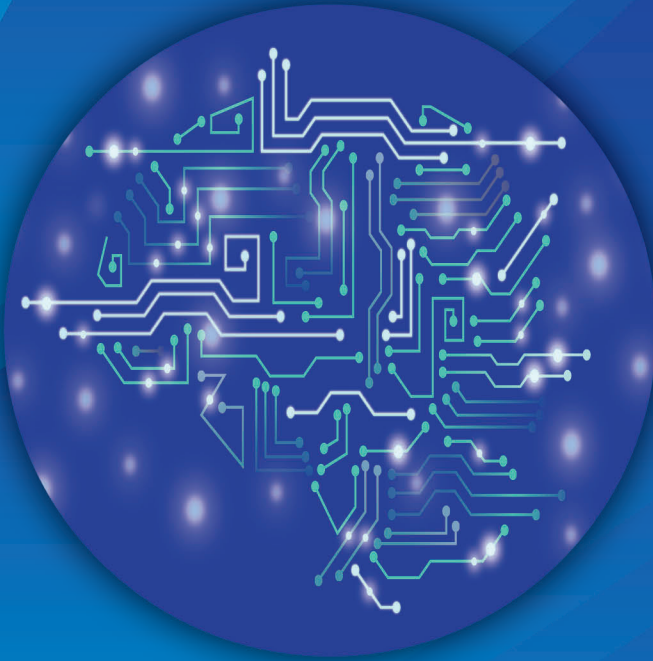


A Practical Approach for

Machine Learning

and

Deep Learning Algorithms



Tools and Techniques Using MATLAB and Python

ABHISHEK KUMAR PANDEY

PRAMOD SINGH RATHORE

DR. S. BALAMURUGAN



A Practical Approach for
**Machine Learning and Deep
Learning Algorithms**

Tools and Technique using MATLAB and Python

By
Abhishek Kumar Pandey
Pramod Singh Rathore
Dr. S. Balamurugan



FIRST EDITION 2019

Copyright © BPB Publications, INDIA

ISBN: 978-93-88511-13-1

All Rights Reserved. No part of this publication can be stored in a retrieval system or reproduced in any form or by any means without the prior written permission of the publishers

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The Author and Publisher of this book have tried their best to ensure that the programmes, procedures and functions described in the book are correct. However, the author and the publishers make no warranty of any kind, expressed or implied, with regard to these programmes or the documentation contained in the book. The author and publisher shall not be liable in any event of any damages, incidental or consequential, in connection with, or arising out of the furnishing, performance or use of these programmes, procedures and functions. Product name mentioned are used for identification purposes only and may be trademarks of their respective companies.

All trademarks referred to in the book are acknowledged as properties of their respective owners.

Distributors:

BPB PUBLICATIONS

20, Ansari Road, Darya Ganj
New Delhi-110002
Ph: 23254990/23254991

BPB BOOK CENTRE

376 Old Lajpat Rai Market,
Delhi-110006
Ph: 23861747

MICRO MEDIA

Shop No. 5, Mahendra Chambers,
150 DN Rd. Next to Capital Cinema,
V.T. (C.S.T.) Station, MUMBAI-400 001
Ph: 22078296/22078297

DECCAN AGENCIES

4-3-329, Bank Street,
Hyderabad-500195
Ph: 24756967/24756400

Published by Manish Jain for BPB Publications, 20, Ansari Road, Darya Ganj, New Delhi-110002 and Printed by Repro India Pvt Ltd, Mumbai

Preface

While the history of technology development by mankind can be considered in terms of thousands of years, the real development of technology has occurred only during the last hundred years.

What is Artificial Intelligence?

One of the key features that distinguish us, humans, from everything else in the world is intelligence. This capacity to understand, practice knowledge and strengthening skills has played vital role in our evolution and developing human civilization. It is believe that the advancement in technology can create super intelligence that can threaten human existence

What Is Machine Learning?

This is a book about Machine Learning with MATLAB, which immediately begs the question: what is *Machine Learning*? It's a surprisingly hard definition to nail down, especially given how ubiquitous the term has become. Vocal critics have differently released the term as unnecessary label or a simple buzzword that only occurred to salt resumes and hold on to the eye of enthusiastic tech recruiters

Data scientist has been called “the most important job of the 21st century,” presumably by someone who has never visited a fire station.

And developing field and it can't take a great extent of detecting to find analyst breathlessly fore sighting that over the next 10 years, we will need billions and billions of more data scientists than we currently have. An aim is to help and develop the data science by learning algorithm skills and the desire is to develop statistical modeling and the mathematics that is the core of Machine Learning and the goal is to help you to get comfortable with the mathematics and statistics that are at the core of data science.

The best way to learn Machine Learning is by Learning Algorithms on things. By reading this book you will get good understanding of the way the Algorithm has been used for various applications. You will get good understanding of Machine Learning using Matlab and some part like deep learning has been touched with Python approach to get the students and

readers a good comparative analysis about classification and prediction and data visualizations. In the book the content part has been organized in such a way that a graduate and post graduate student can get fundamentals of machine learning along with ample of examples to get conceptualize the theories of different machine learning algorithms. This book has focused right from machine learning basic theories along with pattern recognition, visualization of data, brief introduction in Deep learning and applications of tensor flow as well.

- As the real-time application of machine learning is endless but the basics concepts and algorithms are discussed by us using MATLAB language so that from graduate students to researchers can get benefited with this.
- The book focused on MATLAB code for algorithm implementations rather than mathematical formula.
- The book has discussed machine learning workflow for health monitoring.
- The neural network domain has been touched and implementation in Matlab with explicit explanation of code and results
- This book has ability to realize the students that machine learning is easy and interesting.

Foreward

Who should read this book

The book is basically meant for graduate and research students who find the algorithms of machine learning difficult for implementations. We have touched all basic algorithms of machine learning in detail with the practical approach. Primarily beginners can find this book more effective as the chapters are subdivided into such a way they will find the building and implementing algorithms in MATLAB is interesting and easy at the same time.

Why we wrote this book

The writers for this book teamed up from research and academic research domain, so we take care of things that the text and flow of chapter's content are easy enough for the beginners.

Readership (who's the target audience?):

There are numerous books on machine learning and AI. In any case, every one of them is implied for graduate students or research today, applying machine learning does not require a Ph.D. Nonetheless, there are a couple of assets out there that completely cover all the essential parts of actualizing machine learning by and by, without expecting you to take advanced of math courses. We believe this book will help individuals who need to apply machine learning without studying upon years of analytics, calculus math, and probability hypothesis. We are focusing on the engineering students who find difficulties while solving different machine learning algorithms in MATLAB.

Machine learning is most sought to research field and is an integral part of many research projects today including commercial applications, and academic research as well. The machine learning domain starts from finding friends on social networking sites to medical diagnosis and even for satellite processing. In this book, we have made an honest effort to make the concepts of machine learning easy and also give basics programs in MATLAB right from the installation part. As the real-time application of machine learning is endless but the basics concepts and algorithms are discussed by us using MATLAB language so that from graduate students to researchers can get benefited with this.

What you will learn:

- Machine learning in MATLAB
- The Algorithms of machine learning with MATLAB code
- Deriving and access data in MATLAB then preprocessing and preparation of data
- Machine learning workflow for health monitoring
- Neural network domain implementation in MATLAB with explicit explanation of code and results.

Acknowledgment

Writing a book is harder than I thought and more rewarding than I could have ever imagined. First and foremost, I would like to thank my father Mr. Krishan Dev Pandey for being coolest father ever and my mother Mrs. Veena Pandey for allowing me to follow my ambitions throughout my childhood. They taught me discipline, tough love, manners, respect, and so much more that has helped me succeed in life. Also, my gratitude to my elder sister Mrs. Arpna Tripathi, who always stood by me during every struggle and all my successes. She has been my inspiration and motivation for continuing to improve my knowledge and move my career forward. Also, I'm eternally grateful to my wife Mrs. Kajal Pandey for standing beside me throughout my career and writing this book. I also thank my wonderful son Aarudra Pandey, for always making me smile and for understanding on those weekend mornings when I was writing this book instead of playing games with him. I hope that one day he can read this book and understand why I spent so much time in front of my computer. Last but not the least, I want to thank my friends who always backed me in my good or bad days and everyone who ever said anything positive to me or taught me something. I heard it all, and it meant something.

Abhishek K. Pandey

Assistant Professor (Computer science engineering)
ACERC, Visiting faculty, Mdsu, Ajmer Rajasthan, India

First of all, I would like to thank the authors for contributing their excellent chapters to this book. Without their contributions, this book would not have been possible.

I would like to dedicate this book to my father Late Mr. Raghunath Singh Rathore and my mother Late Mrs. Prem Kanwar who always believed in my ability to be successful. I am missing you and at the same time feeling you both around me always. You are gone but your belief and blessing in me has made this journey possible. Also, my gratitude to my elder brother Mr Praveen Singh Rathore, who always stood by me during every struggle and all my successes. Also, I would like to express appreciation to my

beloved wife Mrs. Anita Kanwar who always support in the moments when there was no one to answer my queries. I also thank my wonderful son Raghavendra Singh Rathore, for made me stronger, better and more fulfilled than I could have ever imagined. This book has been a long-cherished dream of mine which would not have been turned into reality without the support and love of these amazing people, who encouraged me despite my not giving them the proper time and attention. Thanks to all my friends specially Abhishek K Pandey for sharing my happiness at the start of this project and following up with their encouragement when it seemed too difficult to completed.

