is an equation that gives an estimate of how close we are to the hypothesis. The smaller the value of cost function, the closer we are from the required curve. So, we try to minimize the cost function in order to reduce errors. The mean squared error cost function is defined as follows:

$$J(\theta_0, \theta_1) = 1/2m \sum i = 1 \text{ to } m(h_\theta(x^i) - y^i)^2$$

Here, *J* is a cost function, *m* refers to the number of data points in our data set, and *y* refers to the actual values that we will like to predict.

Linear regression is a part of predictive modelling in which the value of output corresponding to a given input depends on the previous values of data.

Our goal is to estimate the fit of the line. Best fit means that the accuracy in prediction is more and chances of error are least.

Types of Linear Regression include the following:

- Simple Linear Regression or Linear Regression in one variable- It comprises one independent variable. *E.g.* price of a new house to be purchased depends on its size.
- Multiple Linear Regression or Linear Regression in multiple variable- It comprises multiple independent variables. *E.g.* price of a new house to be purchased depends on multiple factors such as its size, economy, area or society *etc.*

Applications of Linear Regression:

1. It may be used for forecasting business trends and analysis.

2. It may be used for analyzing sales, marketing promotions, pricing, forecasts, *etc.*

3. It is used predominantly in the field of economics for predicting and analyzing the amount of export, expenditure, labour demand and supply, inventory investments, *etc.*

4. In a given biological system, a Linear Regression may be used for modelling causal relationships among various parameters.

Linear Regression In One Variable

Linear Regression is a simple linear regression that comprises one independent variable. Hypothesis function hθ helps in mapping input data with output data. The cost function is a square error function used in regression that helps in determining the difference accurately.

Gradient Descent algorithm helps in reducing the cost function value. In the regression problem, we try to map input values with the corresponding continuous output function. Linear Regression in one variable is also referred to as univariate linear regression. In Linear Regression using one variable, for a given one input x, there is a prediction of only one output y. In Linear Regression, the best fit line or regression line can be found by computing delta which refers to the difference between the actual data points and the line. These errors are squared and then summed. The line with least value of sum of squared errors is chosen as the regression line. In a Linear Regression graph, the independent variable is plotted on x-axis and the dependent variable is plotted on y-axis.

Linear Regression In Multiple Variables

Linear regression in multiple variables explains the relationship between a dependent variable *y* and many independent variables [2, 3, 4].

Instead of a 'Yes' or 'No' reply, the value of *Y* will be a number. It is a continuous dependent variable. Here, the theta values are referred to as regression weights and computed in such a way as to minimize the sum of the squared deviations.

Assumptions that need to be kept in mind while performing Linear Regression include the following:

- Measurement of variables must be done at regular intervals.
- A scatterplot is used to find out the relationship between the given variables.
- During observations, no dependency among the variables must be present.
- There should not be any outliers in the given data.
- Checking if Homoscedasticity exists. It is a situation in which variances remain exactly the same along the best-fit line.
- In a best fit linear regression, errors follow a normal distribution pattern.

Overfitting and Regularization In Linear Regression

Overfitting means when a model is overtrained on the training data, then it performs poorly on the testing data. In Overfitting, the machine learning model learns the noise along with the data. This hinders the performance of a ML system. There exists inflation in coefficients as a result of overfitting. Regularisation helps in adding a penalty to the coefficients and helps the model to achieve a generalised state in the overfit condition.

In regularization, in the cost function; a penalty coefficient is added in order to generalise the model. Regularisation is of two types L1 form or Lasso and L2 form or Ridge. In Lasso or L1 form, the penalty is an absolute value of the magnitude of the coefficient which is added to the loss function. In Ridge or L2 form, the penalty is the squared value of the coefficient which is added to the loss function.

GRADIENT DESCENT

Gradient descent is a method to minimize the value of the Cost Function. It can alter the values of Theta 0 and Theta 1 of a point based on the slope or gradient of the Cost Function curve. The changes introduced in the values of Theta 0 and Theta 1 also bring changes to the hypothesis, thereby bringing a better fit to the data. Gradient descent estimates the derivative of the cost function. It is represented by the following formula:

$$\theta^{new} = \theta^{old} - \alpha 1/m \sum i = 1 \; to \; m \; [(h_\theta X - y)X^T]$$

Here, $h_\theta X$-y is referred to as an error.

is referred to as the learning rate. In this method, we first fetch and clean the data and analyze it. Then, we define the hypothesis, re-gression parameters, and cost function. Then, we run the gradient descent algorithm on all the data points and update the hypothesis.

POLYNOMIAL REGRESSION

Polynomial regression is a type of regression analysis in which the relationship that exists between a dependent variable y and an independent variable x is modeled by the nth degree of the polynomial x [4]. It fits a nonlinear relation that exists between x and its corresponding conditional mean of y. This is denoted as E(y|x). In some of the cases, for the unknown parameters that are collected from the data, E(y|x) is linear. So, polynomial regression is referred to as a special case of linear regression with multiple variables.

Polynomial regression is represented as follows:

$$Y= \theta_0+\theta_1 X+ \theta_2 X^2+ \theta_3 X^3 \ldots\ldots \theta_n X^n$$

Polynomial regression is preferred when the relation between the given variables is curvilinear.

There are two ways in which we can define the degree of equation in polynomial regression. These two ways include the following:

- Forward Selection: It is a method of increasing the degree until a significant model is defined.
- Backward Selection: It is a method of decreasing the degree until a significant model is defined.

Polynomial Regression finds applications in many organizations when they wish to predict the relationship between independent and dependent variables which is non-linear in nature.

Advantages of polynomial regression include the following:

- The Polynomial Regression Equation is used for solving many experimental procedures.
- It is used for studying isotopes of given sediments.
- It is helpful in studying the increase and spread of diseases in a given population.

Features of Polynomial Regression

- It is a kind of nonlinear regression method that defines the relationship between the independent and dependent variable.
- The degree present in a given polynomial equation defines the best fit line.

The model derived from the polynomial regression is affected by the outliers so it is always better to treat outliers before applying the algorithm to the dataset.

The polynomialfeature () function converts into a feature of the matrix depending on the degree of the equation.

The nature of the curve can be studied or visualized by using a simple scatter plot which will give you a better idea about the linearity relationship between the variables and decide accordingly.

LOGISTIC REGRESSION

Logistic regression is a statistical model that makes use of the logistic function to model a binary dependent variable. It may be defined as the transformation of linear regression *mot del* that can probabilistically model the binary variables.

The output in linear regression is continuous, whereas, in logistic regression, the output is discrete. Assumptions of logistic regression include the following:

• In logistic regression, the dependent variable should be binary.

• In logistic regression, linearity must exist between logit and independent variables.

There is no chance of multicollinearity. It requires a large sample size. It removes outliers and misclassified instances from the training data. Logistic regression assumes that there is no error in the output variable.

Overfitting and Regularisation in Logistic Regression

Overfitting causes the addition of noisy data to the training set. Regularisation and removing unnecessary features help in overcoming the overfitting problem. In regularisation, a penalty or a coefficient is added to the loss function. In Logistic Regression, in regularisation, the penalty term added is $\lambda/n \Sigma wi2$

BINARY CLASSIFICATION AND MULTI-CLASS CLASSIFICATION

Binary classification, also referred to as binomial classification, may be defined as the process of classification of elements into two groups on the basis of the classification rule. It is found in the following fields:

• In the medical field, it is used to test whether a patient has any disease or not.

• In factories, it is used to decide whether a product is in accordance with the quality standards or not, or to check if some specification is met or not.

• In information retrieval, it is used in deciding whether a given page or an article should be in the result set of a search or not on the basis of the relevance of an article or its usefulness to the user.

• In spam filtering, it is used to decide whether an email message received is a spam mail or not.

• It is also used in credit card fraudulent transaction detection.

Some of the techniques used for learning binary classifiers include the following:

- Decision trees
- Neural networks
- Support vector machines
- Bayesian classification

There are different testing analysis methods used in binary classification. Consider the following testing data as shown in Table **1** on which we will apply the testing analysis methods:

TABLE 1. Testing Data on which Testing analysis methods is applied.

Instance	Target	Outcome
1	1	0.89
2	1	0.75
3	1	0.60
4	1	0.50
5	0	0.45
6	1	0.44
7	0	0.42
8	1	0.41
9	0	0.42
10	1	0.39
11	1	0.31
12	0	0.30
13	0	0.18
Instance	**Target**	**Outcome**
14	0	0.17
15	0	0.16

(Table 1) cont.....

Instance	Target	Outcome
16	0	0.15
17	0	0.14
18	0	0.13
19	0	0.10
20	0	0.10

Let us now consider the following testing analysis methods:

Confusion Matrix: It is used to present the performance of a binary classifier. The decision threshold, *T* may be used to find out whether the given set of instances is positive or negative. If the probability allotted to the given instance of a classifier is greater than *T*, then it is referred to as *positive,* otherwise, it is referred to as *negative*. When all the given testing instances are classified, then the target labels are compared with the outcome labels to generate the following four terms:

- **True positives (TP)** – The total number of instances that are positive and are classified as positive.
- **False positives (FP)** – The total number of instances that are negative and are classified as positive.
- **False negatives (FN)** – The total number of instances that are positive and are classified as negative.
- **True negatives (TN)** – The total number of instances that are negative and are classified as negative.

The confusion matrix is represented in Table **2**.

TABLE 2. Confusion Matrix.

-	Predicted Positive	Predicted Negative
Real Positive	True Positives	False Negatives
Real Negative	False Positives	True Negatives

Here, the column represents the outcome classes and the rows represent the target classes. The diagonal cells show the number of cases that are classified correctly and the off- diagonal cells show the number of cases that are not classified correctly.

Let the value of the decision threshold be *T=0.4*. Consider the testing data table given above. We get the following confusion matrix(as shown in Table **3**)

TABLE 3. Output Confusion Matrix.

-	Predicted Positive	Predicted Negative
Real Positive	6	2
Real Negative	3	9

Binary Classification Tests

These are the parameters that are derived from the confusion matrix. They involve the following parameters:

Classification Accuracy

It may be stated as the ratio of the instances that are correctly classified. It can be depicted as follows:

Classification_Accuracy = (True_Positives + True_Negatives)/ (Total instances)

Error Rate

It may be stated as the ratio of instances that will not be classified correctly. It can be depicted as follows:

Error Rate= (False_Positives + False_Negatives) /(Total instances)

Sensitivity

It can be stated as the ratio of the correct positives and the total number of positives. It is also referred to as recall or true positive rate. It can be depicted as follows:

Sensitivity= (True_Positives) / (Total Positive instances)

Specificity

It can be stated as the ratio of the correct negatives and the total number of negatives. It is also referred to as the true negative rate. It can be depicted as follows:

Specificity = (True_Negatives) /(Total Negative instances)

From the above mentioned confusion matrix, we obtain 25% error rate, 75%

accuracy, 75% sensitivity, and 75% specificity.

Multinomial or multiclass classification may be defined as the problem of the classification of instances into two or more classes. Multiclass classification makes sure that each sample is assigned only one label.

PYTHON CODES

1.

The Python code for linear regression in one variable is given below. Initially, all python libraries are imported. Then, a dataset is imported which comprises information of no. of umbrellas sold corresponding to rainfall.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import future

df=pd.read_csv("C:\\Users\\computer\\Desktop\\umbrella. csv") # Data is uploaded and a DataFrame is #created using Pandas pd object

df

# The strength of the relationship is found between Rainfall and Umbrellas Sold

X = np.asarray(df.Rainfall.values)

y = np.asarray(df.UmbrellasSold.values)

# Scaling and Normalization of the features are performed

def FeatureScalingNormalization(X):

# Xnorm is a copy of X vector Xnorm = X

# avgx will the contain average value of x in the training set

avgx = np.zeros(X.shape[0])

# rangex will contain the standard deviation values of x rangex = np.zeros(X.shape[0])
```

```
avgx = X.mean()

rangex = X.std(ddof=1) # Calculated with NumPy. It requires degreeoffreedom=1

# The number of training examples is stored in p p = X.shape[0]

# a vector of size p with the average values of x avgx_matrix = np.multiply(np.ones(p), avgx).T

# a vector of size p with the standard deviation values rangex_matrix = np.multiply(np.ones(p), rangex).T

# Normalization is applied on x values Xnorm = np.subtract(X, avgx).T

Xnorm = Xnorm /rangex.T return [Xnorm, avgx, rangex]

featuresNormalizeresults = FeatureScalingNormalization(X) # normalized X matrix is obtained

X = np.asarray(featuresNormalizeresults[0]).T # mean values are obtained

avgx = featuresNormalizeresults [1]

# standard deviation values are obtained rangex = featuresNormalizeresults [2]

X

p = len(y) # number of training examples

X = np.vstack((np.ones(p), X.T)).T # Training Examples, column of 1's is added to X

X
```

```
array ([[ 1.        ,  -1.09702635],
        [ 1.        ,  -0.73221949],
        [ 1.        ,  -0.99279582],
        [ 1.        ,  -0.55850193],
        [ 1.        ,  0.6887901 1],
        [ 1.        ,   1.07444307],
        [ 1.        ,   1.92218473],
        [ 1.        ,    1.0223278 ],
        [1.         ,   0.54634171],
        [ 1.        ,  -0.49943797],
        [ 1.        ,  -0.91288574],
        [ 1.        ,  -0.4612201 ]])
```

plt.scatter(X[:, [1], y, color='blue') # Data is plotted and the Scatter plot is obtained

plt.xlabel("Rainfall") plt.ylabel("Umbrellas Sold")

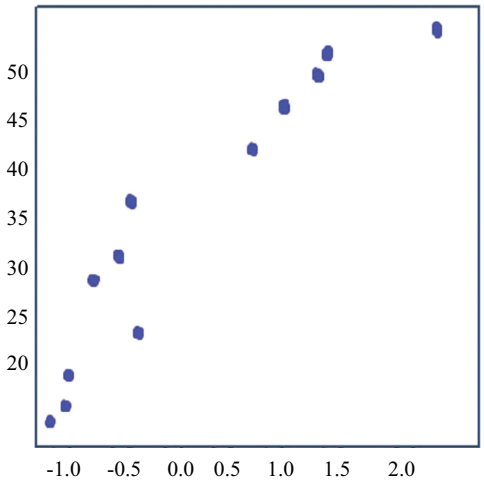

Fig. (1). Linear regression plot showing the relationship between Umbrellas Sold and Rainfall.

In the above graph Fig. (**1**), we can visualize an increasing pattern in the relationship between rainfall and the umbrellas sold.

We calculate the plot when two parameters, Theta is randomly chosen as [140.0,5.0]

theta_0 = 140.0

```
theta_1 = 5.0

theta = np.asarray([theta_0,theta_1]).astype(float)

# Plot the data

plt.scatter(X[:, [1], y, color='black')

# corresponding to the Hypothesis model, the red line is plotted.

plt.plot(X[:, [1], np.sum(np.multiply(X,theta), axis=1), color='red', linewidth=1)

plt.xlabel("Rainfall") plt.ylabel("Umbrellas Sold")
```

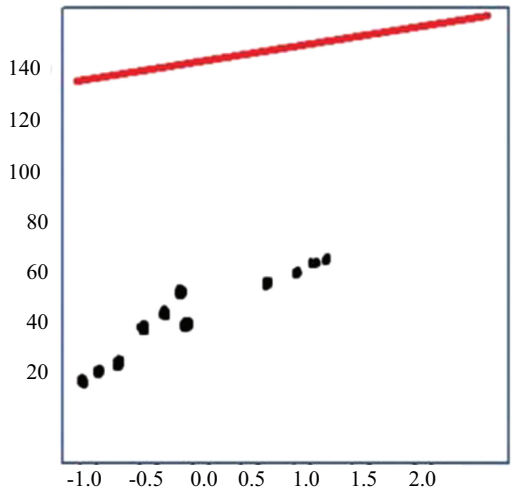

Fig. (2). Linear regression plot when theta is randomly chosen.

Fig. (**2**) shows plot in linear regression when theta is randomly chosen.

2.

Consider the following code that implements linear regression in multiple variables using scikitlearn and statsmodels:

```
import pandas as pd

from sklearn import linear_model import statsmodels.api as am

Stock_Market = {'Year': [2018,2019,2018,2017,2017, 2016,201 7,2019,2018, 2018,2019,2019, 2016,2017,2017,2018,2018,2018, 2018,2018, 2016,2016, 2016, 2016], 'Month': [10,12,10,9,8, 4,6,5,7,5, 1, 2,12,10,11,12,8,9,6, 4,5, 1, 3, 2]
```

```
,'Rateofinterest':[3.25,4.5,3.5,5.5,4.5,4.5,3.5,5.25,6.25,4.25,5,5,6,5.75,4.75,5.75,5.
75,4.75,5.75,4.75,3.75,4.75,5.75,5.75],'RateofUnemployment':[7.3,4.3,3.3,4.3,6.4,
5.6,4.5,5.5,6.5,4.6,6.7,5.9,6,4.9,6.8,5.1,7.2,5.1,6.1,7.1,6.9,5.2,7.2,5.1],'Stock_price
_index':[1764,1594,1457,1493,1756,1244,1254,1175,1329,1547,1230,1375,1057,
945,947,958,918,935,834,786,815,852,724,749] }

df = pd.DataFrame(Stock_ Market,columns=['Year','Month','Rateofinterest',
'RateofUnemployment','Stock_price_index'])

X = df['Rateofinterest','RateofUnemployment']

# here we have used 2 variables in Linear Regression using #Multiple Variables

Y = df['Stock_price_index'] # Using sklearn

regr = linear_model.LinearRegression()

regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)

print('Coefficients: \n', regr.coef_)

# prediction using sklearn New_Rateofinterest = 3.25

New_RateofUnemployment = 7.3

print ('Predicted Stock Price Index: \n', regr. predict([New_Rateofinterest,
New_RateofUnemployment]))