Pramod Singh Rathore

*Assistant Professor (Computer science engineering)
ACERC, Visiting faculty, Mdsu, Ajmer Rajasthan, India*

The authors are always thankful to God for their perseverance.

I would like to thank my father Mr.M.Shanmugam and mother Mrs.S.Sarojini, wife Mrs.S.Charanyaa for being the pillar of support, son Master.B.Surya for his patience and understanding, and to Mr.K.S.Subramanian and Mrs.S.Varalakshmi for support. I wish to thank my sisters Mrs.S.Amudha and Dr.S.Geetha for their valuable support. My special thanks go to brother-in-law Mr.S.Vivek & Family. Also wishes thanks to the management team of QUANTS IS & CS LLP, India for their support for the book work.

Dr.S.Balamurugan

*Head of Research and Development,
Quants IS & CS, India.*

Authors



Abhishek Kumar Pandey is pursuing his Doctorate in computer science and done M.Tech in Computer Sci. & Engineering. He has been working as an Assistant professor of Computer Science at Aryabhata Engineering College and Research center, Ajmer and also visiting faculty in Government University MDS Ajmer. He has total Academic teaching experience of more than eight years with more than 50 publications in reputed National and International Journals. His research area includes- Artificial intelligence, Image processing, Computer Vision, Data Mining, Machine Learning. He has been in International Conference Committee of many International conferences. He has been the reviewer for IEEE and Inder science Journal. He has authored 4 books published internationally and 7 edited book.. He is also member of various National and International professional societies in the field of engineering & research like Member of IAENG (International Association of Engineers), Associate Member of IRED (Institute of Research Engineers and Doctors), Associate Member of IAIP (International Association of Innovation Professionals), Member of ICSES (International Computer Science and Engineering Society), Life Member of ISRD (International Society for research & Development), Member of ISOC (Internet Society).He has got Sir CV Raman life time achievement national award for 2018 in young researcher and faculty Category. He is serving as an Associate Editor of Global Journal on Innovation, Opportunities and Challenges in Applied Artificial Intelligence and Machine Learning.



Pramod Singh Rathore is pursuing his doctorate in Computer Science & engineering and done M. Tech. He has been working as the Assistant professor of Computer Science at Aryabhata Engineering College and Research centre, Ajmer and visiting faculty in Government University MDS Ajmer. He has been edited and authored many books with Wiley, Taylor & Francis Eureka

group, CRC USA. He has total Academic teaching experience of more than eight years with more than 40 publications as Research papers and Chapters in reputed National and International E-SCI SCOPUS. He has done five edited book. His research area includes machine learning, NS2, Computer Network, Mining, and DBMS. He has been serving in editorial and advisory committee of Global journal group, Eureka Group of Journals. He has been member of various National and International professional societies in the field of engineering & research like Member of IAENG (International Association of Engineers).



Dr S. Balamurugan is the Head of Research and Development, Quants IS & CS, India. Formely, he was the Director of Research and Development at Mindnotix Technologies, India. He has authored/co-authored 33 books and has 200 publications in various international journals and conferences to his credit. He was awarded with Three Post-

Doctoral Degrees- Doctor of Science(D.Sc.) degree and Two Doctor of Letters(D.Litt) degrees for his significant contribution to research and development in Engineering, and is the recipient of the Best Director Award, 2018. His biography is listed in “World Book of Researchers” 2018, Oxford, UK and in “Marquis WHO’S WHO” 2018 issue, New Jersey, USA. He carried out a healthcare consultancy project for VGM Hospitals between 2013 and 2016, and his current research projects include “Women Empowerment using IoT”, “Health-Aware Smart Chair”, “Advanced Brain Simulators for Assisting Physiological Medicine”, “Designing Novel Health Bands” and “IoT -based Devices for Assisting Elderly People”. His professional activities include roles as Associate Editor, editorial board member and/or reviewer for more than 100 international journals and conferences. He has been an invited as Chief Guest/Resource Person/Keynote Plenary Speaker in many reputed Universities and Colleges His research interests include Augmented Reality, the Internet of Things, Big Data Analytics, Cloud Computing, and Wearable Computing. He is a life member of the ACM, ISTE and CSI.

Table of Contents

Preface iii

Foreword v

Acknowledgment vii

Pre-requisite to Machine Learning	1
1. Accessing the Data	1
2. Pre - processing Data	1
3. Deriving Data/ Missing Data in MATLAB	2
4. Importing and Organizing Data	9
4.1 Data Types	9
4.2 Categorical Data Plot	33
4.3 Create and Work with Tables	44
4.4 Cross Validation	46
4.5 What is Data Preparation	51
1. An Introduction to Machine Learning	57
1.1 Basics of Machine Learning.....	58
1.2 Machine Learning Types	58
1.3 Selection of Appropriate Algorithm	63
1.4 Linear Programming Algorithms	65
1.4.1 Machine Learning Workflow using a Health Monitoring.....	67
2. Finding Natural Patterns in Data	73
2.1 Unsupervised Learning	73
2.2 Clustering Strategies	74
2.2.1 Hard Clustering Calculations	74
2.2.2 Soft Clustering Calculations	74
2.3 Cluster Evaluation and Interpretation	91
2.3.1 Common Dimensionality Reduction Techniques for Improving Model Performance	91
3. Building Classification Methods	95
3.1 Supervised Learning.....	95
3.2 Supervised Machine Learning.....	98
3.3 Unsupervised Machine Learning	101

3.4	Semi-supervised Learning.....	103
3.4.1	Understanding Semi-supervised Learning	103
3.5	Reinforcement Learning.....	105
3.6	Some Important Consideration in Machine Learning	106
3.7	Training and Validation	107
3.8	Classification of Methods.....	108
3.8.1	Training of Automated Classifier	108
3.8.2	Manual Classifier Training	109
3.8.3	Parallel Classifier Training.....	110
3.9	Algorithm for Classification.....	112
3.9.1	Classification Algorithm in General.....	112
3.9.2	Common Classification Algorithm.....	112
3.9.3	Regression	138
3.9.4	Regression Algorithms	149
3.10	Techniques for Model Improvement	185
3.10.1	Selecting Features for Classifying High-dimensional Data	185
3.10.2	Loading the Data	186
3.10.3	Sequential Feature Selection Application	191
4.	Data Pre – Processing in Python.....	193
4.1	Data Preparation.....	193
4.1.1	Data Preparation Process.....	193
4.2	Feature Selection for Machine Learning.....	195
4.3	Recursive Feature Elimination.....	196
4.4	Principal Component Analysis.....	197
4.5	Feature Importance.....	198
4.6	Feature Scaling.....	198
4.7	Seven Ways to Handle Large Data Files for Machine Learning	201
4.8	Dimensionality Reduction.....	204
4.9	Cross Validation	205
4.10	Feature Transformation	210
5.	Building Regression Models.....	213
5.1	Parametric regression Methods	213
5.2	Nonparametric Machine Learning Algorithms	215
5.3	Evaluation of Regression Models	217

6. Creating Neural Networks	221
6.1 Self-organizing the Maps and their use in Obtaining K-Clusters	221
6.2 Classification with Feed-Forward Networks.....	223
6.3 Regression with Feed-forward Networks.....	243
7. Introduction to Deep Learning	247
7.1 Deep Learning Overview	247
7.2 How Deep Learning Works.....	247
7.2.1 How is Deep Learning Different from Machine Learning?.....	247
7.2.2 Is Deep Learning Different from AI (artificial intelligence)?	248
7.2.3 What is Deep Learning Framework?	248
7.2.4 What are the Dimensions of the Deep Learning?	248
7.3 Deep Learning uses and Functioning.....	248
7.4 Programming Languages used to Program (design) Deep Learning?	248
7.5 Meaning and importance of Deep Learning.....	249
7.6 What Deep Learning can do in Future?	250
7.7 Applications of Deep Learning in Artificial Intelligence	251
7.8 Fields were deep learning boom:	251
7.9 The future of deep learning	251
7.10 Algorithms in Deep Learning.....	252
7.11 Comparison of Machine Learning and Deep Learning.....	253
7.11.1 Data Dependencies.....	253
7.11.2 Hardware dependencies	254
7.11.3 Execution time	254
7.11.4 Interpretability.....	254
7.12 TensorFlow.....	255
7.12.1 What is TensorFlow	255
7.12.2 Steps to install TensorFlow	255
7.12.3 Linear Regression with TensorFlow.....	256
7.13 Artificial Neural Networks.....	257
7.13.1 Neurons	257
7.13.2 How will Artificial Neural Network Work?	258

7.13.3	Neuron Weights.....	259
7.13.4	Feed-forward Deep Networks.....	259
7.13.5	Feed-forward Deep Networks.....	259
7.14	Activation function.....	260
7.14.1	Back propagation	261
7.14.2	Cost Perform and Gradient Descent.....	262
7.15	Multi-layer perceptron (forward propagation).....	263
7.16	Using Activation Perform	265

Pre-requisite to Machine Learning

1. Accessing the Data

Physical-World Data

MATLAB is used wide range of applications in sensor, picture, video, telemetry, parallel and other continuous organizations.