# using statsmodels

X = sm.add_constant(X) # adding a constant

model = sm.OLS(Y, X).fit()

predictions = model.predict(X)

printmodel = model.summary() print(printmodel)
```

Intercept:
2270.912813383781
Coefficients:
[-158.95784362 -57.90074501]
Predicted Stock Price Index:
[1331.62438306]

OLS Regression Results

Dep. Variable:	Stock_price_index	R-squared:	0.242
Model:	OLS	Adj. R-squared:	0.170
Method:	Least Squares	F-statistic:	3.355
Date:	Wed, 13 May 2020	Prob (F-statistic):	0.0544
Time:	13:34:40	Log-Likelihood:	-169.00
No. Observations:	24	AIC:	344.0
Df Residuals:	21	BIC:	347.5
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	(0.025	0.975)
const	2270.9128	446.069	5.091	0.000	1343.261	3198.564
Rateofinterest	-158.9578	73.156	-2.173	0.041	-311.093	-6.822
RateofUnemployment	-57.9007	56.269	-1.029	0.315	-174.919	59.117

Omnibus:	1.697	Durbin-Watson:	0.673
Prob (Omnibus):	0.428	Jarque-Bera (JB):	1.267
Skew:	0.344	Prob (JB):	0.531
Kurtosis:	2.109	Cond. No.	57.4

Fig. (3). Output of Linear Regression using Multiple Variables.

The above code displays values of intercept, coefficient, and predicted stock price index using sklearn and statsmodels:

StockPrice_Index = Intercept + (Rate of interest_Coefficient)*X_1 + (Rate of

Unemployment_Coefficient)*X_2

Substituting the values of intercept and coefficient from sklearn, we get:

Stock Price_Index=2270.9128 + (-158.9578)*X_1 + (-57.9007)*X_2

From the table above, we infer that we get the same values of inter- cept, coefficient, and predicted stock price index using sklearn and statsmodels.

Fig. (**3**) depicts output of Linear regression using multiple variables.

3.

This Gradient descent algorithm is implemented in Python as follows:

def gradientDescent(X, y, theta, alpha, numiters): # number of training examples

p = len(y)

Initialize J_history and Theta_history Jhistory = []

Thetahistory = []

for i in range(numiters):

Calculate h = X * theta

h = np.sum(np.multiply(X,theta), axis=1) # Calculate the error = (h - y)

error = np.subtract(h, y) # Calculate the new theta

thetanew = alpha * 1/p * np.sum(np.multiply(X.T, error), axis=1)

Update theta

theta = np.subtract(theta, thetanew) # Collect all the theta and J Thetahistory.append(theta.tolist())

Jhistory.append(calcCostFunction(X,y,theta).

tolist())

return theta, Thetahistory, Jhistory # Running the Gradient Descent

Initialize theta

theta = np.asarray([0,0]).astype(float)

Set the number of iterations for the Gradient Descent iterations = 2000

Set the Learning Rate alpha = 0.01

Run the gradientDescent() function, and collect the output in “results”

results = gradientDescent(X, y, theta, alpha, iterations) # Get the theta from the results

theta = results[0] # new theta # Get the theta history

Thetahistory = results [1] # Theta history # Get the J history

Jhistory = results [2] # Cost function history plt.scatter(X[:, [1], y, color='blue')

Plot Hypothesis (theta as calculated with the Gradient Descent)

plt.plot(X[:, [1], np.sum(np.multiply(X,theta), axis=1), color='red', linewidth=1)

plt.xlabel(“Rainfall”) plt.ylabel(“Umbrellas Sold”)

The plot of rainfall versus umbrella in gradient descent is depicted using Fig. (**4**) .

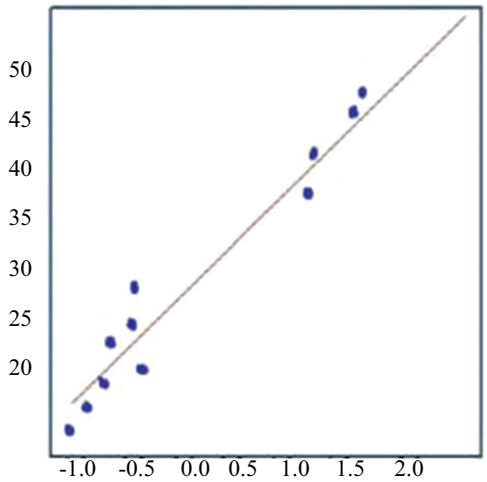

Fig. (4). Plot between rainfall and umbrellas.

Now, we'll plot the Theta history (as shown in Fig. (**5**) as follows:

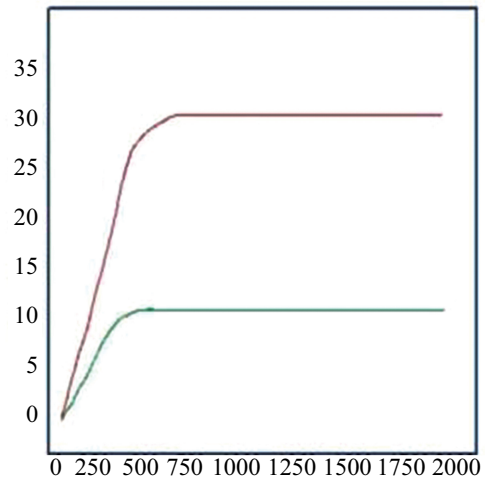

Fig. (5). Plot of Theta0 and Theta1 values.

```
theta_0 = np.asarray(Thetahistory)[:,[0]
theta_1 = np.asarray(Thetahistory)[:, [1]
plt.plot(theta_0[0:len(theta_0)], color='red', linewidth=1)
plt.plot(theta_1[0:len(theta_1)], color='green', linewidth=1)
```

plt.xlabel(“Iterations”) plt.ylabel(“theta”)

We have performed 2000 iterations. In the above figure, the red curve represents θ0 and the green curve represents θ1. After 2000 iterations, the value of θ is [37.5,11.0]. Next, we'll plot the J history as follows:

plt.plot(Jhistory[0:len(Jhistory)], color='blue', linewidth=1)

Put labels

plt.xlabel(“Iterations”) plt.ylabel(“J”)

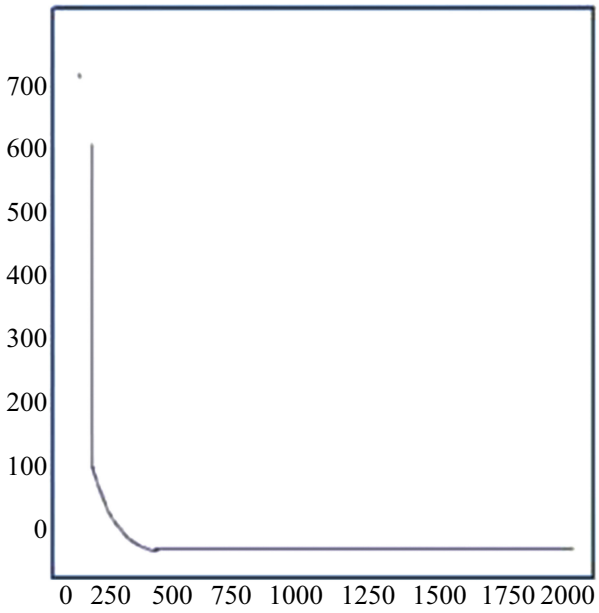

Fig. (6). J history plot.

The above graph (as shown in Fig. (**6**) is a J history plot that shows the value of the cost function falls down after 200 iterations and then becomes stable till 1500 iterations. The minimum value of J is 57.5:

Now if we predict the rainfall in mm to be 82mm, we predict the corresponding number of umbrellas sold.

query = [1, 82]

Scale and Normalize the query queryNormalized = [1, ((query [1]-avgx)/rangex)]

prediction = np.sum(np.multiply(queryNormalized, theta)) prediction

20.373966510153515

So, approximately 20 umbrellas are sold:

Drawing a contour plot of J and θ nslice = 50

theta0_vals = np.linspace(-100, 200, num=nslice) theta1_vals = np.linspace(-400, 400, num=nslice)

Initialize J_val that will collect all the J values

calculated by calcCostFunction()

J_vals = []

for i in range(len(theta0_vals)): for j in range(len(theta1_vals)):

t = np.asarray([theta0_vals[i], theta1_vals[j]). astype(float)

J_vals.append(calcCostFunction(X, y, t).tolist()) J_vals = np.asarray(J_vals)

J_vals = J_vals.reshape((nslice, nslice)).T levels = nslice

s = 1

plot the contour with theta and the J values plt.contour(theta0_vals, theta1_vals, J_vals, levels) # Draw the path of the Gradient Descent convergence for k in range(0, iterations, 10):

plt.scatter(np.asarray(Thetahistory)[k][0],\

np.asarray(Thetahistory)[k] [1], color='blue',

s=s)

Draw a red dot i correspondence of the theta associated to the minimum J

plt.scatter(theta[0], theta [1], color='red', s=10) plt.xlabel(“theta_0”)

plt.ylabel(“theta_1”)

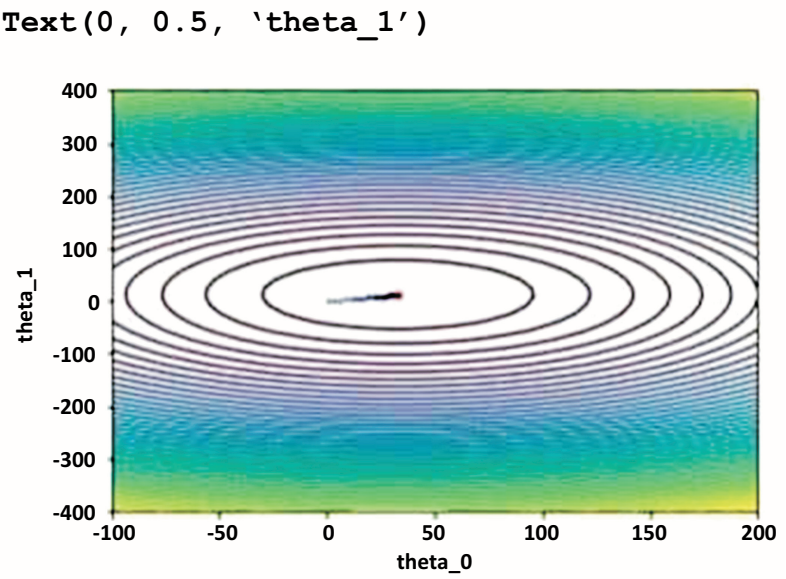

Fig. (7). Contour plot of J and Theta.

In the above plot as shown in Fig. (**7**) , the red dot at the center represents a minimum value

of J, J=57.5 at θ=[37.5,11.0].

4.

Consider the following code in Python on polynomial regression:

import operator import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score from sklearn.preprocessing import PolynomialFeatures np.random.seed(0)

x = 1 - 3 * np.random.normal(0, 1, 50)

y = x - 3 * (x ** 2) + 0.5 * (x ** 3) + np.random. normal(-3, 3, 50)

Data is transformed to include another axis x = x[:, np.newaxis]

y = y[:, np.newaxis]

```
polyfeatures= PolynomialFeatures(degree=3)

xpoly = polyfeatures.fit_transform(x)

model = LinearRegression() model.fit(xpoly, y)

ypoly_pred = model.predict(xpoly)

rmse = np.sqrt(mean_squared_error(y,ypoly_pred))

r2 = r2_score(y,ypoly_pred)

print(rmse) print(r2)

plt.scatter(x, y, s=20)

# The values of x are sorted according to the degree before the line plot

sort_axis = operator.itemgetter(0)

sorted_zip = sorted(zip(x,ypoly_pred), key=sort_axis)

x, y_poly_pred = zip(*sorted_zip)

plt.plot(x, ypoly_pred, color='m')

plt.show()
```

In the above code, we performed a polynomial regression, where the degree of the independent variable x is 2. We also plotted the graph and generated the RMSE and the R2 Score of the resultant curve. The output of the above code is shown in Fig. (**8**) .

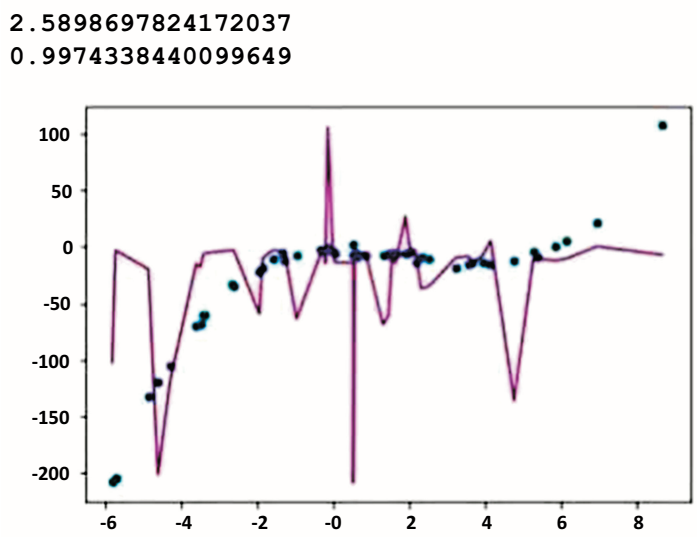

Fig. (8). Plot on polynomial regression.

The RMSE score and the R2 Score obtained are 2.5898697824172037

and 0.9974338440099649, respectively.

y.

5.

Consider an example of a logistic regression, given number of hours, a student studied for exams and number of hours, a student slept. This is shown in Fig. (**9**) . We have to predict whether a student will pass (represented by 1) or fail (represented by 0). It is represented by the following Python code:

```
import pandas as pd

import numpy as np

from sklearn import metrics

from sklearn.linear_model import LogisticRegression from sklearn.model_
selection import train_test_split

df = pd.read_csv("C:\\Users\\computer\\Desktop\\student. csv")

df.head()
```

	No_of_hours_studied	No_of_hours_slept	PassorFail
0	8	5	1
1	4	6	0
2	12	3	1
3	2	4	0
4	1	7	0

Fig. (9). Example of Dataset on which Logistic regression is applied.

x = df.drop("PassorFail",axis = 1)

y = df.PassorFail

In the above code, the drop method is used to remove PassorFail attribute from x and, x is assigned to other attributes. PassorFail attribute is assigned to y.

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=4)

The above code splits the data set into 75% training data and 25%

testing data:

logistic_regression = LogisticRegression() logistic_regression.fit(x_train,y_train)

Here, the fit method is used to train the model. The predict method is used to perform predictions on x_test values. The output of the prediction is stored in y_prediction. The accuracy_score method of metrics class is used to estimate the accuracy of the model. The accuracy obtained in the logistic regression model is 50%.

y_prediction = logistic_regression.predict(x_test) accuracy = metrics.accuracy_ score(y_test, y_prediction)

accuracy_percentage = 100 * accuracy accuracy_percentage

50.0

first_student = logistic_regression.predict((np.array([7, 5]).reshape(1, -1)))

first_student

array([1], dtype= int64)

In the above code, we predict whether the first student will pass or fail on the basis of the number of hours studied; the number of hours slept is 7 and 5. The prediction result is that the first student will pass as indicated by the array ([1],dtype=int64). Also, we predict whether the second student will pass or fail on the basis of the number of hours studied; the number of hours slept is 2 and 10. The prediction result is that the second student will fail as indicated by the array([0],dtype=int64).

second_student = logistic_regression.predict((np.array([2, 10]).reshape(1, -1)))

second_student array([0], dtype=int64)

6.

Consider the following code in Python that performs a binary classification. Here, we have considered breast_cancer, which is a predefined data set in sklearn. This data set comprises 569 instances of tumors, 30 features or attributes such as texture, radius, area, and the smoothness of the tumor. It comprises two classes of tumors such as malignant and benign. Malignant tumors are represented by 0 and benign tumors are represented by 1:

from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score

Loading the predefined iris data set from Sklearn data = load_breast_cancer()

Organizing the data

labelnames = data['target_names'] labels = data ['target'] featurenames = data ['feature_names'] features = data['data']

data print(labelnames)

print('Class label = ', labels[0]) print(featurenames) print(features[0])

Splitting the data

train,test,train_labels,test_labels=train_test_ split(features, labels,test_size=0.33, random_state=42)

Initializing the classifier gnb = GaussianNB()

Training the classifier

model = gnb.fit(train, train_labels) # Making predictions

prediction = gnb.predict(test) print(prediction)

Evaluating the accuracy print(accuracy_score(test_labels, prediction))

The output of the above code is shown below:

```
['malignant' 'benign']
Class label =  0
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0
 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 1 1 1 0 0
 0 1 1]
0.9414893617021277
```

From the above code, we infer that the first data instance represents

0. So, it is a malignant tumor and its mean radius is 1.799e+01.

7.

Consider the following Python code on multiclass classification. Here, we use a predefined data set wine from sklearn. This data set comprises 178 instances of wine and 13 features or attributes related to the wine, such as the amount of alcohol, malic acid, ash, alkalinity of ash, the amount of magnesium, total phenol, flavonoids, non- flavonoid phenols, color intensity, hue, *etc.*

```
from sklearn.datasets import load_wine

from sklearn.model_selection import train_test_split from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score

# Predefined Sklearn Wine data set is loaded

data = load_wine()

# The data is organized labelnames = data['target_names'] labels = data['target']

feature-names = data['feature_names'] features = data['data']
```

```
['class_0' 'class_1' 'class_2']
Class label =  0
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', '
proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
 2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
[0 0 2 0 1 0 1 2 1 2 0 2 0 1 0 1 1 1 0 1 0 1 1 2 2 2 1 1 1 0 0 1 2 0 0 0 2
 2 1 2 0 1 1 1 2 0 1 1 2 0 1 0 0 2 2 1 1 0 1]
1.0
```

```
# Look at our data print(labelnames)

print('Class label = ', labels[0]) print(featurenames) print(features[0])

# Splitting our data

train,test, train_labels, test_labels =train_test_ split (features, labels, test_size=0.33, random_state=42)

# Initializing our classifier gnb = GaussianNB()

# Training our classifier

model = gnb.fit(train, train_labels) # Making predictions

prediction = gnb.predict(test) print(prediction)

# Evaluate the accuracy print(accuracy_score(test_labels, prediction))
```

The following is the output obtained from the above code:

In the above code, it shows that sample 1 is of class 0 and the amount of alcohol in it is *1.423e+01* . Here, the sample is classified into 3 classes of wines, such as class 0, class 1 and class 2.

Consider another code on multiclass classification. Here, we have taken iris, a predefined data set from sklearn. The iris data set comprises 150 instances of iris (a kind of flower)

and 4 features or

attributes of iris, such as the sepal length, sepal width, petal length, and petal width:

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score

# Loading the predefined iris data set from Sklearn data = load_iris()

# Organizing the data

labelnames = data['target_names'] labels = data['target'] featurenames = data['feature_names'] features = data['data']

# data print(labelnames)

print('Class label = ', labels[0]) print(featurenames) print(features[0])

# Splitting the data

train, test, train_labels, test_labels = train_test_ split(features,labels, test_size=0.33, random_state=42)

# Initializing the classifier gnb = GaussianNB()

# Training the classifier

model = gnb.fit(train, train_labels) # Making predictions

prediction = gnb.predict(test) print(prediction)

# Evaluating the accuracy
```

```
['setosa' 'versicolor' 'virginica']
Class label =  0
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[5.1 3.5 1.4 0.2]
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 2 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 1 1 0 0 1 1 2 1 2]
0.96
```

print(accuracy_score(test_labels, prediction))

The above code divides the sample into 3 classes of iris, such as setosa, versicolor, and virginica. It shows that the first sample of iris is of type setosa and its sepal length is 5.1 cm.

CONCLUSION

Supervised learning makes use of labelled input-output pairs for training. In this chapter, we have discussed Linear Regression and Logistic Regression. Overfitting and Regularisation in Linear Regression and Logistic Regression, Gradient descent and binary classification and multi-class classification.

EXERCISES

1. Explain the difference between Linear Regression and Logistic Regression.

2. Explain overfitting and regularisation in Linear Regression.

3. Explain overfitting and regularisation in Logistic Regression.

4. Explain the difference between binary classification and multi-class classification.

REFERENCES

[1] D. Maulud, and A.M. Abdulazeez, "A review on linear regression comprehensive in machine learning", *Journal of Applied Science and Technology Trends,* vol. 1, no. 4, pp. 140-147, 2020. [http://dx.doi.org/10.38094/jastt1457]

[2] K.H. Zou, K. Tuncali, and S.G. Silverman, "Correlation and simple linear regression", *Radiology,* vol. 227, no. 3, pp. 617-628, 2003. [http://dx.doi.org/10.1148/radiol.2273011499] [PMID: 12773666]

[3] R.J. Leatherbarrow, "Using linear and non-linear regression to fit biochemical data", *Trends Biochem. Sci.,* vol. 15, no. 12, pp. 455-458, 1990. [http://dx.doi.org/10.1016/0968-0004(90)90295-M] [PMID: 2077683]

[4] K.F. Nimon, and F.L. Oswald, "Understanding the results of multiple linear regression: Beyond standardized regression coefficients", *Organ. Res. Methods,* vol. 16, no. 4, pp. 650-674, 2013. [http://dx.doi.org/10.1177/1094428113493929]

CHAPTER 4

Support Vector Machine

Abstract: Support Vector Machine (SVM) may be defined as a machine learning algorithm that can be used for regression and classification. It is generally used for classification purposes. In this chapter, we will discuss Margin and Large Margin Methods and Kernel Methods.

Keywords: Data science, Jupyter, Machine learning, Matplotlib, Numpy, Python, Scikit learn, SciPy.

INTRODUCTION

Support Vector Machine (SVM) is one of the most popular supervised machine learning algorithms. It is used for performing classification and clustering tasks [1]. Usually, SVM is used in classification problems. Support Vector Machine creates a decision boundary referred to as a hyperplane that divides the n dimensional space into different classes or categories. After training an SVM model, the new data may be assigned the correct class to which it belongs.

SUPPORT VECTOR CLASSIFICATION

In SVM, the data item is plotted in an n-dimensional space, where n represents the number of features [1, 2]. A classification is performed by finding the hyperplane that can differentiate the two classes. Consider Fig. (**1**); Here, the classification of two different shapes is performed by finding the hyperplane:

Selecting the right hyperplane for a given problem can be done in the following ways:

1. Choose the hyperplane that classifies the data points in a better way. Consider Fig. (**2**). Here, we have three hyperplanes, namely A, B and C. We need to choose one hyperplane out of these. We choose hyperplane A as it classifies the data points efficiently as compared to other hyperplanes:

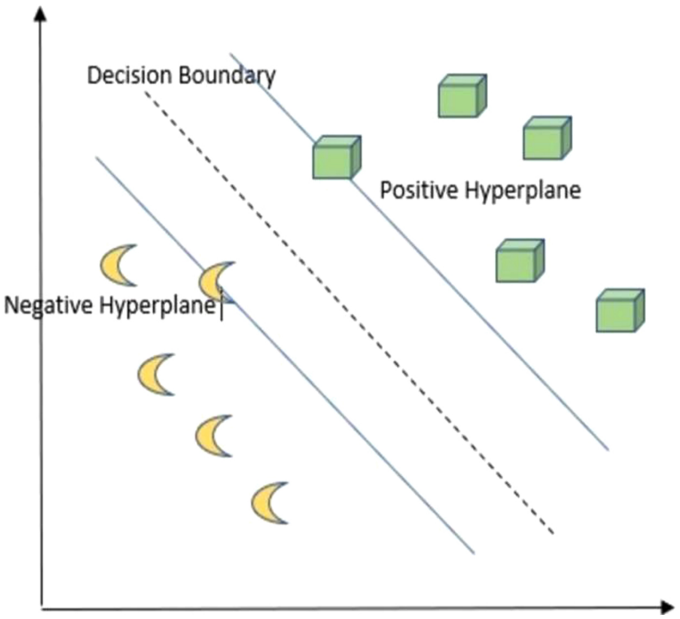

Fig. (1). The classification of shapes using hyperplane in SVM.

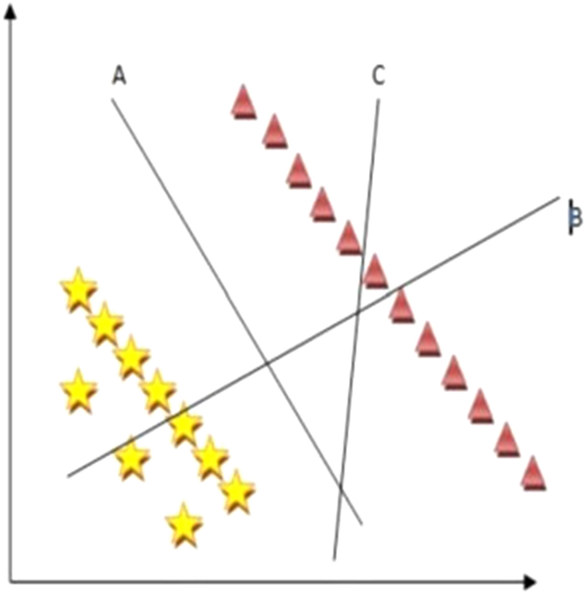

Fig. (2). Hyperplane A classifies the data points efficiently.

2. Use Margin and Large Margin Methods to find the appropriate hyperplane. The distance between the nearest data point and the hyperplane is referred to as the

margin. We must select the hyperplane that has a larger margin; this prevents a chance of miss classification. Consider Fig. (**3**). Here, the margin of hyperplane B is the largest as compared to the margin of hyperplanes A and C. So, we choose hyperplane B for classifying our data points.

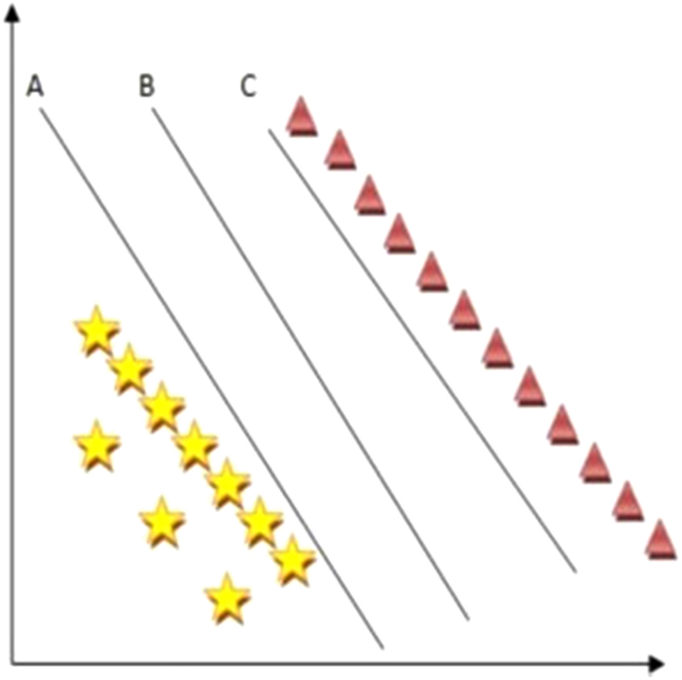

Fig. (3). Margin and Large Margin Methods.

3. Identify the correct hyperplane that can classify all the data points correctly without any error. Consider Fig. (**4**). Here, if we choose hyperplane B instead of hyperplane A, since hyperplane B has a larger margin as compared to hyperplane A, then hyperplane B doesn't classify all the data points correctly resulting in an error. Hyperplane A has a smaller margin but it classifies all the data points correctly. So, SVM will choose hyperplane A over hyperplane B.

4. SVM has the characteristic of ignoring the outliers or the noise. Consider Fig. (**5**). Here, the hyperplane chosen has a large margin and at the same time, the SVM ignores the star that lies in the other boundary and this star is treated as a noise or an outlier.

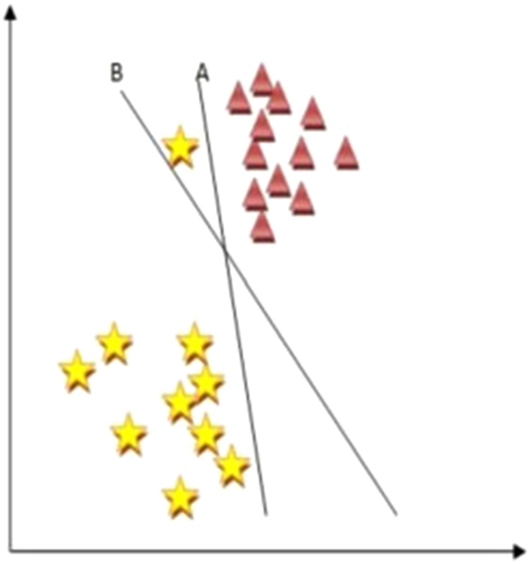

Fig. (4). Choosing hyperplane A over hyperplane B for classification.

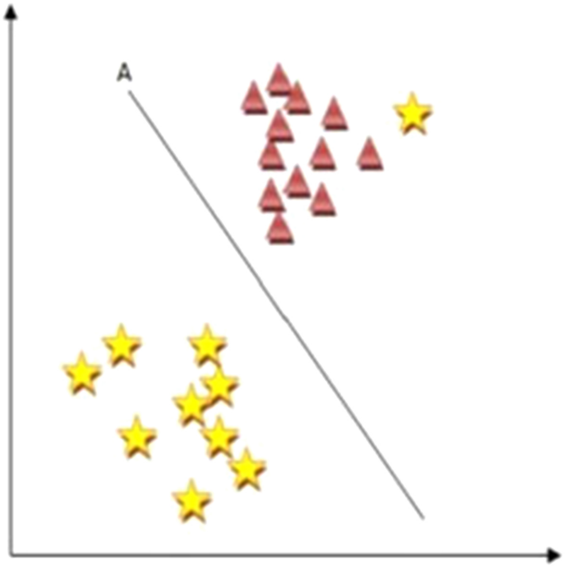

Fig. (5). Classification by SVM is robust to outliers.

The Maximal Margin Classifier

The Maximal margin classifier is defined when two classes are linearly separable.

The maximal margin classifier is a kind of hyperplane with the maximum margin. It is a hypothetical classifier that explains the implementation of SVM. The input

variables form an n-dimensional space. For *e.g.* a two dimensional space is formed by two input variables.

A hyperplane is said to be a line that separates input into two classes *i.e.* class 0 or class 1. In the two dimensional, a hyperplane may be thought of as a line that separates inputs into two classes.

E.g. A0+(A1*I1)+(A2*I2)=0

Here, A0 is the intercept. A1 and A2 are the slopes. I1 and I2 are the inputs. We can perform classification using this line. On the basis of the learning performed, we can determine whether the new input point would lie above or below this line or hyperplane.

If the point lies above the hyperplane, then it is class 0 and the value of equation is more than 0.

If the point lies below the hyperplane, then it is class 1 and the value of the equation returned is less than 0.

If a point lies close to the line, then it is difficult to define its class and the value of the equation returned is close to zero.

If there is a large value of magnitude, then the model is said to have more confidence in the prediction. Margin refers to the distance that lies between the nearest data points and the line. The line that has the largest margin is said to be optimal or the best line capable of separating two classes. It is also referred to as the Maximal-Margin hyperplane. Margin refers to the perpendicular distance that lies between the line and the nearest points. These nearest points help in defining lines and in constructing a classifier. These points are referred to as support vectors. The hyperplane learns through training data with the help of optimization methodology that causes the margin to be maximum.

Soft Margin Optimization

Unlike Hard Margin, a Soft Margin is able to tolerate a few dots. It tries to perform balancing between maximising a margin and minimising the classification.

Following are the two types of misclassification:

1. Dot lies on the wrong side of the decision boundary and on the correct side of the margin.

2. Dot lies on the wrong side of the decision boundary and on the wrong side on the margin.

Soft margin tries to maximise the margin and reduce the distance between misclassified points and the correct margin plane.

In Soft Margin Optimization, we introduce slack variables ξ

w =Σα yi xi. Here, w is defined in terms of the linear combination of x. Here, α is a Lagrange multiplier.

We can modify the constraints as follows:

yi(wxi+b)>=1-ξ; ξ>=0

Linear Programming Support Vector Machines

Support Vector Machine comprises two support hyperplanes that lie between the classes on the most extreme points. These are referred to as Support Vectors. We require to find the maximum distance between support hyperplanes. From here, we can obtain an Optimal hyperplane. It is also referred to as Maximum Margin Hyperplane. Margin is the distance between support hyperplanes.

Given an optimal hyperplane, wxi+b

Margin may be defined as: M=2/||w||. The maximum margin is defined as follows: max(2/||w||)

SUPPORT VECTOR REGRESSION

Support Vector Regression is a kind of regression algorithm that is able to support both non-linear as well as linear regression. Support Vector Regression is based on the methodology of Support Vector Machine. There is a difference between Support Vector Machine and Support Vector Regression. Support Vector Machine is a classifier that is used for the prediction of discrete categorical values, whereas Support Vector Regression is a kind of regressor that is capable of predicting continuous values. The difference between Simple regression and Support Vector regression is, in simple regression, we try to reduce the error rate and in support vector regression, we try to determine the best value within the margin referred as ε- tube.

Following are the steps involved in performing Support Vector Regression:

- The training dataset is collected.

- Selecting the kernel as well as the regularization and parameters.
- Correlation Matrix is created.
- Obtain the contraction coefficients by training the machine.
- With the help of coefficients; estimator is created.

Kernel Ridge Regression

Support Vector Machine kernel is a function that accepts low dimensional input space and converts it into a higher dimensional space. SVM performs the conversion of a non separable problem into a separable problem. It is generally used in non-linear separation problems. Consider the following Fig. (**12**). Here, we have a circular hyperplane that separates the stars and the triangles. We can convert the circular hyperplane into a linear hyperplane by introducing a new feature *z*, *z=x^2+y^2*. This is shown in Fig. (**13**).

Gaussian Processes

Gaussian Processes are a class of generic supervised learning methodology that is used to deal with probabilistic classification and regression problems. Gaussian Processes help in the interpolation of observation using prediction. The prediction in Gaussian Processes is Gaussian or probabilistic that helps in representing adaptive fitting or online fitting and also helps in the computation of empirical confidence. It comprises common or customised kernels. It is not sparse and it makes use of complete information during prediction. Its efficiency degrades when the number of features increases.

Gaussian process regressor helps in prediction without fitting priory. Gaussian Processes may be defined as a group of random variables and the subset of random variables is a multivariate Gaussian.

APPLICATIONS OF SUPPORT VECTOR MACHINE

Support Vector Machine is a very efficient and popular Machine Learning algorithm and is commonly used when dataset is smaller in size. Support Vector Machine is flexible to the addition of new data due to the presence of large margin. Support Vector Machine is a Supervised Machine Learning algorithm that is capable of performing the classification of unseen data [3, 4]. Other applications of Support Vector Machine include the following:

- - Face Recognition System
- - Image Classification
- - Categorising text and Hypertext
- - Bioinformatics

- - Protein Remote homology detection
- - Biometrics
- - Generalised Predictive Control

Some of the applications of Support Vector Machine is discussed below:

Text Categorisation

For transductive as well as inductive model, Support Vector Machine helps in the categorisation of hypertext and given text. It learns itself from the training data and performs the classification of documents into different classes such as web pages, news, e-mails, movies, business, *etc.* It also helps in the classification of different web pages according to whether they are personal home pages or other pages. In support vector machine, a score is computed for each document to be classified and the score is compared with a threshold value that is predefined. When the score of the threshold doesn't match the threshold, then it is classified in another category otherwise it is classified as in the general category. Other uses of Support Vector Machine include: Spam detection, Sentiment analysis of social media content, Named Entity Recognition, Machine Translation, Speech Recognition, Tagging Content online, *etc.*

Image Recognition

Support Vector Machine is a supervised machine learning algorithm and is widely used in biometrics *e.g.* face, iris, fingerprint, signature *etc.* recognition. A lot of research has been going on in the recognition of facial patterns when present along with noisy data.

Support Vector Machine also deals with sentiment analysis by analysing facial patterns.

Bioinformatics

Support Vector Machine is capable of giving useful results in computational biology field. It helps in the detection of gene structure, its function, its role in disease and interaction. It can deal with a large amount of different datasets and provide high accuracy.

PYTHON CODE

Consider the following code that depicts the data points that are classified correctly using SVM:

```
import numpy as np
import matplotlib.pyplot as plt from scipy import stats
import seaborn as sns; sns.set()
from sklearn.datasets.samples_generator import make_blobs
X,y=make_blobs(n_samples=100, centers= 2, random_state= 0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=100, cmap='autumn');
```

It gives the following output as shown in Fig. (**6**) ::

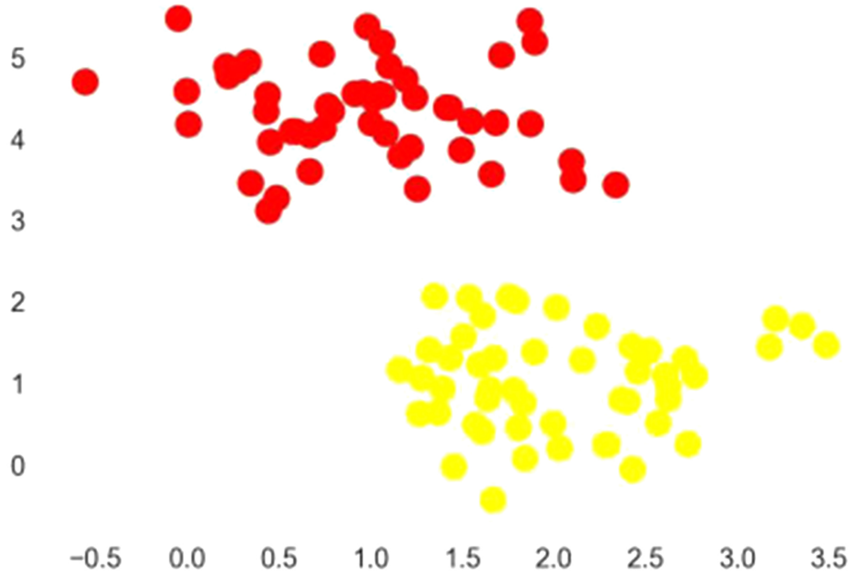

Fig. (6). Classification of data points using SVM.