Machine learning, neural systems, and measurements and beyond:

MATLAB offers a full arrangement of insights and machine learning functionality in addition to cutting edge techniques. For example, nonlinear improvement, framework recognizable proof and a huge number of in-built calculations for picture and video preparing, budgetary displaying, control framework outline.

Rapid preparing of huge data sets

MATLAB's numeric schedules scale straightforwardly for parallel processing on groups and cloud.

2. Pre - processing Data

Information collection can requires pre-processing systems with guaranteed exact, productive or significant investigation. Pre-processing alludes to strategies for discovering, evacuating and supplanting terrible or missing information. Recognizing neighborhood extreme and sudden changes can distinguish huge information patterns. Smoothing and de-trending for expelling commotion and direct patterns from information, while scaling changes the limits of the information. Gathering and binning strategies are procedures that distinguish connections among the information factors.

3. Deriving Data/ Missing Data in MATLAB

Working with missing information is a typical assignment in information pre - processing. In some cases missing qualities imply an important occasion in the information else with inconsistent or unusable information focuses. In either case, MATLAB has numerous alternatives for taking care of missing information.

Make and Organize Missing Data

The shape of missing qualities take in MATLAB relies upon the information write. For instance, numeric information writes such as double use NaN (not a number) deals with missing data.

```
x = [NaN 1 2 3 4];
```

You can likewise utilize the missing value to speak to missing numeric information or different sorts of information. Different information data are such as date time, string and categorical. MATLAB consequently changes over the missing value to the information's for accurate processing.

```
xDouble = [missing 1 2 3 4]
```

```
xDouble = 1x5
```

```
NaN    1    2    3    4
```

```
xDatetime = [missing datetime(2014,1:4,1)]
```

```
xDatetime = 1x5 datetime array
```

```
Columns 1 through 3
```

```
NaT          01-Jan-2014 00:00:00  01-Feb-2014 00:00:00
```

```
Columns 4 through 5
```

```
01-Mar-2014 00:00:00  01-Apr-2014 00:00:00
```

```
xString = [missing "a""b""c""d"]
```

```
xString = 1x5 string array
```

```
<missing>"a""b""c""d"
```

```
xCategorical = [missing categorical ({'cat1' 'cat2' 'cat3' 'cat4'})]
```

```
xCategorical = 1x5 categorical array
```

```
<undefined>    cat1    cat2    cat3    cat4
```

Assume you need to continue missing qualities as a major aspect of your informational collection yet isolate them from whatever remains in the information. A few MATLAB capacities empower you to control the situation of missing qualities for handling additionally. For instance, utilize the 'Missing Placement' option with the sort function to move NaN to a complete number.

```
xSort = sort(xStandard,'MissingPlacement','last')
```

```
xSort = 1×5
```

```
1 2 3 NaN NaN
```

Find, Replace and Ignore Missing Data

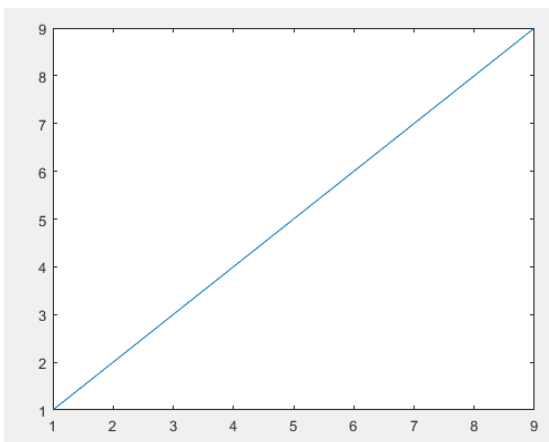
Even though we are not making purposely making missing data value in MATLAB still it brings a kind of information. On the off chance that you don't know about missing qualities in your information this leads to calculation or examination of information.

For instance, that you unknowingly plot a vector containing NaN value, the NaN does not show up on the grounds even plot function based on NaN typically focuses on remaining plot which may be inaccurate.

Program

```
>>nanData = [1:9 NaN];  
plot(1:10,nanData)
```

Output



By the chance you process normal information but still outcome is obtained as NaN. For this situation, it is more useful to know ahead of time information containing NaN function and afterward overlook or expel it before processing as normal information data.

```
>> TF = isnan(nanData)

TF =

     0     0     0     0     0     0     0     0     0     0     1
```

Program

```
>>meanData = mean(nanData)
meanData =
NaN
```

One approach to find NaN in information is utilizing the NaN function, which restores a coherent exhibit showing the area of any NaN value.

Program

```
TFdatetime = ismissing(xDatetime)
TFdatetime = 1x5 logical array
     1     0     0     0     0
```

Assume you are working with a table or timetable made up of factors with numerous information writes. You can discover the greater part of the missing qualities with one call with minimal utilization of time.

```
xTable = table(xDouble',xDatetime',xString',xCategorical')
xTable = 5x4 table
```

	Var1	Var2	Var3	Var4
	NaN	NaN	<missing>	<undefined>
1	01-Jan-2014 00:00:00	"a"	cat1	
2	01-Feb-2014 00:00:00	"b"	cat2	
3	01-Mar-2014 00:00:00	"c"	cat3	
4	01-Apr-2014 00:00:00	"d"	cat4	

```

TF = ismissing(xTable)
TF = 5x4 logical array
    1  1  1  1
    0  0  0  0
    0  0  0  0
    0  0  0  0
    0  0  0  0

```

Missing qualities can speak to unusable information for handling or examination. Use fill missing to supplant missing qualities with esteem or use rm missing to fill missing information data.

```

xFill = fillmissing(xStandard,'constant',0)
xFill = 1x5
    0  1  2  3  0
xRemove = rmmissing(xStandard)
xRemove = 1x3
    1  2  3

```

Numerous MATLAB capacities empower you to overlook missing qualities, without having unequivocally find, fill or expel them first. For instance, in the event you need to register the whole of a vector containing NaN values which yields the outcome as NaN. In any case, you can straightforwardly ignore NaNs in aggregate by utilizing the 'omitnan' option with the sum function.

```

sumNan = sum(xDouble)
sumNan = NaN
sumOmitnan = sum(xDouble,'omitnan')
sumOmitnan = 10

```

Data Splitting Groups and Statistical Calculation

This portrayal shows information of patients in .mat format which are recorded as an social events. This illustrate the mean weight and recorded

figure count then it is converted to strain readings of all patients data. In addition to that resultant outcome of data are plotted based on values in table.

Data regarding patient

Gather data from 100 patients through data sample.

```
>> load patients
```

Conversion of Self Assessed Status of Health in to categorical arrays

```
>> Gender = categorical (Gender);
SelfAssessedHealthStatus = categorical
(SelfAssessedHealthStatus);
whos
```

Name	Size	Bytes	Class
Age	100x1	800	double
Diastolic	100x1	800	double
Gender	100x1	450	categorical
Height	100x1	800	double
LastName	100x1	12416	cell
Location	100x1	15008	cell
SelfAssessedHealthStatus	100x1	696	categorical
Smoker	100x1	100	logical
Systolic	100x1	800	double
TF	1x10	10	logical
Weight	100x1	800	double
meanData	1x1	8	double
nanData	1x10	80	double
x	1x5	40	double

Mean Weight Calculation

Classify the variable of patients as non smoker and smoker variables. Social affair of smoker variable contribution are calculated from mean weight variables.

Program

```
>> [G,smoker] = findgroups (Smoker);
meanWeight = splitapply (@mean,Weight,G)
meanWeight =
149.9091
161.9412
```

Group function is denoted as G, for evaluating number of smokers vector count are calculated. Apply mean function of splitting variables of social event with mean interface of vector weights. For event identification yield the social event as debate variable for event occurrence. Smoker smart attributes variables for smoker in desire qualities. The party identifiers are smart attributes because Smoker contains sensible qualities. As a second variable smoker patients are get-together for evaluating variables.

Program

```
>>smoker
smoker =
  0
  1
```

Mean weight of the patients are classified based on the smoker status and sexual orientation of patients.

```
>> G = findgroups (Gender, Smoker);
meanWeight = splitapply(@mean,Weight,G)

meanWeight =

    130.3250
    130.9231
    180.0385
    181.1429
```

The remarkable mixes across 4 variables based smoker identifier and Gender for patients together classified variables are stated as follows: male nonsmokers & smokers and male nonsmokers & smokers. In the below table consolidated events of mean weight and social events are stated as follows:

```
> [G,gender,smoker] = findgroups (Gender,Smoker);
= table (gender,smoker,meanWeight)
=
```

gender	smoker	meanWeight
Female	false	130.32
Female	true	130.92
Male	false	180.04
Male	true	181.14

Gender group contains clear cut qualities and smoker contains consistent qualities. The information kinds of these table factors coordinate the information about gender of smoker.

```
>>meanBMIfcn = @ (h,w)mean( (w ./ (h.^2) ) * 703);  
BMI = splitapply (meanBMIfcn,Height,Weight,G)
```

```
BMI =
```

```
21.6721  
21.6686  
26.5775  
26.4584
```

Patients BMI (Body Mass Index) are presented based on social occasion of patients. Data conflicts of patients are classified based on height, weight, register BMI and two data conflicts.

Gathering Self-Reports of Patients

Based on the poor or fair classification of patients data are categorized. Regardless, split apply classifies patients in different measures as follows: female nonsmokers & smokers and male nonsmokers & smokers. At this point just check those patients categorized as Poor or fair can be utilizing honest goodness asking for S and G. Through this two approaches of checks, enlist the portion for each social affair.

Program

```
>> [G, gender, smoker] = findgroups (Gender, Smoker);  
S = SelfAssessedHealthStatus ;  
I = ismember (S, {'Poor' , 'Fair' });  
numPatients = splitapply(@numel, S, G);  
numPF = spiltapply(@numel, S(I), G(I);  
numPF ./numPatients
```

```
ans =
```

```
0.2500  
0.3846  
0.3077  
0.1429
```


Diastolic recording of patients are calculated for standard deviation based on Fair health or poor health status of patients, in similar manner health condition of patient are reported in to health condition of Good or Excellent status of patient.

Program

```
>>stdDiastolicPF = splitapply (@std, Diastolic(I),
G(I) );
stdDiastolicGE = splitapply (@std,Diastolic(~I), G(~I)
);
```

In below table consolidated value of patients are stated for patients and recordings of variety of patient health in Fair or Poor health conditions of patients are demonstrated as follow table.

Program

```
>> T = table(gender,smoker,numPatients, numPF,
stdDiastolicPF, stdDiastolicGE, BMI)
T =
gender smoker numPatients numPFstdDiastolicPF stdDiastolicGE BMI
Female false 40 10 6.8872 3.9012 21.672
Female true 13 5 5.4129 5.0409 21.669
Male false 26 8 4.2678 4.8159 26.578
Male true 21 3 5.6862 5.258 26.458
```

4. Importing and Organizing Data

4.1 Data Types

Data type's variables and conversion are stated as characters, tables, cell arrays, structures and conversion of data type and numeric arrays.

Numeric Types

Numeric type considered are data of floating- point and integer type. MATLAB® include numeric class for checked and unsigned numbers for single-precision and twofold exactness of numbers as points. Obviously, MATLAB retrieve data for each and every numeric motivator in twofold precision floating point. In this manner any number are stored and grouped number in single-precision value. Exactness of twofold exactness in single precision of groups memory capable of each number.

Double-precision arrays

```
>> x = 10;
```

```
whos x
```

Name	Size	Bytes	Class	Attributes
X	1x1	8	double	Creation

You can make a twofold exactness exhibit utilizing the [] operator, such as $A = [1\ 2\ 3; 4\ 5\ 6]$. What's more, numerous capacities return twofold accuracy clusters, such as `sin`.

Syntax

```
Y = double(X)
```

Description

`Y = double(X)` converts the values in `X` to double precision

Precision using the `double` function

Integer

Integer Classes

MATLAB® has 4 checked and 4 unsigned entire number classes. Stamped makes enable you to work with negative entire numbers and furthermore positive, yet can't addresses the unsigned number in sorts either positive or negative. In broad numbers unsigned structure are stated in degree of expansion in huge numbers either positive or zero.

In number data MATLAB able to support number bytes of 1, 2, 4 and 8-bytes. You can use For every individual task memory and execution time depends on accumulated data for entire number. Consider the example entire number of 32-bit need not to store as value of 100.

Consider number class count of 8 where every characteristics are stored are different kind and MATLAB try make required changes:

Creation of Data Integer

In MATLAB numeric data are stored as twofold precision drifting point (twofold) as per usual. In order to store number data in MATLAB previously convert the double integer in to desire entire number. Consider the limits change as described in following table.

Consider a example in this we need to store data number of 325 in 16-bit entire number allotted to variable `x`, `typex = int16(325)`; On off chance that the number being changed over to a whole number has a partial part, MATLAB rounds to the closest number. On the off chance that the

fragmentary part is exactly 0.5, at that point from the two similarly adjacent whole numbers, MATLAB picks the one supreme esteem in large amount:

```
x = 325.499;
int16(x)
ans =
int16
    325
x = x + .001;
int16(x)
ans =
int16
    326
```

In case if you want to change the data format of number in default format we have different characteristics to change the function. Fewer functions consider for data conversion are listed as follows: round, fix, floor and ceil. In this fix function engages as default variables then it will be rounded off to zero where their is no nonzero parts are listed.

```
x = 325.9;
int16 (fix (x) )
ans =
int16
    325
```

Math activities that include the two whole numbers and skimming point dependably result in a number information write. MATLAB rounds the outcome, when vital, as per the default adjusting calculation. The case beneath yields a output of 1426.75 in MATLAB at that point round off in following most noteworthy whole number:

```
int16(325) * 4.39
ans =
int16
    1427
```

The whole number change capacities are additionally valuable while changing over different classes, for example, strings, to whole numbers:

```
str = 'Hello World';
int8(str)
ans =
    1×11 int8 row vector
    72 101 108 108 111 32 87 111 114 108 100
```

In the event if you change value of NaN into number class than its resultant outcome is estimated as 0 as in whole number format. Example of this is stated as follows:

```
int32(NaN)
ans =
    int32
    0
```

Integer Classes Arithmetic Operation

MATLAB can perform whole number math on the accompanying kinds of information:

Integers or number varieties of a similar whole number information write. This yields an outcome that has similar information are stated as: `x = uint32([132 347 528]) .* uint32(75);`

- `class(x)`
- `ans =`
`uint32`

In point numbers it can be either whole number or integers based on two folded manner scalar qualities are stated. This resultant point produces same information write as the whole number operands:

- `x = uint32([132 347 528]) .* 75.49;`
- `class(x)`
- `ans =`
`uint32`

Every single paired task in which one operand is a variety of number information compose (with the exception of 64-bit whole numbers) and the other is a scalar twofold, MATLAB figures the activity utilizing component insightful twofold accuracy math and afterward changes over the outcome back to the first number information write. For parallel tasks including a 64-bit whole number exhibit also, a scalar twofold, MATLAB forms the movement just as math type of 80-bit are considered for increasing accuracy rather than minimizing loss.

Integer Class in Larger and Smaller Value

For each entire number data compose, greater and most reliable number can be addressed for number data. For this similar to table format integers are demonstrated in diminutive characteristics for creation of “Extent of Values” area.

The functions `intmax('int8')` is utilized to identify `intmax` and `intmin` qualities.

```
ans =
    int8
    127
intmin('int8')
ans =
    int8
   -128
```

In the event that you change over a number that is bigger than the most extreme estimation of a whole number information write to that compose, MATLAB sets it to the greatest esteem. So also, in the event that you change over a number that is littler than the base estimation of the whole number information writing MATLAB sets it to the base esteem.

Consider following example,

```
x = int8(300)
```

```
x =
```

```
int8
```

```
127
```

```
x = int8(-300)
```

```
x =
```

```
int8
```

```
-128
```

Moreover, when the delayed consequence of a number juggling assignment including entire numbers outperforms the best (or slightest) estimation created data sets, in this MATLAB sets most minimal value for consideration:

```
x = int8(100) * 3
```

```
x =
```

```
int8
```

```
127
```

```
x = int8(-100) * 3
```

```
x =
```

```
int8
```

```
-128
```

Floating Point Number

MATLAB® provides skimming in either of point in twofold single-precision or precision mastermind. In twofold precision; default value is set to be that direct change in work is obtained through single precision value.

Creating Floating-Point Data

Utilize twofold exactness to store esteems more prominent than roughly 3.4×10^{38} or not precisely around -3.4×10^{38} . The number lies between

those two limits are either uses single-precision with utilization of minimal memory. Performance Metrics Analysis for evaluating logistics growth and infrastructure development in India

Double-Precision Data Creation

In MATLAB default double numeric is defined based on the task explanation in two folded manner as below:

```
x = 25.783
```

The whos function demonstrates that in MATLAB type double value are estimated by, one by one in the name of x:

```
whos x
```

Name	Size	Bytes	Class
x	1x1	8	double

Use of float if you essentially need to affirm x as drifting number value. In this limit is consider as 1 for desire drifting point with sensible information 0 as different point function:

```
isfloat(x)
ans =
logical
1
```

This can be converted into other data formats like numeric, strings, character data in two folded manner in MATLAB function. With the precision point delineation changes are evaluated for checking whole number.

```
y = int64(-589324077574);    % Create a 64-bit integer
x = double(y)                % Convert to double
x =
-5.8932e+11
```

Single-Precision Data Creation

In default MATLAB stores numeric data information in default manner for making it sole exact number it should have single capacity conversion number format.

```
x = single(25.783);
```

In MATLAB whos function is utilized to restore structure of variable x characteristics. This structure describes x when it has been put solely and it requires 4 bytes with 8 bytes double values:

```
xAttrib = whos('x');
xAttrib.bytes
ans =
    4
```

You can change other numerical number information either characters or strings and legitimate information for accuracy utilization in single function. Single-accuracy point are marked over changes presented in the numerical point:

```
y = int64 (-589324077574);      % Create a 64-bit integer
x = single(y)                  % Convert to single
x =
    single
   -5.8932e+11
```

Arithmetic Operations over Floating-Point Number in MATLAB

This functional area facilitates class of depicts in which point number tasks are obtained.