Consider the following code that displays straight lines and separates or classifies the different sets of data. This is shown in Fig. (**7**) *:* :

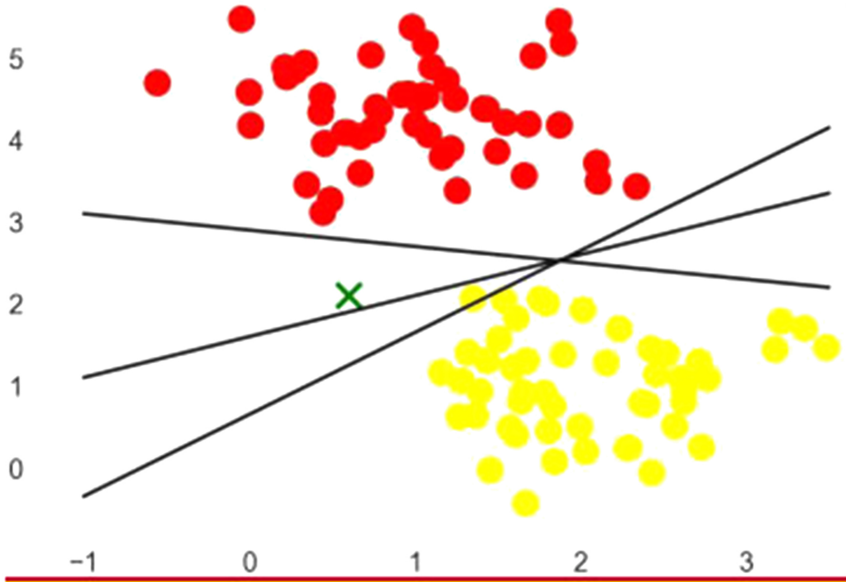

Fig. (7). Straight lines separating different sets of data.

```
fit = np.linspace(-1, 3.5)

plt.scatter(P[:, 0], P[:, 1], c=q, s=100, cmap='autumn')

plt.plot([0.6], [2.1], 'x', color='green', markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:

plt.plot(fit, m * fit + b, '-k')

plt.xlim(-1, 3.5);
```

Instead of drawing zero width straight lines, margins may be drawn around straight lines up to the nearest data point. This is shown in Fig. (**8**) :

```
fit = np.linspace(-1, 3.5)

plt.scatter(P[:, 0], P[:, 1], c=q, s=100, cmap='autumn')

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2,
```

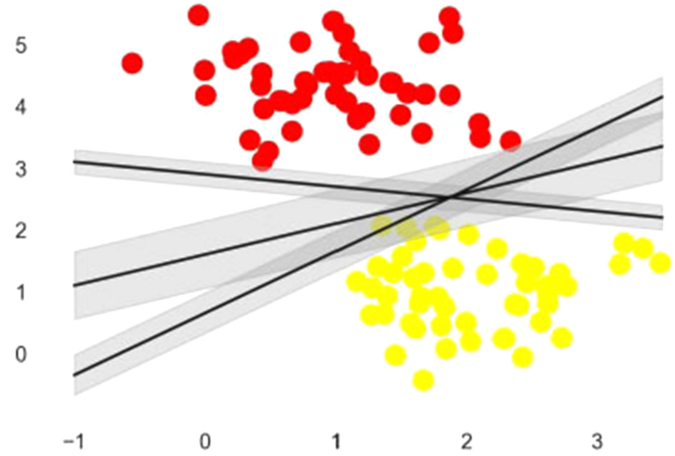

Fig. (8). Margins drawn around straight lines for classification in SVM.

2.9, 0.2)]:

fit2 = m * fit + b plt.plot(fit, fit2, '-k')

plt.fill_between(fit, fit2 - d, fit2 + d, edgecolor='none',

color='#AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);

Consider the following code using Scikit Learn that can be used to train the SVM model. This is depicted in Fig. (**9**) :

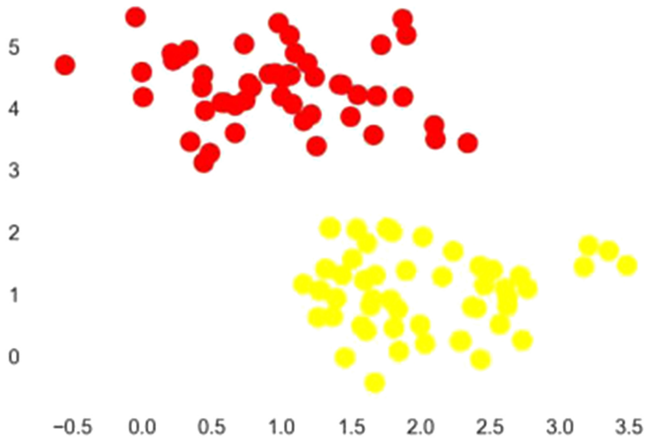

Fig. (9). Classification of the outcome using SVM

from sklearn.svm import SVC # “Support vector classifier” model = SVC(kernel='linear', C=1E10)

model.fit(P,q)

SVC (C=10000000000.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,decision_function_ shape='ovr',degree=3, gamma=' scale', kernel= ' linear',max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

def plot_svc_decision_function(model, px=None, plot_ support=True):

“””Decision function is plotted for a 2D SVC””” if px is None:

px = plt.gca() limx = px.get_xlim() limy = px.get_ylim()

To evaluate model, grid is created

p = np.linspace(limx[0], limx[1], 30)

q = np.linspace(limy[0], limy[1], 30) Q, P = np.meshgrid(q, p)

pq = np.vstack([P.ravel(), Q.ravel()]).T

P = model.decision_function(pq).reshape(P.shape)

Margins and decision boundary are plotted px.contour(P, Q, P, colors='k',

levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])

plot support vectors if plot_support:

px.scatter(model.support_vectors_[:, 0],

model.support_vectors_[:, 1],

s=400, linewidth=1, facecolors='none'); px.set_xlim(limx)

px.set_ylim(limy)

plt.scatter(P[:, 0], P[:, 1], c=q, s=100, cmap='autumn')

plot_svc_decision_function(model); model.support_vectors_ array([[0.44359863, 3.11530945],

[2.33812285, 3.43116792],

[2.06156753, 1.96918596])

Consider the following code that considers the first 50 and 130 points of the data set. This is shown in Fig. (**10**) :

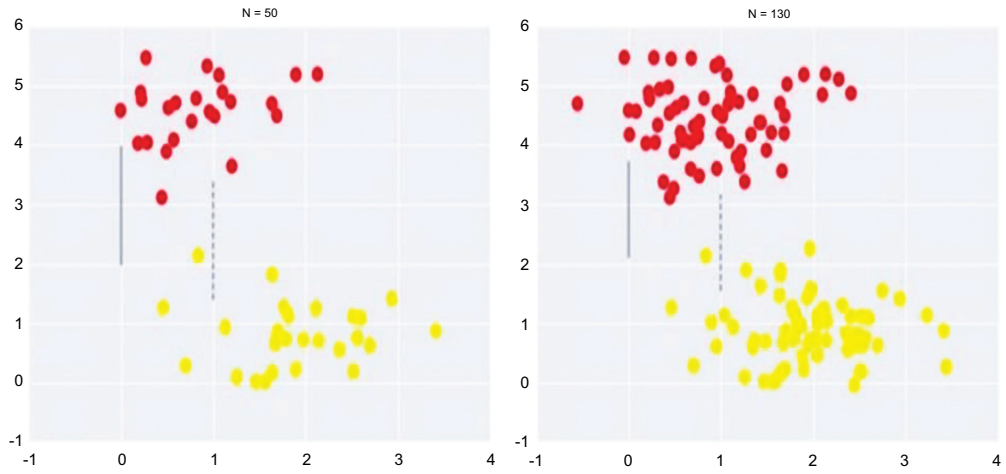

Fig. (10). Classification of the first 50 and 130 data points using SVM.

def plot_svm(N=10, ax=None):

P, q = make_blobs(n_samples=200, centers=2,

random_state=0, cluster_std=0.60)

P = P[:N]

q = q[:N]

model = SVC(kernel='linear', C=1E10) model.fit(X, q)

px = px or plt.gca()

px.scatter(X[:, 0], X[:, 1], c=q, s=100, cmap='autumn') px.set_xlim(-1, 4)

px.set_ylim(-1, 6) plot_svc_decision_function(model, px)

fig, px = plt.subplots(1, 2, figsize=(16, 6))

fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1) for pxi, N in zip(px, [50, 130]):

```
plot_svm(N, pxi)

pxi.set_title('N = {0}'.format(N))
```

Consider the following code that shows the classification of data set which is not linearly separable. This is shown in Fig. (**11**) :

```
from sklearn.datasets.samples_generator import make_circles P, q = make_circles(200, factor=.1, noise=.1)

clf1 = SVC(kernel='linear').fit(P, q)

plt.scatter(P[:, 0], P[:, 1], c=q, s=50, cmap='autumn') plot_svc_decision_function (clf1, plot_support=False);
```

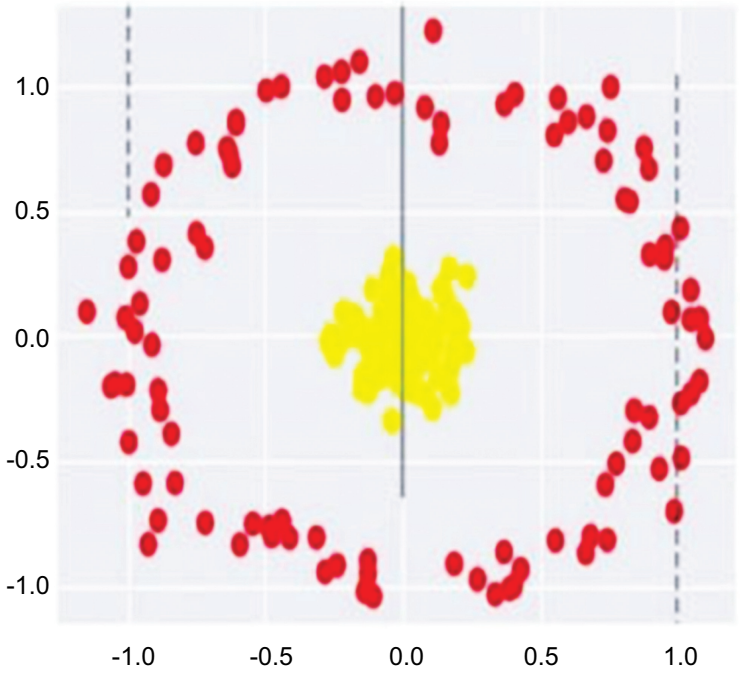

Fig. (11). Classification of data which is not linearly separable.

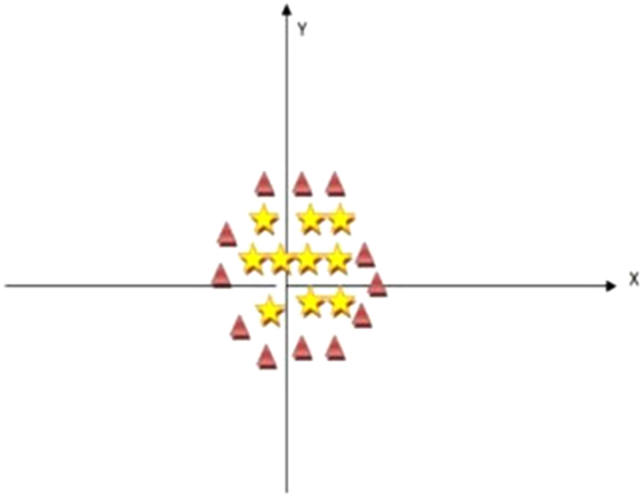

Fig. (12). Circular hyperplane separating the stars and the triangles.

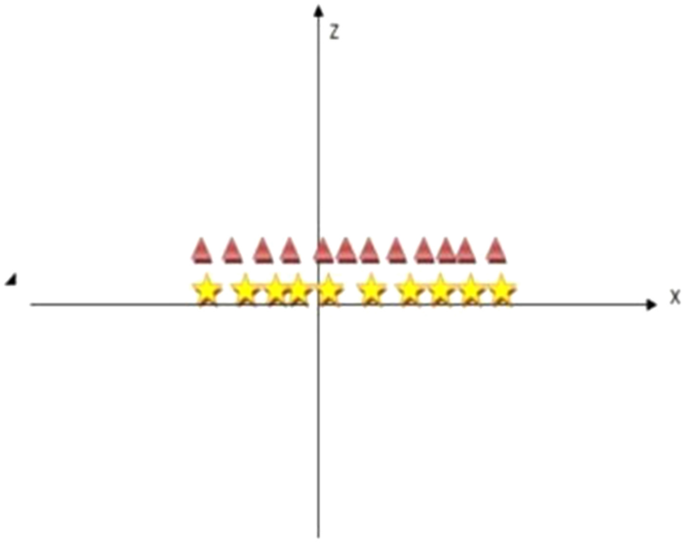

Fig. (13). Linear hyperplane separating the stars and the triangles.

CONCLUSION

Support Vector Machine is a supervised machine learning algorithm that has numerous applications in the field of text classification, document categorisation, sentiment analysis, image recognition, bioinformatics, *etc.* In this chapter, we have discussed about Support Vector Machine in detail, soft margin optimization, Gaussian processes, concepts of margin, hyperplane, *etc.*

EXERCISES

1. Explain soft margin optimization.

2. Define Margin and Hyperplane.

3. Explain Gaussian Processes.

4. Explain the use of a Support Vector Machine in Text classification.

REFERENCES

[1] W.S. Noble, "What is a support vector machine?", *Nat. Biotechnol.,* vol. 24, no. 12, pp. 1565-1567, 2006.
[http://dx.doi.org/10.1038/nbt1206-1565] [PMID: 17160063]

[2] A. Widodo, and B.S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis", *Mech. Syst. Signal Process.,* vol. 21, no. 6, pp. 2560-2574, 2007.
[http://dx.doi.org/10.1016/j.ymssp.2006.12.007]

[3] L. Zhang, W. Zhou, and L. Jiao, "Wavelet support vector machine", *IEEE Trans. Syst. Man Cybern. B Cybern.,* vol. 34, no. 1, pp. 34-39, 2004.
[http://dx.doi.org/10.1109/TSMCB.2003.811113] [PMID: 15369048]

[4] A. Pradhan, "Support vector machine-a survey", *Int. J. Emerg. Technol. Adv. Eng.,* vol. 2, no. 8, pp. 82-85, 2012.

CHAPTER 5

Decision Trees

Abstract: Neural networks are a way to mimic the working of a human brain. Decision trees refer to the decision support structure that uses a tree to make decisions and draw all possible consequences. Decision trees are a way to display conditional control statements. In the following chapter, we will discuss decision trees, regression trees, stopping criterion and pruning loss functions in a decision tree, categorical attributes, multiway splits and missing values in decision trees, and instability in decision trees.

Keywords: Decision trees, Instability in decision trees, Multiway splits, Regression trees.

INTRODUCTION

Decision trees refer to the non-parametric method of supervised learning. Decision trees are used for the purpose of regression and classification.

In a Decision Tree algorithm, a class of a given dataset is predicted. We begin from the root node and follow the branch according to the condition or the property satisfied by the corresponding dataset.

Pruning in a Decision Tree involves the elimination of unwanted branches of a tree which will never contribute to the resultant path.

Features of Decision Trees include the following:

- It is referred to as a Supervised Machine Learning Algorithm which is non-parametric in nature.
- It is capable of representing both regression and classification tasks.
- A decision tree is represented by a hierarchical tree-like structure having components such as root node, internal node and leaf nodes.
- Decision tree employs greedy search and divide and conquer approach to solve the problem.
- For building a given tree, the CART algorithm is used. CART refers to the Classification and Regression Tree algorithm.

The advantages of decision trees include the following:

- It is simple and easy to understand.
- It is helpful in solving decision related problems efficiently.
- It provides all possible different kinds of results from a given problem.
- Compared to other Machine Learning algorithms; it requires less data cleaning.

REGRESSION TREES

Decision tree algorithms may be used for the process of the prediction of results based on the given data. Decision tree algorithms are of two types, namely: classification tree and regression tree algorithm. **The classification and Regression Tree** (**CART**) methodology came into existence in 1984. It was introduced by *Leo Breiman*, *Jerome Friedman*, *Richard Olshen*, and *Charles Stone*. In a classification tree algorithm, the outcome variable is categorical or fixed. For example, using the classification tree algorithm, we may decide what type of car a customer will purchase. In the regression tree algorithm, the target outcome value is a real number. For example, the selling price of residential places may be predicted using the regression tree algorithm. In the classification tree algorithm, the data set is split into classes such as Yes or No. In the regression tree algorithm, the target variable is continuous, for example, temperature, price, *etc.*

STOPPING CRITERION AND PRUNING LOSS FUNCTIONS IN DECISION TREE

The pruning technique is associated with the decision trees that can perform a reduction in the size of the decision trees by eliminating the parts of the tree that do not classify instances. Overfitting can be prevented by including pruning along with the decision trees. Overfitting occurs when the training is done so thoroughly that it also learns noise along with the pattern. Under-fitting occurs when the amount of training is so insufficient that all the patterns cannot be identified. Pruning means that the tree is cut back. *Quinlan* in 1987 suggested a simple method for pruning decision trees, referred to as reduced error pruning. In reduced error pruning, internal nodes are traversed from the bottom to the top and pruned only if it doesn't reduce the tree's accuracy. *Olaru* and *Wehenkel* in 2003 suggested the use of minimum error pruning. In minimum error pruning, for every node we perform a comparison of 1-probability error rate estimation without and with pruning. Pessimistic pruning is a fast method of pruning in which the nodes are traversed in a top to down manner. If a given internal node is pruned then all descendants of this internal node are not sent for the pruning process. Optimal pruning is used to guarantee optimality and is based on the concept of dynamic

programming. In optimal pruning, the tree obtained after pruning is much smaller as compared to the original tree and the number of internal nodes is much smaller as compared to the number of leaves.

CATEGORICAL ATTRIBUTES, MULTIWAY SPLITS AND MISSING VALUES IN DECISION TREES

CART refers to the decision tree algorithm that either generates binary regression or classification trees based on whether the target variable is numeric or categorical. Optimal partitioning must be followed while performing the partitioning of decision trees. In CART, the same variables may be reused in the different parts of decision trees. Splitting may be binary or multiway.

In a binary splitting, each node is further divided into at most two subgroups, whereas in the case of multiway splitting, each node is further split into multiple subgroups. Decision trees are very easy to comprehend if we follow multiway splitting as a particular attribute rarely reappears while traversing a path from the root to the leaf.

There are several methods used for dealing with missing values in decision trees. Missing values may be ignored or may be assigned some other category.

Missing values instances may be distributed among the child nodes

as follows:

1. Everything goes to a node having the largest number of instances.

2. Distribution is done among all child nodes but with minimum weights, which is proportional to the number of instances from every child node.

3. Distribution is done randomly according to the categorical distribution of a single child node.

4. Sort, build, and use input features that decide how the distribution of instances is done in a child node.

ISSUES IN DECISION TREE LEARNING

Preventing Overfitting of Data

Decision Trees is a supervised machine-learning approach used for performing regression and classification [1]. Overfitting occurs when the model trains data along with noise and fails to collect the important patterns. A decision tree in an

overfitting situation performs well using the training data but it shows poor performance for the unseen data or testing data. An overfitting situation arises when a decision tree is grown into full depth. Pruning is one of the techniques to prevent overfitting. Pruning prevents the growing of decision trees to their complete depth and hence prevents overfitting to take place.

The two different types of Pruning techniques are: pre-pruning and post-pruning. Pre-pruning technique is the early prevention of further growth of a decision tree by fine-tuning the hyperparameters such as min_samples_split, min_samples_leaf, max_depth *etc.* Post pruning technique is a technique for removing overfitting condition in decision trees by initially allowing the decision tree to grow to full depth and then eliminating its branches to prevent overfitting to take place. One of the techniques of post-pruning is cost complexity pruning. In cost complexity pruning, tuning of ccp_alpha is performed to obtain the best fit model. In Scikit-learn, the function cost_complexity_pruning_path() is used to calculate ccp_alpha values of a given decision tree.

Incorporating Continuous Valued Attributes

We can manage continuous valued attributes in a decision tree either by using a comparison operator (=>,<=) or creating finite range sets [2]. In this way, we can obtain a decision tree with a binary split and decision tree with multiway split respectively. A decision tree with a binary split is obtained using comparison-based test conditions. A decision tree with multiway split or categorical attribute is obtained using a finite set of range buckets [3]. The categorical data or continuous data or multiway split may be obtained by using the equal width approach.

Different categories of equal widths are obtained in a continuous valued attribute.

E.g. for attributes in the range 0-100; we have 5 categories mainly [(0-20),(2--40),(40-60),(60-80),(80-100)].

In the equal frequency technique, the same number of data points are present in each category. Outliers are sensitive to this kind of approach. Unsupervised clustering algorithms may be used for defining different categories.

We can also obtain binary attributes or two way splits from a given continuous valued attribute. A comparison based test such as attribute>=value can help in obtaining binary attributes [4]. In case of numbers; we can perform sorting and split the array from the middle to obtain two classes.

Other Measures for Attributes Selection

In a given decision tree, it is difficult to decide which attribute should be placed at the root or at the internal nodes of a tree. By randomly selecting a node and considering it as a root node will give poor results. There are some criteria that can help in attribute selection. These include the following:

- Gini index
- Gain ratio
- Entropy
- Chi Square
- Information Gain
- Reduction in Variance

Gini index is a kind of cost function that helps in the evaluation of splits in a given dataset. Gini index is computed by subtracting the sum of squared probabilities of individual classes from one. Large partitions are favoured more. Smaller partitions are favoured by information gain. Gini index performs two-way split or binary split. It generates two categories of datasets mainly success or failure. Higher heterogeneity and inequality are the result of high mini index value. With the help of success(p) and failure(q); gini is calculated as p2+q2. Compute Gini index with the help of Gini score which is computed for every node in the split. Classification and Regression Tree (CART) make use of Gini index for generating split points. Here, the value of all attributes is computed, then sorting of all the values is performed. The attribute having the highest value is made at the root. For Gini index, attributes are considered to be continuous. If attributes are categorical, then the criterion used is information gain.

Entropy refers to the randomness in the information that is to be processed. If entropy is higher, then it is difficult to draw useful results. According to ID3, if the entropy of a branch is zero, then it is said to be a leaf node. If the entropy of a branch is more than zero, then the node is split further.

Information Gain (IG) is a property that measures the way a given attribute classifies the training examples. Information Gain is calculated as the difference in entropy before node splitting and taking an average of entropy after node splitting. Mathematically, Information Gain is defined as follows:

Information Gain(P,Q)=Entropy(P)-Entropy(P,Q)

The gain ratio is the ratio of Information gain and split information. Reduction in variance is an algorithm used for the splitting of the population. The node with the

less value of variance is chosen for further split. Chi-squared Automatic Interaction Detector (CHAID) is a classification method in decision trees. It is calculated as the sum of squares of differences between observed and expected frequencies of target variable. If the Chi-Square value is high, then there is a significant difference in the statistical values of child and parent node. The tree generated is referred to as CHAID.

Handling Missing Values

There are several methods used by various decision trees. Simply ignoring the missing values (like ID3 and other old algorithms does) or treating the missing values as another category (in case of a nominal feature) are not real handling missing values. However, those approaches were used in the early stages of decision tree development.

The real handling approaches to missing data does not use data point with missing values in the evaluation of a split. However, when child nodes are created and trained, those instances are distributed somehow.

The following approaches are used to distribute the missing value instances to child nodes:

- All goes to the node which already has the biggest number of instances.
- Distribute to all children, but with diminished weights, proportional to the number of instances from each child node.
- Distribute randomly to only one single child node, eventually according to a categorical distribution.
- Build, sort and use surrogates to distribute instances to a child node, where surrogates are input features which resemble best how the test feature sends data instances to the left or right child node.

Handling of Attributes with Differing Costs

There may be a cost associated with the attributes in the learning tasks. For *e.g.* in a given hospital, where people are treated for COVID, there may be an associated cost in the attributes type of room, test type, *etc.* that will be according to patient's comfort. In this case, we would design a decision tree having both low cost attributes and high cost attributes.

We can modify ID3 in which the cost term may be included along with the attributes. Low cost attribute will be considered if the gain is divided by cost. These cost-sensitive attributes do not result in optimal cost-sensitive decision tree; these generally favour the attributes having low cost.

INSTABILITY IN DECISION TREES

The instability problem in a decision tree classifier means the resulting constructed rules might be different from the original or the actual ones if there is a modification in the training sample. This instability is caused due to invalid selection of the split candidate. The split candidate is found using a split evaluation function that can be used to partition the data. If at a particular stage, no dominant split is found, then the split candidate is used to partition the node. If a different split is selected at any stage, then it results in a tree which is absolutely different from the original tree. So, the selection of the correct splitting candidate is very important in order to prevent inaccuracy or instability in decision trees.

PYTHON CODE

1. Consider the following code on decision trees using scikit learn. Here, DecisionTreeClassifier is a class that can perform multi- class classification on a given data set. DecisionTreeClassifier takes two arrays as input, namely P and Q.

```
from sklearn import tree

P = [[0, 0], [1, 1]]

Q = [0, 1]

DTclf = tree.DecisionTreeClassifier() DTclf = DTclf.fit(X, Y) DTclf.predict([[2., 2.]]) array([1])

DTclf.predict_proba([[2., 2.]])

array([[0., 1.]])

from sklearn.datasets import load_iris from sklearn import tree

P, q = load_iris(return_X_y=True) DTclf = tree.DecisionTreeClassifier() DTclf = DTclf.fit(P, q) tree.plot_tree(DTclf)

from sklearn.datasets import load_iris

from sklearn.tree import DecisionTreeClassifier from sklearn.tree import export_text

iris = load_iris()

decisiontree = DecisionTreeClassifier(random_state=0, max_ depth=2)
```

```
decisiontree = decisiontree.fit(iris.data, iris.target)

er = export_text(decisiontree, feature_names=iris['feature_ names'])

print(er)

from sklearn import tree

X = [[0, 0], [2, 2]]

y = [0.5, 2.5]

DTclf = tree.DecisionTreeRegressor() DTclf = DTclf.fit(X, y) DTclf.predict([[1, 1]]) array([0.5])
```

In the above code, DTclf.predict() is used for the prediction of the samples of a class. DTclf.predict_proba() is used to find the probability of each class. DecisionTreeClassifier can be used for binary classification and multi-class classification. plot_tree function can be used to plot the tree.

CONCLUSION

In this chapter, we learned about decision trees, the difference between regression trees and classification trees, stopping criterion and pruning loss functions in decision trees, categorical attributes, multiway splits, missing values in decision trees, and instability in decision trees. In the next chapter, we will discuss supervised learning, statistical decision theory, the nearest neighbour method, *etc.*

EXERCISES

1. Explain the difference between classification trees and regression trees.

2. Explain the stopping criterion and pruning loss functions in decision trees.

3. What are multiway splits in decision trees?

4. Explain instability in decision trees.

REFERENCES

[1] S.B. Kotsiantis, "Decision trees: a recent overview", *Artif. Intell. Rev.,* vol. 39, no. 4, pp. 261-283, 2013.
[http://dx.doi.org/10.1007/s10462-011-9272-4]

[2] L. Rokach, and O. Maimon, "Decision trees", In: *Data mining and knowledge discovery handbook.* Springer: Boston, MA, 2005, pp. 165-192.
[http://dx.doi.org/10.1007/0-387-25465-X_9]

[3] V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman, "Decision trees: an overview and their use in

medicine", *J. Med. Syst.,* vol. 26, no. 5, pp. 445-463, 2002.
[http://dx.doi.org/10.1023/A:1016409317640] [PMID: 12182209]

[4] J.R. Quinlan, "Simplifying decision trees", *Int. J. Man Mach. Stud.,* vol. 27, no. 3, pp. 221-234, 1987.
[http://dx.doi.org/10.1016/S0020-7373(87)80053-6]

CHAPTER 6

Neural Network

Abstract: A Support Vector Machine is used for the purpose of classification and regression. In the previous chapter, we discussed Support Vector Machine, margin and large margin methods, and kernel methods. A neural network refers to a parallel computing device that attempts to mimic the model of the brain. In the following chapter, we will discuss the early models of Neural Networks as well as the Perceptron learning model, back propagation, and Stochastic Gradient Descent.

Keywords: Artificial neural network, Backpropagation, Data science, Machine learning,, Neural network, Perceptron learning, Python.

INTRODUCTION

Neural network is a means of performing a machine learning task, in which a computer learns by the analysis of training examples. Following are the objectives covered in this chapter:

- To know early models and perceptron learning.
- Understanding back propagation.
- Understanding stochastic gradient descent.

EARLY MODELS

Neural network is one of the subfields of machine learning. Neural network accepts the input data, performs training on the data, and produces the output based on the training performed [1]. In 1943, *Warren Mc Culloch* and *Walter Pits* described the working of neurons. They modeled neural networks with the help of electrical circuits in order to explain the working of neurons in the brain. In 1949, *Donald Hebb* wrote *The Organization of Behavior*, which pointed that the connection between the two neurons is enhanced if they are fired together. In 1959, *Bernard Widrow* and *Marcian Hoff* at Stanford developed neural network based models called **'ADALINE'** and **'MADALINE'. ADALINE** stands for **Adaptive Linear Elements** and **MADALINE** stands for **Multiple Adaptive Linear Elements**. ADALINE was used for the recognition of binary patterns. For a given stream of bits, it can predict the occurrence of the next bit. MADALINE

stands for Multiple Adaptive Linear Elements. It is the first neural network that is applied to real-world problems and is still in use for commercial purposes.

PERCEPTRON LEARNING

Perceptron is based on a neutron, which is the basic processing unit of the brain. A neuron comprises dendrites, a cell body, and an axon. Signal flows from the axon to the dendrites. An action signal is fired by a neuron when a particular threshold is met by a cell. This action either takes place or it does not. There is no concept of partial firing by a neuron. Consider Fig. (**1**) depicting the Perceptron model:

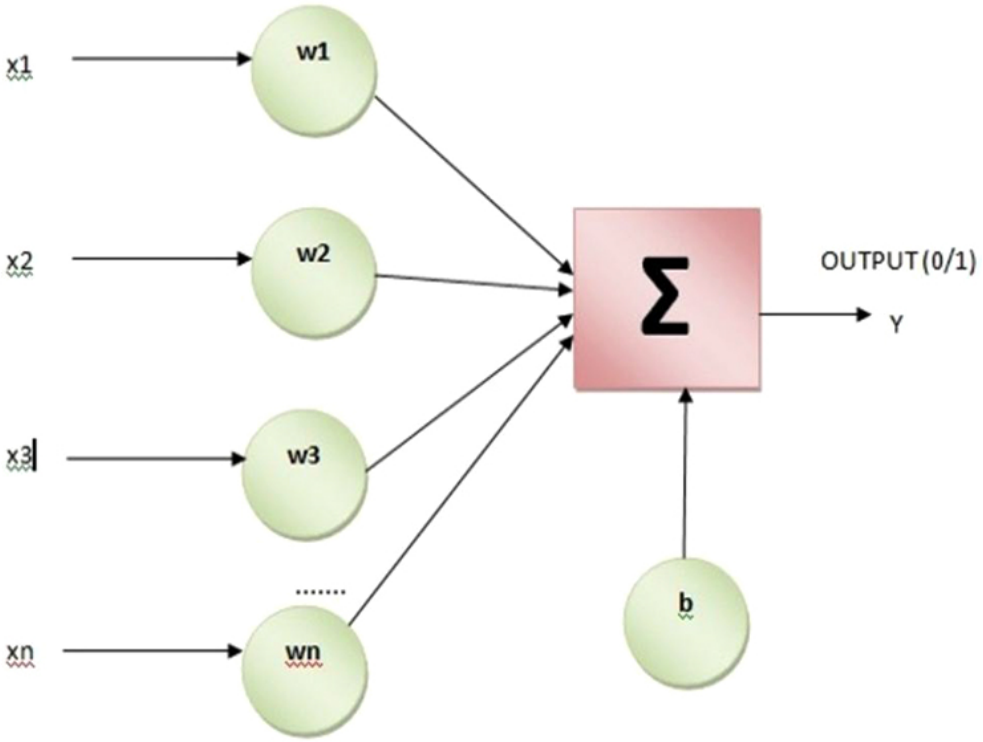

Fig. (1). Perceptron model.

Perceptron can be used for solving binary classification problems where the sample that needs to be identified belongs to two classes. Many features or inputs are sent to the linear unit of a Perceptron and it generates one binary output.

So, a single neuron neural network is referred to as a Perceptron. It accepts the input and the weight, performs the weighted sum of inputs, and applies an activation function over it. It accepts and generates only binary values. One of the

limitations of the Perceptron learning model is that it can solve only linearly separable problems.

BACKPROPAGATION

Back propagation is also referred to as Gradient Computation. Back propagation learning algorithm comprises two phases, namely: Gradient Computation Phase and Weight Updation Phase. The first phase is the Propagation phase. It involves the following steps:

Forward Propagation - Here, the training input pattern is sent to the neural network and the propagation's output activation is generated.

Backward Propagation – Here, the input is the propagation's output activation that is sent to the neural network, and it generates the deltas of all the output and hidden neurons.

The second phase is the Weight Updation phase. It involves the following steps:

1. The gradient of the weight is calculated by multiplying the output delta and the input activation.

2. A ratio or percentage of the gradient is subtracted from the weight. This ratio or percentage affects the quality of learning and speed. It is referred to as the learning rate. A neuron is able to train faster if the learning rate is higher. If the learning rate is lower, then the training is considered accurate.

AN ILLUSTRATIVE EXAMPLE: FACE RECOGNITION

Facial Recognition is an area of research in the field of pattern recognition and image processing. Face recognition systems are very important and are in great demand in commercial as well as law enforcement applications. A lot of work has already been done in the field of face recognition systems but still research is going on in this field to overcome the challenges that are faced in the face recognition system. One of the challenges faced in face recognition systems is an increase in the database size increases the time of recognition. Other issues in recognition include: changes in scene illumination, pose changes, updations in expressions, orientation, *etc.*

Face Recognition application is important in the field of Computer Vision, Image Processing, Psychology, Security, multimedia, *etc.* In Face Recognition System, a person’s face is captured using a camera. Facial recognition analyzes the characteristics of a person's face images input through a digital video camera or online face capturing. The first step in Face Recognition System is to collect the

image using a camera, database or a video. This image is sent for further processing in the Face Recognition System. After the collection of an image, the face in the image is detected. After the detection of face, data preprocessing task is performed. In the data preprocessing step, issues like blur, unwanted noise, differing lightening conditions, shadow effect, *etc.* may be eliminated. After data preprocessing task, feature extraction task is performed. In the feature extraction task, feature extraction algorithm is used for the extraction of features of the face. Extraction task is performed for noise removal, information extraction, and dimension reduction. After extraction task, analysis is performed in order to obtain an automatic face recognition system. After extracting the features, these features are compared with already stored features in the database and a useful conclusion is drawn. There are two applications of face recognition system including identification and verification. Face identification is used for detecting a person's identity using his facial features. In face verification, the system must be capable of authenticating that the face belongs to a particular person. There are two approaches of Face recognition namely-local feature-based method and the global feature-based method. Human faces may be represented using local or global features. Today face recognition is performed with the help of multiple-views of faces. These multiple views include left, right, top and bottom views of the face. There are many machine learning approaches used for building face recognition systems. One such approach is Neural Network. Neural Network is used for the prediction of known as well as unknown data. Neural Network is applied on linear and non-linear separable datasets. Neural Network has applications in other fields also such as speech recognition, iris recognition, face recognition and fingerprint recognition.

STOCHASTIC GRADIENT DESCENT

Stochastic gradient descent is referred to as an iterative method that can be used for the optimization of an objective function with the necessary properties (sub-differentiable or differentiable). Stochastic gradient descent may be used to minimize the computations to be performed. It randomly selects a data point at a given iteration that reduces the computations enormously.