Double-Precision Operations

The essential functional performance considered for number-crunching activities class of double or any other form. It must contain at least one whole number for operand in scalar form. Resultant outcome of double integer are listed as below:

- `single` — Resultant in single type function
- `double`
- `int*` or `uint*` — Similar to integer operand it has same data type.
- `char`
- `logical`

The below stated example provides result in double for input arithmetic performance char and double data type.

```
c = 'uppercase' - 32;
class(c)
ans =
    double
char(c)
ans =
UPPERCASE
```

Single-Precision Operations

- You can perform essential number juggling tasks in single or any other different task classes however outcome is obtained as single for any of following characteristics:
- Single class
- Double class
- Char class
- Logical class

Consider as following example in this double class is set as 7.5 default value where the result is single class:

```
x = single([1.32 3.47 5.28]) .* 7.5;
class(x)
ans =
    single
```

Class of Floating Point in Largest and Smallest Values

In single and double classes we can able to write the coding for biggest value integer in MATLAB.

Largest and Smallest Double-Precision Values

In double data type of MATLAB function `realmax` and `realmin` composes minimal and least qualities as follows:

```
str = 'The range for double is:\n\t%g to %g and\n\t %g to %g';  
sprintf(str, -realmax, -realmin, realmin, realmax)  
ans =
```

The range for double is:

```
-1.79769e+308 to -2.22507e-308 and  
2.22507e-308 to 1.79769e+308
```

In MATLAB positive and negative infinity ranges are assigned based on the function `realmax` and `-realmax` function respectively. In this number higher than `realmax` are positive and number lower than `-realmax` are negative values as stated below:

```
realmax + .0001e+308  
ans =  
Inf  
-realmax - .0001e+308  
ans =  
-Inf
```

Single-Precision Largest and Smallest Values

MATLAB function with `realmax` and `realmin` their will be a 'single' argument to single data type of least and extreme data type qualities:

```
str = 'The range for single is:\n\t%g to %g and\n\t %g to %g';
sprintf(str, -realmax('single'), -realmin('single'), ...
    realmin('single'), realmax('single'))
```

```
ans =
```

The range for single is:

```
-3.40282e+38 to -1.17549e-38 and
1.17549e-38 to 3.40282e+38
```

```
realmax('single') + .0001e+038
```

```
ans =
```

```
single
```

```
Inf
```

```
-realmax('single') - .0001e+038
```

```
ans =
```

```
single
```

```
-Inf
```

Accuracy Calculation in Floating-Point Data

In case if the drifting point and estimation are obtained correctly than the limitation lies in the PC's hardware. In results the likelihood values are had lacking bits of address for minimal accuracy range hence it is considered as truncated consequence value.

Double-Precision Accuracy

In two fold precision number it contains set of number values with exact values. In PC their lies a precision range between two higher numbers. For this gap between degrees confines results of precision value with eps function. In following example division of 5 for precision value is stated as below:

```
format long
eps(5)
ans =
      8.881784197001252e-16
```

The above example stated that in two folded manner for division 5 precision range lies between 5 and 5 + eps(5). For proper estimation of precision values of 5 the exact value is calculated with eps(5) function.

In this eps(x) function relies on variable x. Further in below example variable x is greater than MATLAB function eps(x):

```
eps(50)
ans =
      7.105427357601002e-15
```

On the off chance that eps function have desire contention information to estimate eps(1) function in MATLAB for exact number calculation from separation value 1.

Accuracy of Single-Precision

It have certain distance between two single number for precision calculation. Consider x is an integer of single type for that precision count is returned with eps(x) function. For this consider below example,

```
x = single(5);
eps(x)
returns
ans =
      single
      4.7684e-07
```

The outcome is more noteworthy than eps(5). Since less single-accuracy numbers than twofold accuracy numbers, openings between numbers of single value are higher than gap between number precision values. This

enables single-accuracy number minimal precision than that of double - precision value.

For double number stated as x provides aggregate upper header for `eps(single(x))` when it changes from double to single value. In case when precision value exactness obtained as 3.14 in single number than for double value it is calculated as `double(single(3.14) - 3.14)`.

ans =

1.0490e-07

The amount that 3.14 is rounded is less than

`eps(single(3.14))`

ans =

single

2.3842e-07

Floating-Point Arithmetic Common Point Avoidance

All errands function of MATLAB are performed in two folded math precision based on IEEE 754 standard. Since PCs simply address numbers restricted to precision value (twofold exactness related to 52 bits), nonintuitive results are obtained via numerical computation. In MATLAB this are not considered as bug.

Utilize accompanying cases which enable to distinguish cases mentioned below:

Illustration 1 — Round-Off or What You Get Is Not What You Expect

In below example $4/3$ decimal number is nor precision value its in two-folded precision value. Further resultant value is not obtained as 0 instead it is denoted as eps.

`e = 1 - 3*(4/3 - 1)`

e =

2.2204e-16

Similarly, 0.1precisely represents paired number. Along these lines, accompanying non intuitive conduct value is obtained:

22 - *A Practical Approach for Machine Learning and Deep Learning Algorithms*

```
a = 0.0;
for i = 1:10
    a = a + 0.1;
end
a == 1
ans =
    logical
    0
```

In computation order of operation is not possible:

```
b = 1e-16 + 1 - 1e-16;
c = 1e-16 - 1e-16 + 1;
b == c
ans =
    logical
    0
```

Between skimming point numbers their lies holes. With increase in number holes get higher:

```
(2^53 + 1) - 2^53
ans =
    0
```

In this π is actual number, further $\sin(\pi)$ is also not exactly zero:

```
sin(pi)
ans =
    1.224646799147353e-16
```

Example 2 —Cancellation of Catastrophic

Exactly with operands in comparable manner operation of subtraction is performed, in all of sudden cancellation is performed. The going instance for cancellation leads to overpowering (precision loss for insignificant development).

```
sqrt(1e-16 + 1) - 1
```

```
ans =
```

```
0
```

In MATLAB function for sudden cancellation fewer examples are utilized which are `expm1` and `log1p`.

Example 3 — Operation of Floating-Point and Linear Algebra

Round-off value, cancellation and diverse qualities skimming guide math unite toward make startling figuring when dealing with the issues of direct factor based math. MATLAB alerts that the going with matrix A is not very much shaped and along these lines the system $Ax = b$ may be fragile to little irritations:

```
A = diag([2 eps]);
```

```
b = [2; eps];
```

```
y = A\b;
```

Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 1.110223e-16.

These are just a couple of the cases indicating how IEEE coasting point math influences calculations in MATLAB. The calculations performed in IEEE 754 number juggling are influenced; which incorporates applications written in C or FORTRAN and MATLAB

Complex Number

Complex numbers contain two distinct parts as real and imaginary. In this imaginary parts are square root of -1 unit. In MATLAB this real and imaginary parts are denoted by letters `i` or `j`.

With announcement exhibits one technique for making an astounding

an function in MATLAB. In this variable x is defined by many - sided precision value of two part and part 3 as not available value.

```
x = 2 + 3i
```

In MATLAB another approach for dealing with complex number is utilization of complex function. In this limit solidifies numeric commitments of two numbers with marvelous yield, influencing main to include bonafide and the second nonexistent:

```
y = rand(3) * -8;
```

```
z = complex(x, y)
```

```
z =
```

```
4.7842 -1.0921i 0.8648 -1.5931i 1.2616 -2.2753i
```

```
2.6130 -0.0941i 4.8987 -2.3898i 4.3787 -3.7538i
```

```
4.4007 -7.1512i 1.3572 -5.2915i 3.6865 -0.5182i
```

You can isolate a mind boggling number as real and imaginary part by utilizing the function for real:

```
zr = real(z)
```

```
zr =
```

```
4.7842 0.8648 1.2616
```

```
2.6130 4.8987 4.3787
```

```
4.4007 1.3572 3.6865
```

```
zi = imag(z)
```

```
zi =
```

```
-1.0921 -1.5931 -2.2753
```

```
-0.0941 -2.3898 -3.7538
```

```
-7.1512 -5.2915 -0.5182
```

Infinity and NAN

Infinity

MATLAB® represents unprecedented value `inf`. Boundlessness comes to fruition in view of errands like division by zero and surge, which provoke results excessively sweeping, making it impossible to address as general skimming point regards. MATLAB moreover gives a limit based on IEEE arithmetic depiction as `inf` for permeability in positive manner regarded as double-precision.

Couple instance of declarations which leads to positive or negative boundlessness in MATLAB is showed up here.

Use the `isinf` function to affirm that `x` is positive or negative perpetuation:
`x = log(0);`

```
isinf(x)
ans =
    1
```

NaN

MATLAB addresses esteems that are not a complex numbers with a one of a kind regard called NaN, which stays for “Not a Number”. Explanations like `0/0` and `inf/inf` result in NaN, as do any calculating exercises including a NaN:

```
x = 0/0
x =
    NaN
```

You can also create NaNs by:

```
x = NaN;
whos x
```

Name	Size	Bytes Class
x	1x1	8 double

MATLAB function NaN returns depiction of IEEE double scalar value.

For significant NaN value for hexadecimal is stated as follows,

```
format hex
x = NaN
x =
    fff8000000000000
```

To evaluate the array elements of NaN MATLAB always uses isnan function:

```
isnan(x)
ans =
    1
```

MATLAB software represent a "Not a Number" provides current value of NaN representations in alternate manner further it deals with assorted depictions equivalently for NaN. Regardless, in certain uncommon stages (due to gear hindrances), MATLAB does not have right piece case for NaN representation in alternate manner all through estimation rather uses configuration portrayed beforehand bit of NaN.

NaN Logical Operations

Generally, two NaN functions are proportional to each other it deals with lucid exercises for returning false value with involvement of NaN beside a test lopsidedness, (NaN ~= NaN):

```
NaN > NaN
ans =
    0

NaN ~= NaN
ans =
    1
```

Numeric ValueDisplay Format

Default Display

Normally, MATLAB provides numerical value in 5-digit display based on settled points. Numeric characteristics value appeared at going with:

Settled point in 5-digit display as drifting point or with two significant value.

Settled point with 15-digit display with gliding point or two points best value

Minimal whole number proportion

Hexadecimal (base 16)

Bank documentation

Each and every open course of action are recorded in reference page format.

Example of Display Format

In these two or three instances of the distinctive courses of action and the yield conveyed from the going with two-part as x vector, different degrees section are stated as follows:

Current format setting checking:

```
get(0, 'format')
```

```
ans =
```

```
short
```

Previously set value to x and display in 5-digit scaled fixed point:

```
x = [4/3 1.2345e-6]
```

```
x =
```

```
1.3333 0.0000
```

Set the format to 5-digit floating point:

```
format short e
```

```
x
```

```
x =
```

```
1.3333e+00 1.2345e-06
```

Set the format to 15-digit scaled fixed point:

```
format long
```

```
x
```

```
x =
```

```
1.333333333333333 0.000001234500000
```

In small integer ratio format need to set 'rational' output:

```
format rational
```

```
x
```

```
x =
```

```
4/3 1/810045
```

In hexadecimal format integer value is set to x and used for display with base 16:

```
format hex
```

```
x = uint32(876543210)
```

```
x =
```

```
343efcea
```

Program for Numeric Setting Format

To by chance need to modify certain values in program instead of numeric values, get the principle affiliation utilizing similar function addition to variable. The path towards strategy will reestablish the setting utilizing appeared function here:

```
origFormat = get(0, 'format');
format('rational');
-- Work in rational format --
set(0,'format', origFormat);
```

Combining Integer Type

Overview

If you join particular number created system (e.g., set apart in unsigned value or 8-bit entire numbers in 16-bit). Typically, in MATLAB with compose of typical function all structure are segmented. In MATLAB resulting system are set to desire data kind uttermost left segment data cross section. For example, the delayed consequence going with interface three 16-bit vector checked entire numbers:

```
A = [int16(450) uint8(250) int32(1000000)]
```

Example of Combining Integer Sizes

In the wake of impairing the whole number link notices as appeared above, connect the accompanying two numbers once and afterward switch their request. The arrival esteem relies upon the request in which the whole numbers are connected. The furthest left write decides the information compose of vector components:

```
A = [int16(5000) int8(50)]
```

```
A =
```

```
5000 50
```

```
B = [int8(50) int16(5000)]
```

```
B =
```

```
50 127
```

The principle action reestablishes 16-bit number vector. Entire numbers returns 8-bit vector count. In this element are set to 127 with function

`int16(5000)`, the most outrageous motivating force for a 8-bit checked number.

Similar tenets apply to vertical link:

```
C = [int8(50); int16(5000)]
```

```
C =
```

```
50
```

```
127
```

Note

You can locate the most extreme or least qualities for any MATLAB whole number write utilizing the `intmax` and `intmin` functions. Functions like `realmax` and `realmin` are used for point coasting.

Signed with Unsigned Combination

Presently similar exercise is marked and unsigned with whole numbers. Once more, the furthest left component decides the information write for all components in the subsequent network:

```
A = [int8(-100) uint8(100)]
```

```
A =
```

```
-100 100
```

```
B = [uint8(100) int8(-100)]
```

```
B =
```

```
100 0
```

The element `int8(-100)` is set to zero because it is no longer signed.

MATLAB assesses every element concatenating prior into a consolidated cluster. As such, the accompanying proclamation assesses to a 8-bit marked number (equivalent to value 50) and provided with whole number with unsigned number in 8-bit (i.e in this value is set to zero as unsigned value instead of 50) preceding are consolidated as two components. In second component holds esteem however goes up against the unsigned `int8` type:

```
A = [int8(50), uint8(-50)]
```

```
A =
```

```
50  0
```

Combining Integer and Non Integer Data

In the event that you join numbers with double, single, or logical classes, all components have subsequent lattice provided with information kind furthest left whole number. In certain instance, components accompanying are set to int32 vector:

```
A = [true pi int32(1000000) single(17.32) uint8(250)]
```

Empty Matrices

In the event that you build a framework of matrix element matrix, where void lattices overlooked as subsequent grid:

```
A = [5.36; 7.01; []; 9.44]
```

```
A =
```

```
5.3600
```

```
7.0100
```

```
9.4400
```

Concatenation Examples

Single and double type combination

Combination of single and double value matrix provide double value. In this 5.73×10^{300} is excessively tremendous, making it impossible secured as single function, (In this the function class is used as a piece of this case reestablishes the data compose for the data regard).

```
x = [single(4.5) single(-2.8) pi 5.73*10^300]
x =
    4.5000 -2.8000  3.1416   Inf
class(x)      % Display the data type of x
ans =
    single
```

Double and Integer type Combination

Joining entire number characteristics with double values yields a number cross section. Note that the halfway part of pi is changed in accordance with the nearest entire number. (The int8 function used as a piece of this case changes over its numeric conflict to a 8-bit number).

```
x = [int8(21) int8(-22) int8(23) pi 45/6]
x =
    21 -22  23  3  8
class(x)
ans =
    int8
```

Double and Character type combination

To retain character matrix values of double and character need to be combined.

In this scenario MATLAB values are changed to double element value to equivalent character values:


```
x = ['A' 'B' 'C' 68 69 70]
```

```
x =
```

```
ABCDEF
```

```
class(x)
```

```
ans =
```

```
char
```

Double and Logical type Combination

Double matrix is obtained through combination of double and logical value matrices. MATLAB to delineation logical value need to change in true and false element:

```
x = [true false false pi sqrt(7)]
```

```
x =
```

```
1.0000    0    0  3.1416  2.6458
```

```
class(x)
```

```
ans =
```

```
double
```

4.2 Categorical Data Plot

Try below example given in MATLAB

Categorical Array data plot as example

Sample Data Load

Data of 100 patients to load sample data

Program

```
>> load patients
```

```
Whos
```

Name	Size	Bytes	Class	Attributes
Age	100x1	800	double	Diastolic
		100x1	800	double
Gender	100x1	12212	cell	

34 - A Practical Approach for Machine Learning and Deep Learning Algorithms

```
Height      100x1    800    double
LastName100x112416    cell
Location 100x115008    cell
SelfAssessedHealthStatus      100x1    12340    cell
Smoker      100X1      100      logical
Syatolic100x1      800      double
Weight      100x1      800      double
X            1x1       8        double
```

Creation of Character Vectors through Categorical Arrays from Cell Arrays

In workplace of vector variable and location of cell are presented with vector character with three of kind restorative offices where patients were watched. To access and think about information all the more effortlessly, convert Location to an absolute exhibit.

Program

```
>> Location = categorical (Location);
>>summary (Location)
      County General Hospital      39
      St. Mary's Medical Center   24
      VA Hospital                 37
>> SelfAssessedHealthStatus = categorical
(SelfAssesedHealthStatus,...
    {'Poor' 'Fair' 'Good' 'Excellent'}, 'Ordinal',
true) ;
>> summary (SelfAssessedHealthStatus)
      Poor      11
      Fair      15
      Good      40
      Excellent 34
>>figure
histogram (SelfAssessedHealthStatus) |
title ('Self Assessed Health Status From 100
Patients')
```

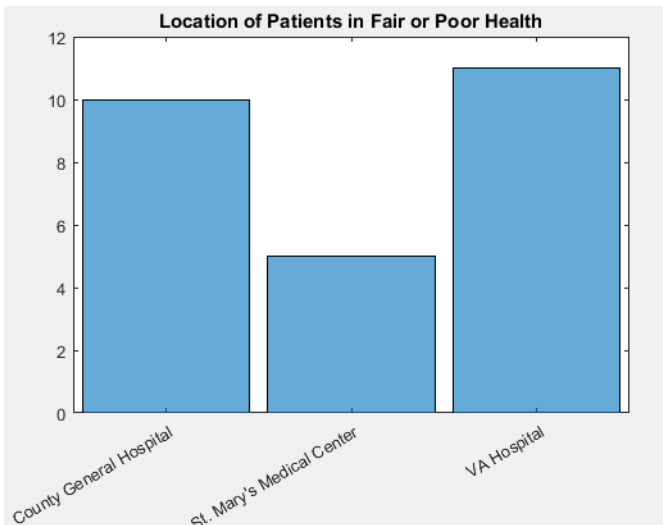
Output



Program

```
>>figure  
histogram(Location(SelfAssessedHealthStatus<='Fair'))  
title('Location of Patients in Fair or Poor Health')
```