Difference between Stochastic Gradient Descent and Gradient Descent:

- Gradient Descent is more expensive in terms of computations as compared to Stochastic Gradient Descent.
- For the purpose of optimization of learning algorithms, the Stochastic Gradient Descent algorithm is preferred over the traditional Gradient Descent algorithm.
- Stochastic Gradient Descent requires more iterations to achieve minima as compared to Gradient Descent Algorithm.

ADVANCED TOPICS IN ARTIFICIAL NEURAL NETWORK

Alternative Error Functions

Other error functions are also used for penalising large weights. The values as well as target slopes can be trained together. *e.g.* in phoneme recognition to minimise the error. We wish to minimise the error and this is achieved using an error function.

The most commonly used error function is the square of difference between the expected outcome and the actual outcome. Using the optimization process, a Neural Network model is trained and a loss function is used to compute the error [2, 3]. When training a given machine learning model, a loss function that computes the error may be found using maximum likelihood. When we are training a neural model, two types of loss functions are used; namely: Mean Squared Error and Cross Entropy.

Alternative Error Minimization Mechanism

The training of a Neural Network is performed using stochastic gradient descent and a loss function is used to design a Neural network in order to minimise the error occurred. An optimisation process is used to train a given neural network that involves the computation of model error by the loss function. A maximum likelihood is a kind of framework used in training various machine learning models for selecting a suitable loss function.

A deep learning neural network learns to map a set of inputs to a set of outputs from training data. In a Neural Network, we need to compute the perfect value of weights with the help of optimization algorithm in order to make best predictions. The training of a Neural Network is performed using a stochastic gradient descent optimization algorithm and updation of weights is performed using back propagation of error algorithm. The gradient in stochastic gradient descent is referred to as an error gradient. A neural network having weights is used for making predictions and the error using these predictions is computed. The gradient descent algorithm acts as an optimisation algorithm and makes changes in the weights so that the error is reduced and there is a navigation in downward direction of slope (gradient) . In a given optimization algorithm, an objective function is a function that is used for computing weights or candidate solution. In a Neural Network, we would like to minimise the given error. The objective function is also known as a loss function or cost function. A loss function computes a value which is referred to as loss. The objective function is the function that we would like to maximize or minimise. It is also referred to as criterion. When we try to minimise the objective function or criterion then it is

referred to as an error function, loss function or cost function. In a maximum likelihood, the weights in a neural network are estimated. In maximum likelihood, as the training includes more and more examples, the estimation of weights or model parameters also improvises. This property of maximum likelihood is referred to as consistency.

Recurrent Networks

A Recurrent Neural Network (RNN) is a kind of neural network in which the output of one Neural Network is sent as an input to the next Neural Network. In a traditional Neural Network, the output and input of a given Neural Network Model are independent and there lies no relationship between them . But, in case of Neural Networks, the output of a given Neural Network depends on the previous output of the same or other neural networks. RNN is characterised by a hidden state and it has unique characteristics to memorise the state sequence.

A recurrent neural network is a kind of artificial neural network that can be used to build an undirected or a directed graph by connecting adjacent nodes. RNN makes use of its memory or internal state for processing its input sequence. RNN has its application in forensic sciences, speech recognition, handwriting recognition, *etc*. RNNs are referred to as Turing complete. Recurrent neural network is an infinite impulse response network which is a directed cyclic graph and unrolling cannot be performed. A convolutional neural network is a finite impulse response network which is a directed acyclic graph and unrolling can be performed [3]. Storage states may be present in both finite impulse response network and infinite impulse response network. Instead of this storage, we can use a neural network also. This state is also referred to as gated memory or gated state and it is part of LSTM (Long Short-term Memory Networks). It is also referred to as FNN(Feedback Neural Network) [4].

Dynamically Modifying Network Structures

Dynamic Neural Networks give better and improved performance over Static Neural Networks. In Dynamic Neural Network, we add many decision algorithms that make learning automatically with any kind of input.

Dynamic Neural networks can be considered as the improvement of static neural networks by adding more decision algorithms we can make neural networks learn dynamically for the input. These networks have the capability to take input and learn from the environment. These networks are adaptive according to the given situation. Dynamic Neural Networks are more efficient and can perform allocation of computation energy according to the size of the input. These models are flexible and comprise input dependent architecture. These models have the

capability of achieving different efficiency and accuracy according to the running time and the computational cost. Some of the features of these networks include: Improved algorithm optimization, improved data preprocessing, improved architecture design optimization, Network pruning, low-rank optimization *etc.* These networks are also referred to as data dependent neural networks that process the given input like a human brain. It observes the part of that input which is useful, processes it and discards the input which is not useful. These models have applications in image classification, image segmentation, *etc.*

PYTHON CODES

1. Consider the following code on Perceptron learning. In this code, we have added 1 to the input_ size in order to include bias in the weight vector:

```
import numpy as np

class Perceptronlearning(object):

“””Implementation of a perceptron learning network”””

def

init (self, input_size): self.W = np.zeros(input_size+1)

def activation_fn(self, p):

“””” 1 is returned if p>=0 otherwise it returns 0””””” return 1 if p >= 0 else 0

“””” Prediction is a process of sending an input to the perceptron and returning an output. Bias is added to the input vector. We can compute the inner product, and the activation function is applied ”””””

def predict(self, p):

p = np.insert(p, 0, 1)

q = self.W.T.dot(p)

r = self.activation_fn(q) return r

def

init (self, input_size, lr=1, epochs=10): self.W = np.zeros(input_size+1)

# add one for bias self.epochs = epochs self.lr = lr
```

```
def fit(self, P, d):
for _ in range(self.epochs): for i in range(d.shape[0]):
y = self.predict(P[i]) e = d[i] - y
self.W = self.W + self.lr * e * np.insert(P[i],
0, 1)
class Perceptronlearning(object): “””Implements a perceptron network”””
def
init (self, input_size, lr=1, epochs=100): self.W = np.zeros(input_size+1)
# add one for bias self.epochs = epochs self.lr = lr
def activation_fn(self, p):
#return (p >= 0).astype(np.float32) return 1 if p >= 0 else 0
def predict(self, p):
q = self.W.T.dot(p)
r = self.activation_fn(q) return r
def fit(self, P, d):
for _ in range(self.epochs): for i in range(d.shape[0]):
p = np.insert(P[i], 0, 1) y = self.predict(p)
e = d[i] - y
self.W = self.W + self.lr * e * p
if name == ' main ':
P = np.array([
])
d = np.array([1, 1, 0, 1])
perceptron = Perceptronlearning(input_size=2) perceptron.fit(P, d)
```

print(perceptron.W) [0. -1. 2.]

2. Consider the following code using stochastic gradient descent. Here, we take 2 arrays, P and Q. Array P comprises the training samples. Array Q holds the target values.

import numpy as np

from sklearn import linear_model

P = np.array([[-1, 2], [1, -1], [2, 1], [2, 2]])

Q = np.array([1, 2, 1, 1])

```
[0,    0],
[0,    1],
[1,    0],
[1,    1]
```

SGDClassif = linear_model.SGDClassifier(max_iter = 1000, tol=1e-3,penalty = “elasticnet”)

SGDClassif.fit(P, Q)

SGDClassifier(alpha=0.0001, average=False, class_weight= None, early_ stopping= False, epsilon=0.1, eta0=0.0,fit_intercept=True,l1_ratio=0.15, learning_rate='optimal',

loss='hinge',max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty= 'elasticnet', power_t=0.5, random_state=None,shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0,warm_start=False)

SGDClassif.predict([[3.,3.]]) array([1])

SGDClassif.coef_array([[9.77200712, -19.54811198]])

SGDClassif.intercept_ array([-10.])

SGDClassif.decision_function([[3., 3.]]) array([-39.32831456])

import numpy as np

from sklearn import linear_model

nsamples, nfeatures = 9, 4

rng = np.random.RandomState(0)

q = rng.randn(nsamples)

p = rng.randn(nsamples, nfeatures)

SGDReg =linear_model.SGDRegressor (max_iter = 1000, penalty = “elasticnet”,loss = 'huber',tol = 1e-3, average = True

)

SGDReg.fit(p, q)

SGDRegressor(alpha=0.0001, average=True, early_ stopping=False, epsilon=0.1,

eta0=0.01, fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling', loss='huber',max_iter=1000,n_iter_no_change=5,
penalty='elasticnet',power_t=0.25,

random_state=None, shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0, warm_start=False)

SGDReg.coef_array([-0.00602635, 0.00566224, -0.00235155, 0.01238438])

SGDReg.intercept_ array([0.0043785])

SGDReg.t_ 55.0

3. Consider the following code on back propagation.

import numpy as np def sigmoidfun(x):

return 1.0/(1.0 + np.exp(-x)) def sigmoid_primefun(x):

return sigmoidfun(x)*(1.0-sigmoidfun(x)) def tanh(x):

return np.tanh(x) def tanh_prime(x):

return 1.0 - x**2

```
class NeuralNetwork:

def

init (self, layers, activation='tanh'): if activation == 'sigmoid':

self.activation = sigmoidfun self.activation_prime = sigmoid_primefun

elif activation == 'tanh': self.activation = tanh self.activation_prime = tanh_prime

# Setting weights self.weights = []

# let layers is [2, 2, 1]

# weight values range= (-1,1)

# hidden and input layers - random((2+1, 2+1)): 3 x 3 for i in range(1, len(layers)
- 1):

r = 2*np.random.random((layers[i-1] + 1,

layers[i] + 1)) -1

self.weights.append(r) # output layer - random

r = 2*np.random.random((layers[i] + 1, layers[i+1])) - 1

self.weights.append(r)

def fit(self, P, q, learning_rate=0.2, epochs=100000): # Adding the column of
ones to P

# This is to add the bias unit to the input layer ones =
np.atleast_2d(np.ones(P.shape[0]))

P = np.concatenate((ones.T, P), axis=1) for k in range(epochs):

i = np.random.randint(X.shape[0]) a = [P[i]]

for l in range(len(self.weights)):

dot_value = np.dot(a[l], self.weights[l]) activation = self.activation(dot_value)
a.append(activation)

# output layer error = q[i] - a[-1]

deltas = [error * self.activation_prime(a[-1])] # we need to begin at the second to
```

last layer # (a layer before the output layer)

for l in range(len(a) - 2, 0, -1): deltas.append(deltas[-1].dot(self.

weights[l].T)*self.activation_prime(a[l]))

reverse

[level3(output)->level2(hidden)] => [level2(hidden)->level3(output)]

deltas.reverse()

backpropagation

1. Multiply its output delta and input activation # to get the gradient of the weight.

2. Subtract a ratio (percentage) of the gradient from the weight.

for i in range(len(self.weights)): layer = np.atleast_2d(a[i]) delta = np.atleast_2d(deltas[i])

self.weights[i] += learning_rate * layer.T.dot(delta)

if k % 10000 == 0: print('epochs:', k) def predict(self, x):

c = np.concatenate((np.ones(1).T, np.array(x))) for l in range(0, len(self.weights)):

c = self.activation(np.dot(c, self.weights[l])) return c

if name == ' main ':

neu = NeuralNetwork([2, 2, 1])

P = np.array([[0, 0]

```
[0,          1],
[1,          0],
[1,          1]])
```

q= np.array([0, 1, 1, 0]) neu.fit(P,q)

for x in P:

print(x,neu.predict(x)) epochs: 0

epochs: 10000

epochs: 20000

epochs: 30000

epochs: 40000

epochs: 50000

epochs: 60000

epochs: 70000

epochs: 80000

epochs: 90000

[0	0]	[4.28817961e-05]
[0	1]	[0.99667242]
[1	0]	[0.99666444]
[1	1]	[4.62864326e-05]

CONCLUSION

In this chapter, we learned about the early models of neural networks, perceptron learning, application of neural networks in face recognition systems, recurrent networks, back propagation, and stochastic gradient descent. In the next chapter, we will discuss decision trees and regression trees, and their implementation in Python.

EXERCISES

1. Explain Stochastic Gradient Descent.

2. Explain applications of Neural Networks.

3. Explain Perceptron Learning.

4. Explain back propagation in Neural Networks.

REFERENCES

[1] O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, N.A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey", *Heliyon,* vol. 4, no. 11, p. e00938, 2018. [http://dx.doi.org/10.1016/j.heliyon.2018.e00938] [PMID: 30519653]

[2] M. Feindt, and U. Kerzel, "The NeuroBayes neural network package", *Nucl. Instrum. Methods Phys. Res. A,* vol. 559, no. 1, pp. 190-194, 2006. [http://dx.doi.org/10.1016/j.nima.2005.11.166]

[3] S. Albawi, T.A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network", *International conference on engineering and technology (ICET),* 2017pp. 1-6

[4] K.L. Du, "Clustering: A neural network approach", *Neural Netw.,* vol. 23, no. 1, pp. 89-107, 2010. [http://dx.doi.org/10.1016/j.neunet.2009.08.007] [PMID: 19758784]

CHAPTER 7

Supervised Learning

Abstract: Supervised learning involves training using well "labelled" training data, and on the basis of the training data, machines are able to predict the output. The labelled data means that the correct output is attached along with the corresponding input data. In supervised learning, as the name suggests, the training data acts as the supervisor and provides training to the machine to predict the correct output. This chapter discusses Statistical Decision Theory, Gaussian & Normal Distribution, Conditionally Independent Binary Components, Learning Beliefs Network and Nearest-Neighbour Methods.

Keywords: Belief networks, Normal distribution, Statistical decision theory, Supervised learning.

INTRODUCTION

Supervised Learning is a machine learning technique that involves mapping of input data with the corresponding output [1, 2]. Supervised learning is performed using supervised learning algorithm that provides a mapping function which can, input data with the output data. Real-life application of supervised learning is: Fraud Detection, Spam Filtering, Risk Assessment, Image Classification *etc* [3, 4].

USING STATISTICAL DECISION THEORY

Decision Theory is defined as the study of choices being made by the customers, professionals, voters, *etc*. Decision Theory is of two types: Normative Decision Theory and Optimal Decision Theory. In the analysis of decision theory, analysis is carried out regarding how an optimal decision is made, what are the characteristics of an optimal decision maker and how an optical decision maker can arrive at an outcome.

Decision Theory may be defined as the study of choices of agents or persons. It assists in understanding various choices that are made at the time of decision by customers, professionals, voters, *etc*.

The three kinds of uncertainty found in decision theory are: Actions, States and Consequences. Marketers use decision theory as an excellent tool for understanding consumer behaviour.

Normative Decision performs analysis of the result of decisions. Normative Decision Theory is responsible for taking optimal decisions on the basis of the result obtained. It is responsible for obtaining results for a specific situation. In Optimal Decision Theory, a detailed analysis and an investigation is carried out on how and why the choices are made by the individuals and agents after decision. Decision Theory is helpful in predicting consumer behaviour. It is helpful in knowing whether the product is helpful in understanding why customer chooses a particular product, and which transportation is more suitable for carrying out finished products? It is helpful in getting information that the product will be successful or not. The three different uncertainties found in decision making include- state, actions and consequences. States represent the existing list of facts that may affect the decision. Consequences represent the characteristics of decisions that are taken by the decision maker. Action is the bond between the state and consequences.

Gaussian or Normal Distribution

Gaussian distribution is also referred to as Normal distribution or Gauss or Laplace - Gauss distribution. For a given real valued random variable, gauss distribution represents the continuous probability distribution. The probability density function of the gaussian or normal distribution is represented as follows:

$$f(x)=(1/\sigma 21/2\pi)e-1/2[(x-\mu)/\sigma]2$$

Here, μ represents the expectation or mean. It also represents median or mode. The parameter σ represents the standard deviation. $\sigma 2$ represents the variance. A variable having Gaussian distribution is said to exhibit normal distribution or normal deviation. Gaussian distribution is used for the representation of real-value random variables whose distribution is unknown. Gaussian distribution makes use of the central limit theorem. According to the central limit theorem, for a given random variable, having a finite value of variance and mean, as the number of observations of these random variables increases, its distribution converges to normal. One of the characteristics of a Gaussian distribution is that a linear combination of various normal deviates is a normal deviation. A Gaussian distribution is also sometimes referred to as a bell curve. But, there are other distributions as well that may have a bell curve. A special case of a Gaussian distribution is referred to as a Unit Normal distribution or Standard Normal

distribution when μ=0 and σ=1 and the density or probability density function is represented as follows:

$$\varphi(z)=[\{(e-(z2)/2\}/21/2\pi]$$

Here, the mean and variance of z is 0 and the standard deviation of z is 1.

Conditionally Independent Binary Components

The term conditional independence for a given set of multiple variables was given by Dawid in 1980.

An important concept for probability distributions over multiple variables is that of conditional independence (Dawid, 1980).

Consider three variables p, q, and r, and imagine that the conditional distribution of p, given q and r, is such that it is not dependent on the value of q, then:s

$$p(p|q, r) = p(p|r).$$

We can say that p is conditionally independent of q given r. The above equation can be rewritten in a different way if we consider joint distribution of p and q conditioned on r, which we can write in the following form:

$$p(p, q|r) = p(p|q, r)p(q|r)$$

$$= p(p|r)p(q|r)$$

In the above equation, product rule has been used. So, it is contained on q. The joint distribution of p and q factories into a product of marginal distribution of p as well as marginal distribution of q. It means that p and q are statistically independent given r.

Conditional independence has been used for creating probabilistic models for pattern recognition by performing simplification of complex structures as well as computations required in applying inference under various conditions.

If the product of conditional distributions is given in form of an expression for a particular directed graph, then we can check if any conditional independence property holds or not. Looking at the graphical models; we can derive the conditional independence property of joint distribution. This graphical framework

for deriving conditional independence is referred to as a d-separation. Here, d denotes 'directed'. The concept of d-separation was given by Pearl in 1988.

Conditional independence is a probabilistic approach and it has applications in machine learning and causal inference. In Machine Learning, Conditional independence is used for feature selection and it acts as a non-parametric method in machine learning.

LEARNING BELIEF NETWORKS

A belief network provides information about probability distribution over a set of random variables. A model can be built in which a belief network may be learnt from the given set of data. There may be many variants of a belief network. In one of the networks, the conditional probability of each variable given its parent information is learned. The learning agent might be provided with the structure of the model along with observed variables. The agent then learns the conditional probabilities. The task of learning the conditional probabilities in a belief network may be considered an instance of supervised machine learning. If there exist very less parent nodes, then conditional probability may be learned with the help of training examples and using prior information. The prior information is known as pseudo counts. Other names of belief networks are: Relevance diagrams, causal networks, influence diagrams and Bayesian networks. One type of belief network is the sigmoid belief network. In the sigmoid belief network, the values that may be accepted are 0 or 1 and -1 or 1. Here, the weight from unit j to unit I is shown as wij. The bias unit 0 is set to 1 and its weight is denoted as wi0. Another belief network is the noisy-OR belief network in which Gibbs sampling is used. The learning procedure is based on maximum likelihood estimation.