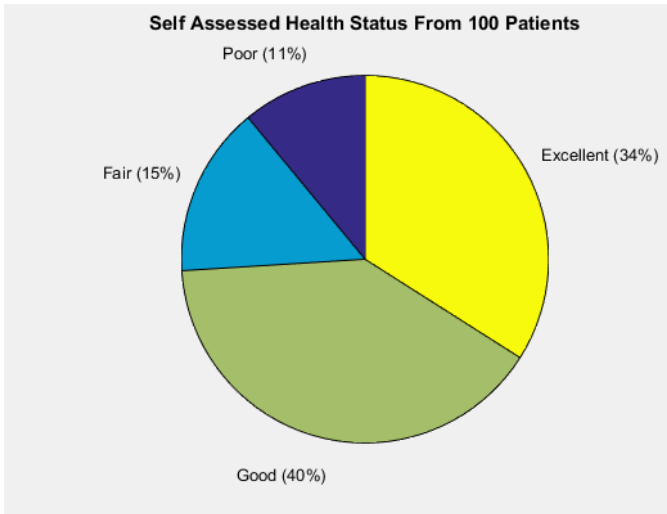
Output



Program

```
>>figure  
pie (SelfAssessedHealthStatus) ;  
title ('Self Assessed Health Status From 100  
Patients')
```

Output



Program

```
>>figure  
A = countcats (SelfAssessedHealthStatus);  
C = categories (SelfAssessedHealthStatus);  
pareto (A,C) ;  
title ('Self Assessed Health Status From 100  
Patients')
```

Output

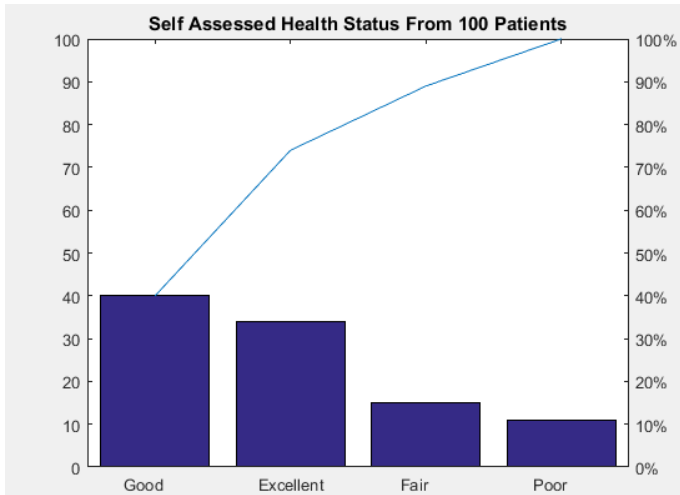


Table Data Access

Table indexing ways

A table is a compartment for securing area sorted out factors that have a relative number of portions. Separated zones engage pick subset information table with secure table compartment. Wavy sponsorships and spot asking for enable you to oust information from a table.

On the off chance that you utilize wavy support, the subsequent gathering is the level association of the predefined table component containing just the predestined lines. The information sorts of all the predefined factors must be helpful for affiliation. You would then have the ability to perform computations utilizing MATLAB functions.

Contact asking for expels information from one table variable. The outcome is a grouping of tantamount information shape as segregated variable. You can take after the spot asking for with areas to choose lines subsets in variables. In an array horizontal variables are interface with every variable in table. Consider variable T is proportional to $T\{:,:\}$.

Accessing Specific Rows

While asking for into a table with areas, wavy sponsorships, or spot asking for colon in particular rows, recorded numeric values or keen articulation.

Other than this we can record through solitary fragment of cell in names as names of lines.

In savvy articulation provides wavy support or spot asking expel information from fragmented subset portray. Consider a example where rows = T with $\text{var2} > 0$ with gathering sensible information of true logical lines with help of Var2 is more evident-able than zero.

Specify Variable Accessing

While asking for into a table with areas, wavy sponsor ship or spot asking for numeric records, colon or other variables. Moreover, with a solitary area cell are assigned with name of area and line.

A shrewd verbalization can contain wavy support or spot asking for expel information to depict the fragmented subsets. Consider a example in which value of row is equal to T with $\text{Var2} > 0$ with logical pack of sensible information with assigned value of zero as (varindex).

Larger Table creation from subset

Try following example

This example describes creation of larger table from subset.

Load Sample Data

Consider example of patient's information and make a table. Utilize the one of a kind identifiers in LastName as push names

Program

```
>> load patients
patients = table (Age, Gender, Height, Weight,
Smoker,...
    'RowNames', LastName);
```

The table, patients, contains 100 lines and 5 factors.

View information composes depiction, units and other clear insights for every factor by using summary to outline the table.

```
>>summary (patients)
```

Program

```

Variables:

  Age: 100x1 double
    Values:
      min      25
      median   39
      max      50

  Gender: 100x1 cell string

  Height: 100x1 double
    Values:
      min      60
      median   67
      max      72

  Weight: 100x1 double
    Values:
      min      111
      median   142.5
      max      202

  Smoker: 100x1 logical
    Values:
      true     34
      false    66

```

Numeric Indices Index

Make a sub-table containing the initial five lines and every one of the factors of patient in tabular form. Utilize ordering of numeric values inside enclosures for indicating coveted columns and factors. This is like ordering with numeric exhibits.

```

>> T1 = patients(1:5,:)
T1=5x5 table

```

T1 =

	Age	Gender	Height	Weight	Smoker
Smith	38	'Male'	71	176	true
Johnson	43	'Male'	69	163	false
Williams	38	'Female'	64	131	false
Jones	40	'Female'	67	133	false
Brown	49	'Female'	64	119	false

Notwithstanding files with numeric values can able to utilize column or name of variables within enclosures. This situation enable utilizing line files and colon with more conservative through utilizing line or name of variables.

Utilization of Index Names

Choose every one of information of patients name containing 'Adams' and 'Brown'. In this it provides minimal complex utilize names in column than numeric record utilization.

```
>> T2 = patients({'Adams', 'Brown'},)
T2=2x5 table
```

	Age	Gender	Height	Weight	Smoker
Adams	48	'Female'	66	137	false
Brown	49	'Female'	64	119	false

T2 provide 2 * 5 table

Logical Expression Indexing

Make another table, T3, containing the sexual orientation, tallness and weight of the patients younger than 30. Select just the columns where the incentive in the variable, Age is below 30.

Program

```
>>rows = patients.Age<30;
Vars = {'Gender', 'Height', 'Weight'};
```


Utilize speck documentation to extricate information from a table variable and an intelligent articulation to characterize the subset of columns in view of that removed information.

Rows are a 100-by-1 consistent exhibit containing logical true (1) for columns where the incentive in the variable, Age, is below 30.

Utilize brackets to restore a table containing the coveted subset of the information.

```
>> T3 = patients (rows, vars)
T3=15x3 table
```

T3 =

	Gender	Height	Weight
	-----	-----	-----
Moore	'Male'	68	183
Jackson	'Male'	71	174
Garcia	'Female'	69	131
Walker	'Female'	65	123
Hall	'Male'	70	189
Young	'Female'	63	114
Hill	'Female'	64	138
Rivera	'Female'	63	130
Cooper	'Female'	65	127
Cox	'Female'	66	111
Howard	'Female'	68	134
James	'Male'	66	186
Jenkins	'Male'	69	189
Perry	'Female'	64	120
Alexander	'Male'	69	171

T3 is a 15-by 3 table.

Create Array from the Contents of Table

Attempt This Example

This case demonstrates to separate the substance of a table utilizing wavy props or dab ordering.

Load Sample Data

Load the example patient's information and make a table. Utilize one of a kind identifiers in LastName as push names

Program

```
>> load patients
patients = table(Age, Gender, Height, Weight, Smoker,...
    'RowNames', LastName);
\\ |
```

The table, patients, contains 100 rows and 5 variables.

Extract Multiple Rows and Multiple Variables

Concentrate information from numerous factors in the table, patients by utilizing wavy supports. Since dab ordering extricates information from a solitary variable at any given moment, supports are more advantageous when you need to separate in excess of one variable.

Concentrate the stature and weight for the initial five patients. Utilize numeric records to choose the subset of rows, 1:5 and variable names to choose the subset of variables, {Height,Weight}

```
>> A = patients{1:5, {'Height', 'Weight'}}
A = 5x2
```

```
A =
    71    176
    69    163
    64    131
    67    133
    64    119
```

A provides numeric array value of 5 * 2

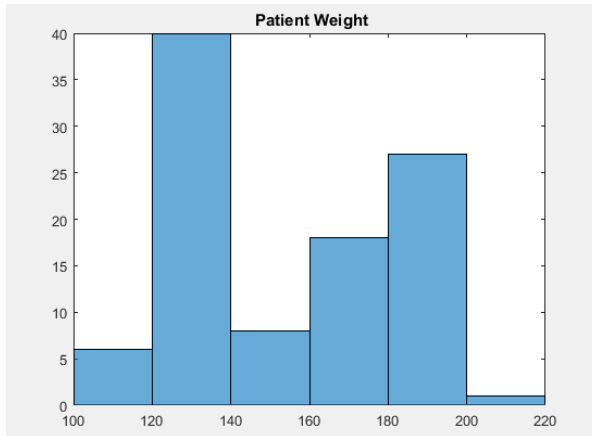
One Variable Data Extraction

For extraction of single variable use dot in indexing. Histogram plot with variable weight, numeric values provides plot.

Program

```
>>figure ( )
histogram(patients.Weight)
title(' Patient Weight')
```

Output



Weight of patients is a twofold accuracy segment vector with 100 lines. On the other hand, you can utilize wavy braces, `patients{:,'Weight'}`, to extricate every one of the lines of variable weight.