NEAREST-NEIGHBOUR METHODS

The Nearest Neighbour algorithm is used to implement travelling salesman problem. In travelling salesman problem, a salesman starts his journey from a random city and keep visiting the next city which is nearest. This process is repeated until all the cities are visited exactly once. This travelling salesman problem using the nearest neighbour finds the short tour but the tour may or may not be an optimal tour. The nearest neighbour algorithm is easy to implement but it may sometimes miss the shortest route or the optimal outcome. The K-nearest neighbours algorithm(KNN) is referred to as non-parametric supervised learning technique that is being used for regression and classification purposes [3, 4]. It was developed in 1951 by Evelyn Fix and Joseph Hodges and later on, it was expanded by Thomas Cover. In regression using KNN, the output is found by taking an average of k nearest neighbours values [3, 4]. In classification using KNN, an object may be classified on the basis of the majority voting of its

neighbours. If K=1, then the object is allotted the class of nearest neighbour. The large value of k helps in the reduction of noise during the classification process but may also lead to more boundaries between the classes which are less different. The value of k can be calculated using the heuristics technique. The Nearest Neighbour algorithm is a special case of the KNN algorithm where K=1 and the class is predicted to be a class in the nearest training dataset. The presence of noise can affect the accuracy of the KNN algorithm. Supervised metric learning can be used to improve the performance of the KNN algorithm. Supervised metric learning makes use of label information for learning a pseudo metric or a new metric. Examples of these algorithms include: large margin nearest neighbour, neighbourhood component analysis *etc*. Condensed Nearest Neighbour (CNN) is also referred to as the Hart algorithm. This algorithm reduces the size of the training dataset and performs classification as accurately as the classification performed by KNN algorithm.

KNN algorithm is similar to the Supervised Learning technique. KNN algorithm works on the similarity technique and allots the category to the new data which is similar to the available categories. KNN is said to be non-parametric algorithm. So, it does not make any pre assumptions on the underlying dataset. It is known as lazy learner algorithm; since KNN is not capable of quickly learning from training data instead it stores the data and produces results when the classification task is performed.

Advantages of KNN Algorithm include the following:

- It is easier to implement.
- Its performance is not impacted by the presence of noisy training data.
- It produces better results if the training data is huge.

Disadvantages of KNN Algorithm include the following:

- Determining the value of K is a complex process.
- The cost of computation may be high.

CONCLUSION

In this chapter, we have learned about supervised learning, belief network, the K-nearest neighbour algorithm, conditional independence, statistical decision theory and Gaussian distribution. In the next chapter, we will discuss Clustering (K-means Clustering, Hierarchical Clustering), and Principal Component Analysis.

EXERCISES

1. What is supervised machine learning? Explain with an example.

2. Explain conditional independence.

3. Explain Gaussian distribution.

4. Explain with example, k nearest neighbour algorithm.

REFERENCES

[1] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms", *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom),* 2016pp. 1310-1315

[2] T. Jiang, J.L. Gradus, and A.J. Rosellini, "Supervised machine learning: a brief primer", *Behav. Ther.,* vol. 51, no. 5, pp. 675-687, 2020.
[http://dx.doi.org/10.1016/j.beth.2020.05.002] [PMID: 32800297]

[3] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques", *Emerging artificial intelligence applications in computer engineering,* vol. 160, 2007.

[4] F.Y. Osisanwo, J.E.T. Akinsola, O. Awodele, J.O. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison", *Int. J. Comput. Trends Tech.,* vol. 48, no. 3, pp. 128-138, 2017.
[http://dx.doi.org/10.14445/22312803/IJCTT-V48P126]

CHAPTER 8

Unsupervised Learning

Abstract: Unsupervised learning is a complex processing task involving the identification of patterns in datasets having data points that are neither labeled nor classified. Unsupervised learning is a kind of machine learning algorithm that can be used to draw useful conclusions without the presence of labeled responses in the input data. In the following chapter, we will discuss Clustering (K-means Clustering, Hierarchical Clustering), and Principal Component Analysis.

Keywords: Clustering, Principal component analysis, Unsupervised learning.

INTRODUCTION

In unsupervised learning, an uncategorized and unlabeled data is sent to the AI system and the algorithms act on this data without any prior training. In unsupervised learning, it is important to know the following outcomes:

- To know about unsupervised learning
- Understanding Clustering and Clustering Algorithms
- (K-means Clustering, Hierarchical Clustering)
- Understanding Principal Component Analysis

CLUSTERING

The term Clustering was first used in 1932 in anthropology by *Driver* and *Kroeber*. It was used in 1938 in psychology by *Joseph Zubin* and in 1939 by *Robert Tryon*. It was used for trait theory classification in personality psychology in 1943 by *Cattell*. Clustering also referred to as cluster analysis is a process of grouping together similar objects into the same group called cluster in such a way that objects in one cluster are not similar to the objects in another cluster. Cluster analysis is used in many fields such as data compression, pattern recognition and image processing, machine learning, computer graphics, information retrieval, and bioinformatics [1]. In the following chapter, we will discuss clustering algorithms such as k-means clustering algorithm and hierarchical clustering algorithm.

K-means Clustering

K-means clustering involves the partitioning of observations into k clusters in which a given observation belongs to the cluster having the closest mean [2, 3].

K-means Clustering for solving general problems involves the following steps:

- Cleaning and Transforming data
- Choosing the value of K and K-means algorithm is made to run.
- Reviewing the result obtained
- Iterating over different values of K

Hierarchical Clustering

Hierarchical clustering is also referred to as hierarchical cluster analysis. This method involves building a hierarchy of clusters [1]. Two approaches used in hierarchical clustering analysis include the following:

- Agglomerative clustering
- Divisive clustering

Agglomerative clustering is a 'bottom-up' approach that involves each individual cluster, and the pairs of clusters merge and move higher in the hierarchy. Divisive clustering is a 'top-down' approach in which a single cluster is further split into multiple clusters as we proceed down in the hierarchy.

Cluster dissimilarity is a metric that measures the distance between the pair of observations and the linkage criterion that specifies a dissimilarity between the pair of observations. Cluster dissimilarity is used in agglomerative clustering to decide which cluster would join together to form the bigger cluster. It is also used in divisive clustering to decide which cluster would finally break.

Principal Component Analysis (PCA)

PCA refers to the dimensionality reduction methodology that can be used for reducing the dimensions of large data sets into smaller ones, while still preserving as much information as possible [4].

Steps performed in PCA include the following:

1. Standardization

2. Computation of Covariance Matrix

3. Identification of Principal Components by computing the Eigen Values and Eigen Vectors of Covariance Matrix

4. Creating a Feature Vector

5. Recasting the data

In the first step, standardization is important because if there are some values that are large and some that are small, then the larger ranges would dominate over the smaller ranges. So, standardization is performed to solve this problem. Standardization can be done by subtracting the mean from the value and dividing it by the standard deviation. This is represented as follows:

Z=(value-mean)/standard deviation

In the second step, the covariance matrix is computed to find whether there exists redundant information in the input data. In the third step, the principle components are generated, which are nothing but the variables that are formed by the linear combination of the initial variables. In the fourth step, a feature vector is created by considering those principle components that are of a higher significance over the ones with a lower significance.

PYTHON CODE

1. Consider the following code in python using k-means clustering:

```
print(  doc  ) import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

from sklearn.cluster import KMeans from sklearn import datasets

np.random.seed(5)

irisdata = datasets.load_iris()

P = irisdata.data

q = irisdata.target
```

In the following code, n_init is set to 1 instead of 10 which is a default value. So, this bad initialization has an impact on the classification process as it reduces the number of times an algorithm runs with different centroid seeds.

```
estimatorskmeans = [('k_means_iris_8', KMeans(n_ clusters=8)),

('k_means_iris_3', KMeans(n_clusters=3)), ('k_means_iris_bad_init', KMeans(n_

clusters=3, n_init=1,

init='random'))] fignum=1

titles = ['8 clusters', '3 clusters', '3 clusters, bad initialization']

for name, est in estimatorskmeans:

fig = plt.figure(fignum,figsize=(4,3))

px = Axes3D(fig, rect=[0, 0, .82,1], elev=52,azim=147) est.fit(P)

labels = est.labels_

px.scatter(P[:, 3], P[:, 0], P[:, 2],

c=labels.astype(np.float), edgecolor='k')

px.w_xaxis.set_ticklabels([]) px.w_yaxis.set_tick labels([]) px.w_zaxis.set_tick labels([]) px.set_xlabel('Petal width') px.set_ylabel('Sepal length')

px.set_zlabel('Petal length') px.set_title(titles[figno-1]) px.dist = 12

fignum=fignum+1
```

Plotting the ground truth values takes place:

```
fig = plt.figure(fignum, figsize=(4, 3))

px = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

for name, label in [('Setosa', 0),

('Versicolour', 1),

('Virginica', 2)]: px.text3D(P[q == label, 3].mean(), P[q == label, 0].mean(),

P[q == label, 2].mean() + 2, name, horizontalalignment='center', bbox=dict(alpha=.2, edgecolor='w',

facecolor='w'))
```

Labels are reordered so that the colors are matched with the cluster results:

```
q= np.choose(q,[1, 3, 0]).astype(np.float)

px.scatter(P[:, 3], P[:, 0], P[:, 2], c=q, edgecolor='k')

px.w_xaxis.set_ticklabels([]) px.w_yaxis.set_tick labels([]) px.w_zaxis.set_tick labels ([]) px.set_xlabel('Petal width') px.set_ylabel('Sepal length') px.set_zlabel('Petal length') px.set_title('Ground Truth') px.dist = 12

fig.show()
```

The output of the above code is shown in Fig. (**1**) :

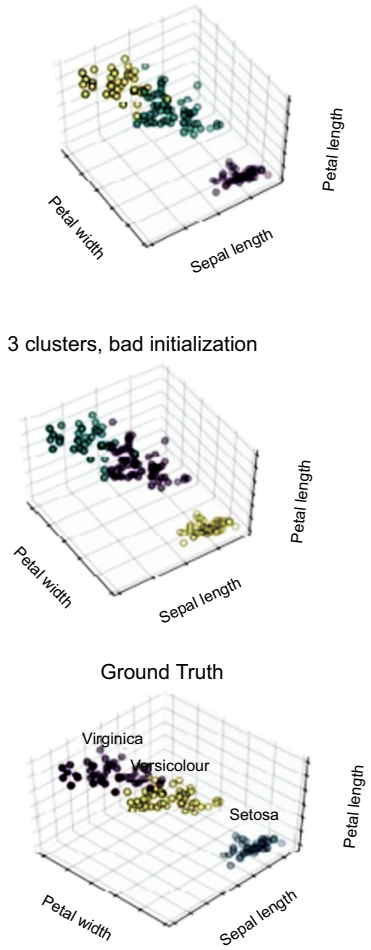

Fig. (1). K-means clustering.

2. Consider the following code in Python on PCA. We are considering a two-

dimensional data set having 600 points:

```
import numpy as np

import matplotlib.pyplot as plt import seaborn as sns; sns.set() ranstat = np.random.RandomState(1)

P = np.dot(ranstat.rand(2, 2), ranstat.randn(2, 600)).T plt.scatter(P[:, 0], P[:, 1])

plt.axis('equal');
```

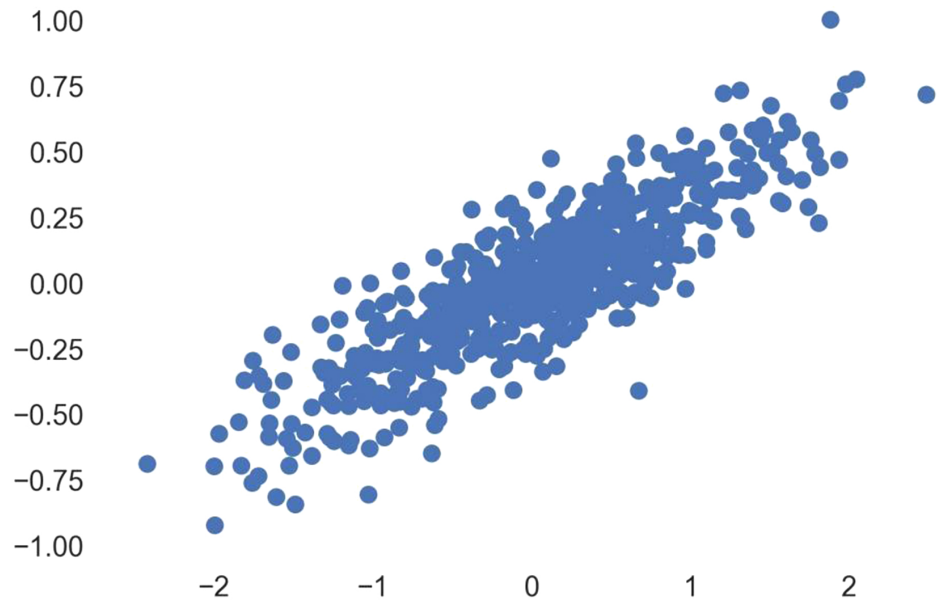

Fig. (2). Example on Principal Component Analysis.

In the above Fig. (**2**) , there exists a linear relation between X and Y. In Principal Component Analysis, a list of principal axes is found and these axes are used to describe the data set. Using scikit-learn, we can compute the principal axes as follows:

```
from sklearn.decomposition import PCA principalaxis = PCA(n_components=2) principalaxis.fit(P)

PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,

svd_solver='auto', tol=0.0, whiten=False)
```

Principal axis fit learns components and explained variance as

follows: print(principalaxis.components_) [[0.9513521 0.3081058]

[0.3081058 -0.9513521]]

print(principalaxis.explained_variance_) [0.70649509 0.02104542]

def draw_vector (n0, n1, px=None): px = px or plt.gca() arrowprops=dict(arrowstyle= '->',

linewidth=2, shrinkA=0, shrinkB=0)

px.annotate(“, n1, n0, arrowprops=arrowprops)

plot data

plt.scatter(P[:, 0], P[:, 1], alpha=0.2)

for length, vector in zip(principalaxis.explained_ variance_, principal axis.components_):

p = vector * 3 * np.sqrt(length) draw_vector(principalaxis.mean_, principal axis.mean_ + p)

plt.axis('equal');

We can define the input data as vectors and represent the direction of a vector using the components and the squared length of vector is defined using an explained variance. The principal axis is represented by the data. The variance of the data is represented by the length of the vector. This is shown in the code given below:

def draw_vector(n0, n1, px=None): px = px or plt.gca() arrowprops= dict (arrowstyle= '->',

linewidth=2, shrinkA=0, shrinkB=0)

px.annotate(“, n1, n0, arrowprops=arrowprops)

plot data

plt.scatter(P[:, 0], P[:, 1], alpha=0.2)

for length, vector in zip(principalaxis.explained_ variance_, principal axis.components_):

p = vector * 3 * np.sqrt(length) draw_vector(principalaxis.mean_, principal axis.mean_ + p)

plt.axis('equal');

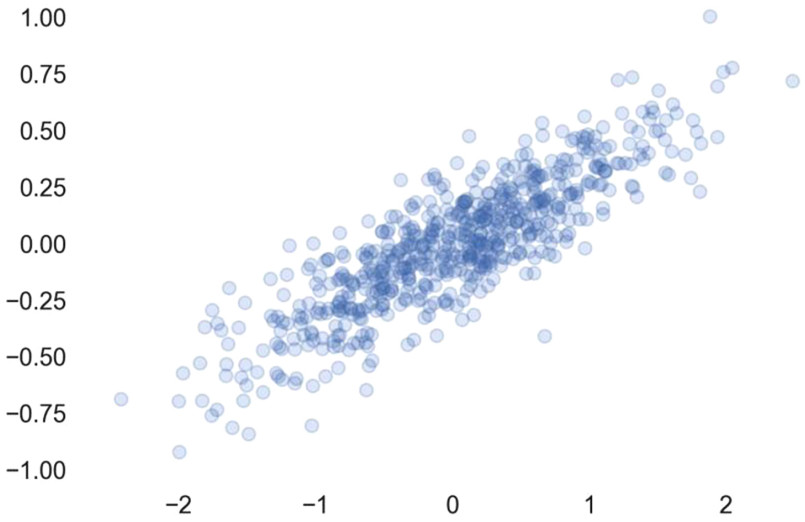

Fig. 3. Variance represented as the length of the vector.

In Fig. (**3**) , the variance is represented as the length of the vector. The value of one or more of the principal components is made 'zero' in order to perform dimensionality reduction. Inverse transform on the reduced data can be found and the plot can be drawn on the original data.

principalaxis = PCA(n_components=1) principalaxis.fit(P)

X_principalaxis = principalaxis.transform(P)

print("original shape: ", P.shape)

print("transformed shape:", X_principalaxis.shape)

original shape: (600, 2)

transformed shape: (600, 1)

P_new = principalaxis.inverse_transform(X_principalaxis) plt.scatter(P[:, 0], P[:, 1], alpha=0.2)

plt.scatter(P_new[:, 0], P_new[:, 1], alpha=0.8) plt.axis('equal');

Fig. (**4**) shows inverse transform on the reduced data.

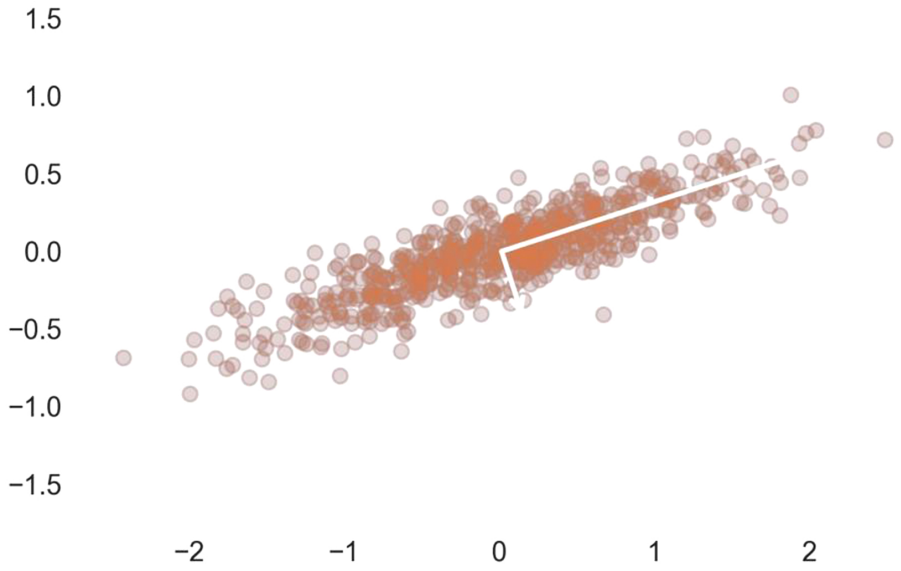

Fig. (4). Inverse transform on the reduced data.

CONCLUSION

In this chapter, we learnt about clustering, k-means clustering, hierarchical clustering, and Principal Component Analysis. In the next chapter, we will discuss the theory of generalization, training versus testing, bounding the testing error, Vapnik Chervonenkis inequality, VC Dimension, and the proof of VC inequality.

EXERCISES

1. Explain k-means clustering with an example.

2. Explain hierarchical clustering.

3. Explain Principal Component Analysis with an example.

REFERENCES

[1] F. Murtagh, and P. Contreras, "Algorithms for hierarchical clustering: an overview", *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.,* vol. 2, no. 1, pp. 86-97, 2012. [http://dx.doi.org/10.1002/widm.53]

[2] A. Likas, N. Vlassis, and J.J. Verbeek, "The global k-means clustering algorithm", *Pattern Recognit.,* vol. 36, no. 2, pp. 451-461, 2003. [http://dx.doi.org/10.1016/S0031-3203(02)00060-2]

[3] P.S. Bradley, and U.M. Fayyad, "Refining initial points for k-means clustering", In: *ICML* vol. 98. , 1998, pp. 91-99.

[4] H. Abdi, and L.J. Williams, "Principal component analysis", *Wiley Interdiscip. Rev. Comput. Stat.,* vol. 2, no. 4, pp. 433-459, 2010.
[http://dx.doi.org/10.1002/wics.101]

CHAPTER 9

Theory of Generalisation

Abstract: Unsupervised learning is a kind of machine learning algorithm that can be used to draw useful conclusions without the presence of labeled responses in the input data. In the previous chapter, we discussed clustering (k-means clustering, hierarchical clustering) and Principal Component Analysis. In this chapter, we will discuss training versus testing, bounding the testing error, and the VC dimension.

Keywords: Testing, Training, VC dimension.

INTRODUCTION

Training data helps the algorithm to learn from experience. In supervised learning, each observation comprises an input variable and the corresponding target variable. For building a model, a training set is implemented and for validating a test set, a testing set is required. The data set is divided into the training set and the test set [1, 2].

In machine learning, a model is created in order to perform testing on the test data. To fit the model, training data is used and to perform testing, test data is used. It is not necessary to use 70% of the data set for developing the training set and the rest for the purpose of testing. It depends on the data set that is being used and the task that needs to be accomplished.

BOUNDING THE TESTING ERROR

Principal Component Analysis refers to the dimensionality reduction methodology that can be used for reducing the dimensions of large data sets into smaller ones, while still preserving as much amount of information as possible.

Steps performed in Principal Component Analysis include the following:

1. Standardization

2. Computation of Covariance Matrix

3. Identification of Principal Components by computing the Eigen Values and the Eigen Vectors of the Covariance Matrix

4. Creating a Feature Vector

5. Recasting the data

The first step, standardization, is important because if there are some values that are large and some small, then the larger ranges would dominate over the smaller ranges. So, standardization is performed to solve this problem. Standardization can be done by subtracting the mean from the value and dividing it by the standard deviation. This is represented as follows:

Z=(value-mean)/standard deviation

In the second step, the covariance matrix is computed to find whether there exists redundant i.

VAPNIK CHERVONENKIS INEQUALITY

VC dimension was originally given by *Vladimir Vapnik* and *Alexey Chervonenkis*. VC dimension may be defined as a measure of some of the features in terms of complexity, flexibility, richness or the expressive power of the set of functions that may be learned using a statistical binary classification algorithm.

Uses of VC dimension include the following:

- VC dimension is used in the statistical learning theory for the prediction of the probabilistic upper bound of the test error of a classification model [3, 4].
- VC dimension is also used in sample complexity bounds. Sample complexity may be defined as the linear function of the VC dimension of the hypothesis space.
- VC dimension is used in computational geometry for the prediction of the complexity of approximation algorithms.

PROOF OF VC INEQUALITY

The main outcome in Statistical learning is VC inequality which is a difference in generalisation and empirical risk.

The VC inequality is represented for binary hypothesis classes.

$$\{h:X\rightarrow\{0,1\}\}$$

It represents the upper bound on the absolute difference between the given empirical and the true risk, when they are defined with 0-1 loss .The VC dimension can be defined for binary classifiers only.

The inequality can be generalized with the help of different complexity measures (like Rademacher complexity or pseudo-dimension).

CONCLUSION

In this chapter, we learned about training versus testing, bounding the testing error, and VC dimension. In the next chapter, we will discuss how to detect bias and how to fix bias or achieve fairness in ML.

EXERCISES

1. Explain the difference between training and testing in machine learning.

2. Explain bounding the testing error.

REFERENCES

[1] C.A. Corneanu, S. Escalera, and A.M. Martinez, "Computing the testing error without a testing set", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 2677-2685, 2020.
[http://dx.doi.org/10.1109/CVPR42600.2020.00275]

[2] Z. Yuan, Y. Yan, R. Jin, and T. Yang, "Stagewise training accelerates convergence of testing error over SGD", *Adv. Neural Inf. Process. Syst.,* p. 32, 2019.

[3] R.P. Lamberts, J. Swart, R.W. Woolrich, T.D. Noakes, and M.I. Lambert, "Measurement error associated with performance testing in well-trained cyclists: application to the precision of monitoring changes in training status", *International SportMed Journal,* vol. 10, no. 1, pp. 33-44, 2009.

[4] S.L. Wise, L.E. Lukin, and L.L. Roos, "Teacher beliefs about training in testing and measurement", *J. Teach. Educ.,* vol. 42, no. 1, pp. 37-42, 1991.
[http://dx.doi.org/10.1177/002248719104200106]

CHAPTER 10

Bias and Fairness in Ml

Abstract: In machine learning and AI, future predictions are based on past observations, and bias is based on prior information. Harmful biases occur because of human biases which are learned by an algorithm from the training data. In the previous chapter, we discussed training versus testing, bounding the testing error, and VC dimension. In this chapter, we will discuss bias and fairness.

Keywords: Bias, Confidence intervals, Fairness, Hypothesis testing.

INTRODUCTION

Bias is referred to as a disproportionate prejudice or inclination towards a particular thing or an idea. Bias may be found in the following different fields:

- Research
- Statistics
- Social sciences

In machine learning, algorithmic biases are referred to as unwarranted associations. Algorithmic biases are the bugs that can be harmful to the business and people. Following are the objectives of this chapter:

- Understanding how to identify bias.
- Understanding how to achieve fairness in ML.

HOW TO DETECT BIAS?

Bias is one of the popular topics that one encounters while building AI-based models. Many uncommon and common biases may be found in the following stages of AI model development:

- Data collection
- Data preprocessing

- Data analysis
- Modeling

During data collection, biases may take place. This happens due to the occurrence of outliers and errors that happen while collecting data.

Biases that are found during the data collection process include the following:

- **Selection bias**: While preparing the sample data, the selection of data must be done in a proper manner to avoid bias. For example, if the participants are students who are to undergo tests, then they may include the bias results.
- **The Framing Effect**: Survey questions are framed in such a manner that biases are avoided and displays positivity in sentences, or else biases crop up.
- **Systematic bias**: It occurs because of faulty equipment. It leads to repeatable and consistent errors.
- **Response bias**: It occurs due to questions that are answered incorrectly by the participants.

During data preprocessing, the following steps may be undertaken:

1. Outlier detection

2. Missing values

3. Filtering data

Outliers lead to a disproportionate effect on many of the analyses that are conducted.

While dealing with the missing values, if all the missing values are replaced by the mean values, then it would mean being biased towards a particular group that is closer to the mean.

Biases may be found during the process of data analysis. Biases may be found using the following approaches:

- **Missing graphs**: Incorrect conclusions may be drawn from a distorted graph that provides incorrect information.
- **Confirmation bias**: It involves the tendency to focus and confirm information that is related to someone's preconceptions.

When performing data modelling, it is very important to detect biases. For example, Amazon created a hiring algorithm that showed gender bias by favoring

men as high-potential candidates. A model that has high variance focuses on training data and doesn't generalize well. Data always behaves in the same way in high bias. When we increase bias, variance decreases and *vice versa* . In supervised machine learning, training is performed on the input variables in such a manner that there is closeness between the predicted values and the actual values. The error refers to the difference between the actual and the predicted values. There are 3 types of errors in supervised machine learning:

- Bias error
- Variance error
- Noise

Bias and variance are reducible errors that we can minimize to a large extent. Noise is said to be an irreducible error that cannot be eliminated.

HOW TO FIX BIASES OR ACHIEVE FAIRNESS IN ML?

There are already many definitions of fairness as per literature and these cover the following elements:

- Equalized odds
- Unawareness
- Individual fairness
- Demographic parity
- Counterfactual fairness
- Predictive rate parity

We should avoid including a sensitive attribute as one of the features in training data. There are many ways to mitigate biases. Some of these techniques include:

- Preprocessing
- In-processing
- Post-processing

The pre-processing approach takes place before the development of a model. Its main intent is to eliminate the underlying bias from the data set before modelling. This is one of the basic approaches to removing biases from the data. In-processing is the process of removing biases during the training phase. In post-processing, the elimination of biases takes place after the training phase is over.

CONFIDENCE INTERVALS

Confidence intervals are used as a measure of estimation in an uncertain situation. They can be used to add estimation or likelihood and make use of parameters such as mean. They are derived from a field known as estimation statistics.

A confidence interval poses a bound on the population estimation variable. There lies a difference between the confidence interval, tolerance interval and prediction interval. Tolerance interval is meant for bounds of data which is sampled from a given distribution. Prediction interval is related to bounds on a given single observation. The confidence parameter is related to the representation of skills in a predictive model and classification model. Error bars on a graph may be used to represent uncertainty. The confidence interval is known as an error margin. A precise estimate is usually drawn from a large sample and so it has a smaller or a better value of confidence interval. When there is a less precise value of estimate, it means that it has a large value of confidence interval. Confidence interval is a branch of statistics referred to as estimation statistics that is used for the estimation of experimental outcomes. Practitioners usually prefer confidence intervals and not statistical tests; as confidence intervals are much easier to use and may be used for the comparison of various machine learning models. Confidence interval may also be used by consumers to know whether a given model is good or bad. If the confidence interval of the two models is the same; then the model which is simple and interpretable is preferred over the complex model.

HYPOTHESIS TESTING

Hypothesis testing is used by analysts for carrying out tests related to the population parameter [1, 2]. The kind of data and reason for doing analysis decide the methodology being used by the analyst. The sample data for checking the hypothesis may be derived from a given large population or with the help of the data generation process. An analyst performs testing on the statistical sample. The two hypotheses mainly: null hypothesis and alternative hypothesis are tested by the analysts by using a random sample population. The null hypothesis is the opposite of alternative hypothesis. Both the hypotheses are mutually exclusive. Only one out of the two given hypotheses can be true.

Steps involved in Hypothesis Testing include the following:

- The first step in hypothesis testing involves stating both the hypotheses but only one out of the two hypotheses would be correct.

- The next step is an analysis plan that states how the evaluation of data will be performed.
- The third step in hypothesis testing involves the execution of the plan and a detailed analysis of sample data is performed.
- The last step in hypothesis testing involves a detailed analysis of the result and on the basis of the analysis, either reject or accept the null hypothesis.

COMPARING LEARNING ALGORITHMS

There are numerous Machine Learning algorithms that are used in different walks of life *e.g.* in industries and in research. It is extremely difficult for us to understand and remember each and every detail of all the machine learning models. It is wrong to say that all these algorithms are not related to each other. We have some parameters on the basis of which we may decide that algorithm A will be used over algorithm B. Following are the parameters; with the help of these parameters, we can decide which machine learning algorithm should be chosen in a particular situation:

- **Time Complexity-** The number of operations performed by an algorithm is measured by its time complexity [3]. We must use a standard method to compute the time complexity of ML models. The time complexity during the training and testing phase of the Machine Learning model may vary. The linear regression has a large training time but a smaller and more efficient testing time.
- **Space Complexity-** It is a measure of amount of memory or space needed to run a given machine learning algorithm. A Machine Learning algorithm must efficiently run even if a large amount of input data is loaded into the machine by the algorithm.
- **Sample Complexity-** It is defined as the number of training samples required to train a given Machine Learning algorithm. A deep learning neural network has high sample complexity, since a large amount of training data is required to train this model.
- **Bias-variance tradeoff-** In a Machine Learning model, a bias error takes place when the model is biased and gives a specific assumption or solution only. For *e.g.* bias will be high if a linear decision boundary fits on nonlinear data. Variance is a measure of errors that occur due to the variance of model. Variance is defined as the average square difference in the prediction of a given model and the expected model [4].
- **Online and Offline-** It is the type of learning required to update a given Machine Learning model. The Machine Learning algorithm which has an Online learning is more efficient as compared to the Machine Learning algorithm having Offline learning. In Online learning, parameters are updated immediately when a new training sample is available. In offline learning, a machine learning model is

retrained when a new training sample is available. For *e.g.* stochastic gradient descent is based on Online learning, whereas ID3 Decision Tree algorithm is based on Offline learning.

- **Parallelizability-** It is a property through which a given Machine Learning algorithm is capable of performing multiple different operations at the same time. Gradient boosting decision tree is an example of the sequential algorithm in which a given decision tree is built when the previous decision tree is built and all the errors that occurred in previous tree are removed in the next tree to be drawn. K-nearest neighbor is an example of parallel algorithm.
- **Parametricity-** It is a property of number of parameters to be added when more data is available. In a parametric model, parameters remain fixed and no more parameter is added even if more data is available. In a non-parametric model, more parameters may be added when more data is available. Neural Networks and linear regression are an example of Parametric models whereas KNN and SVM are examples of Non-Parametric models.

The above mentioned criterias help in analysing whether a given Machine Learning algorithm is efficient or effective.

CONCLUSION

In this chapter, we learnt about biases, how to detect biases, and how to fix them and achieve fairness. We also discussed hypothesis testing and parameters that tell whether a given algorithm is effective and efficient.

EXERCISES

1. Explain how we can detect a bias.

2. Explain how we can fix a bias.

3. On what parameters can we judge which machine learning algorithm is efficient for a particular situation?

REFERENCES

[1] W.K. Newey, and D. McFadden, "Large sample estimation and hypothesis testing", In: *Handbook of econometrics* vol. 4. , 1994, pp. 2111-2245.

[2] M. Snyder, and W.B. Swann, "Hypothesis-testing processes in social interaction", *J. Pers. Soc. Psychol.,* vol. 36, no. 11, pp. 1202-1212, 1978. [http://dx.doi.org/10.1037/0022-3514.36.11.1202]

[3] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms", *J. Syst. Softw.,* vol. 137, pp. 184-196, 2018. [http://dx.doi.org/10.1016/j.jss.2017.11.066]

[4] M. Aniche, E. Maziero, R. Durelli, and V. Durelli, "The effectiveness of supervised machine learning algorithms in predicting software refactoring", *IEEE Trans. Softw. Eng.,* 2020.

APPENDIX

Case Study- Use of Machine Learning in Harley Davidson

The owner of New York dealership Asaf Jacobi had to sell one or two Harley-Davidson motorcycles in a coming week. One day, during Winter, he was walking in Riverside Park and he met Or Shani, the CEO of Adgorithms. Asaf Jacobi discussed the low sales number with Shani. Shani then suggested him to use Albert, which is AI enabled and has been developed by Algorithms for marketing purposes. Albert can work on different platforms like Facebook and Google. When Asaf Jacobi used Albert; he was able to sell 15 motorcycles that week. The sales were increasing day by day. The sale just doubled of what he managed to sell during summer. In the first month, the new lead was 15%. In the third month, the lead grew by 29-30%. Jacobi now had to expand his current call center so as to handle the current growing business. Today AI and Machine Learning have been used by all leading firms such as Amazon, Google, Facebook, *etc.* AI and Machine Learning have helped these companies to reach the target customers using personalised marketing campaigns. Using Albert in case of Harley Davidson, store traffic increased as it generated leads which were target customers that had shown interest in the form that was filled. Albert is provided content from Harley Davidson in order to improve its sales. Albert could analyse the behaviour of target customers. Albert targeted people who have completed their purchases, have been adding items to the cart and are considered among the top 25% people to visit the Harley Davidson website. Albert ran tests in various groups and got data related to high-value customers. Albert tried to improvise digital marketing campaigns by providing successful conversations. Albert could gather and do the processing of millions of interactions in a minute. So, it saved a lot of time. For the same task, humans would have taken a lot of time.

CONCLUSION

Incorporating AI and Machine Learning specially in business such as in the case of Harley Davidson would give great success by increasing decision-making power, taking the right decisions, reducing marketing time, *etc.* It will definitely lead to an escalation in its growth in terms of sales in no time.

SUBJECT INDEX

G

H

I

K

L

M

N

O

P

R

S

T

V

W

Deepti Chopra

Dr. Deepti Chopra has done PhD in the area of Natural Language Processing from Banasthali Vidyapith. Currently, she is working as Associate Professor at JIMS Rohini, Sector 5. Dr. Chopra is an author of five books and two MOOCs. Two of her books have been translated into Chinese and one has been translated into Korean. She has 2 Australian Patents and 1 Indian Patent to her credit. Dr. Chopra has several publications in various International Conferences and journals of repute. Her areas of interest include Artificial Intelligence, Natural Language Processing and Computational Linguistics. Her primary research works involve machine translation, information retrieval, and cognitive computing.

Roopal Khurana

Mr. Roopal Khurana is working as Assistant General Manager at Railtel Corporation of India Ltd., IT Park, Shastri Park, Delhi. Currently, he is working in the field of Data Networking, MPLS Technology. He has done BTech in Computer Science and Engineering from GLA University, Mathura, India. He is a technology enthusiast. Previously, he has worked with companies, such as Orange and Bharti Airtel.