To indicate a subset of columns for a solitary variable, you can take after the speck ordering with brackets or wavy props. Concentrate the statures of the nonsmoker patients younger than 30.

Utilize speck documentation to remove information from table factors and a consistent articulation to characterize the subset of lines in view of that extricated information.

```
>>rows = patients. Smoker= =false &patients.Age<30;
```

Use dot notation to extract the desired rows from the variable, Height.

```
>> patients.Height (rows)
```

```
ans = 11×1
ans =
    68
    71
    70
    63
    64
    63
    65
    66
    68
    66
    64
```

The output is a 11-by-1 numeric array. Alternatively, you can specify the single variable, Height, within curly braces to extract the desired data, `patients{rows,'Height'}`.

4.3 Create and Work with Tables

Attempt This Example

This illustration demonstrates to make a table from workspace factors, work with table information and compose tables to documents for later use. Table is an information write for gathering heterogeneous information and meta data properties, for example, factor names, push names, depictions and variable units, in a solitary holder.

Tables are appropriate for segment arranged or unthinkable information that are regularly put away as sections in a content document or in a spreadsheet. Every factor in a table can have an alternate information write, yet should have a similar number of columns. In any case, factors in a table are not limited to segment vectors. For instance, a table variable can contain a grid with different segments as long as it has an indistinguishable number of lines from the other table factors. An average use for a table is to store test information, where lines speak to various perceptions and segments speak to various estimated factors.

Tables are advantageous holders for gathering and sorting out related information factors and for survey and abridging information. For instance, you can separate factors to perform counts and advantageously include the outcomes as new table factors. When you complete your estimations,

compose the table to a record to spare your outcomes.

View Table Creation

Make a table from workspace factors and view it. On the other hand, utilize the Import Tool or the read table function to make a table from a spreadsheet or a content document. When you import information from a record utilizing these capacities, every section turns into variable table.

Test information of patient details with 100 are documented in MAT with factor workspace.

Program

```
>> load patients
whos
```

Name	Size	Bytes	Class	Attributes
A	5x2	80	double	
Age	100x1	800	double	
Diastolic	100x1	800	double	
Gender	100x1	12212	cell	
Height	100x1	800	double	
LastName	100x1	12416	cell	
Location	100x1	15008	cell	
SelfAssessedHealthStatus	100x1	12340	cell	
Smoker	100x1	100	logical	
Systolic	100x1	800	double	
T1	5x5	3785	table	
T2	2x5	2970	table	
T3	15x3	5886	table	
Weight	100x1	800	double	
ans	11x1	88	double	
patients	100x5	29556	table	
rows	100x1	100	logical	
vars	1x3	372	cell	

In table section populate arranged factors with tolerant information. In this to and allocate factors by name in table. When you appoint a table variable from a workspace variable, you can allocate the table variable an alternate name.

Make a table and populate it with the Gender, Smoker, Height and Weight workspace factors. Show the initial five columns.

Program

```
>> T = table (Gender, Smoker, Height, Weight) ;
T(1:5, :)
ans=5x4 table
```

Output

```
ans =
```

Gender	Smoker	Height	Weight
'Male'	true	71	176
'Male'	false	69	163
'Female'	false	64	131
'Female'	false	67	133
'Female'	false	64	119

As an option, utilize the `readtable` function to peruse the patient information from a comma-delimited file. `Readtable` reads every one of the segments that are in a record.

Make a table by perusing all segments from the file, `patients.dat`.

```
>> T2 = readtable('patients.dat');
T2 (1:5, :)
ans=5x10 table
```

LastName	Gender	Age	Location	Height	Weight	Smoker	Systolic	Diastolic	SelfAssessedHealthStatus
'Smith'	'Male'	38	'County General Hospital'	71	176	1	124	93	'Excellent'
'Johnson'	'Male'	43	'VA Hospital'	69	163	0	109	77	'Fair'
'Williams'	'Female'	38	'St. Mary's Medical Center'	64	131	0	125	83	'Good'
'Jones'	'Female'	40	'VA Hospital'	67	133	0	117	75	'Fair'
'Brown'	'Female'	49	'County General Hospital'	64	119	0	122	80	'Good'

You can relegate more segment situated table factors utilizing spot notation, `T.varname`, where `T` is the table and `varname` is the coveted variable name. Make identifiers that are arbitrary numbers. At that point dole out them to a table variable and name the table variable ID. Every one of the factors you allocate to a table must have a similar number of lines. Show the initial five lines of `T`.

4.4 Cross Validation

To show signs of improvement feeling of the prescient precision of your tree for new information, cross approve the tree. Of course, cross approval parts the preparation information into 10 sections indiscriminately. It

trains 10 new trees, everyone on nine sections of the information. It at that point looks at the prescient precision of each new tree on the information excluded in preparing that tree. This technique gives a decent gauge of the prescient precision of the subsequent tree, since it tests the new trees on new information.

Cross Validate a Regression Tree

This case demonstrates to inspect the re-substitution and cross-approval precision of a relapse tree for anticipating mileage in view of the carsmall data.

Load the carsmall data set. Think about increasing speed, dislodging, strength and weight as indicators of MPG.

Program

```
>> load carsmall
X = [Acceleration Displacement Horsepower Weight] ;
Grow a regression tree using all of the observations.
>>rtree = fitrtree(X,MPG) ;
Compute the in-sample error.
>>resuberror = resubLoss(rtree)
resuberror =
    4.7188
```

The re-substitution misfortune for a relapse tree is the mean-squared blunder. The subsequent esteem demonstrates that a normal prescient blunder for the tree is about the square base of 4.7, or somewhat more than 2.

Gauge the cross-approval MSE.

Program

```
>>rng 'default' ;
cvrtree = crossval (rtree) ;
cvloss = kfoldLoss(cvrtree)
cvloss =
    23.8065
```

The cross-approved misfortune is just about 25, which means a run of the mill prescient mistake for the tree on new information is around 5. This shows cross-approved misfortune is typically higher than basic re-substitution misfortune.

A few random things

Get the size of an object with 'size'. Take an optional argument to specify the dimensions (without, it returns an array with the sizes of all dimensions).

Example: `A = [1 2 3 4 5 6]`

`Size(A,1) = 2`

`Size(A,2) = 3`

`Size(A) = [2 3]`

`repmat`

Use 'repmat' to build new matrices from smaller ones.

Example: `x = [1;2]`

`repmat(x,1,3) = [1 1 1 2 2 2]`

`repmat(x,2,2) = [1 1 2 2 1 1 2 2]`

Used mostly to 'vectorize' things

We are frequently managing heaps of information and typically that information is given as genuine vectors. If not, we can regularly knead it as needs be... (Performing highlight extraction)

It is helpful to utilize networks to store information, for instance by stacking the vectors segment astute:

`X = (x1x2 ...xN)`

To display up to 3-d data given in this form,

you could use `scatter(X(1,:),X(2,:)) % display 2-d dataset`

`scatter3(X(1,:),X(2,:),X(3,:)) % display 3-d dataset`

Access Data in Cell Array

This example shows how to read and write data to and from a cell array.

Create a 2-by-3 cell array of text and numeric data.

Program

```
>> C = { 'one', 'two', 'three' ;
1, 2, 3}
C =
    'one'    'two'    'three'
    [ 1]    [ 2]    [ 3]
```

There are two approaches to allude to the components of a cell exhibit. Encase files in smooth parentheses, (), to allude to sets of cells- - for instance, to characterize a subset of the cluster. Encase lists in wavy braces, {}, to allude to the content, numbers, or other information inside individual cells.

Cell Indexing with Smooth Parentheses, ()

Program

```
>> C (1, 1:3) = {'first', 'second', 'third' }
C =
    'first'    'second'    'third'
    [ 1] [ 2] [ 3]
```

If cells in your array contain numeric data, you can convert the cells to a numeric array using the cell2mat function.

numericCells is a 1-by-3 cell array, but numericVector is a 1-by-3 array of type double.

Program

```
>> numericCells = C (2, 1:3)
numericCells =
    [1]    [2]    [3]
>> numericVector = cell2mat (numericCells)
numericVector =
    1    2    3
>>
```

Content Indexing with Curly Braces, {}

Access the contents of cells--the numbers, text, or other data within the cells--by indexing with curly braces. For example, to access the contents of the last cell of C, use curly braces.

```
>>last = C{2,3}
last =
     3
```

You can index with curly braces to replace the contents of a cell.

```
>>C{2,3} = 300
C =
    'first'          'second'          'third'
    [ 1]           [ 2]           [ 300]
```

You can get to the substance of numerous cells by ordering with wavy supports. MATLAB restores the substance of the cells as a comma-isolated rundown. Since every cell can contain an alternate kind of information, you can't dole out this rundown to a solitary variable. Be that as it may, you can dole out the rundown to an indistinguishable number of factors from cells. MATLAB allocates to the factors in section arrange.

Allot substance of four cells of C to four factors.

```
>> [r1c1, r2c1, r1c2, r2c2] = C {1:2, 1:2}
r1c1 =
first

r2c1 =
     1

r1c2 =
second

r2c2 =
     2
```

If each cell contains the same type of data, you can create a single variable by applying the array concatenation operator, [], to the comma-separated list.

Concatenate the contents of the second row into a numeric array.

Program

```
>>nums = [C{2, :}]
nums =
1         2         300
```

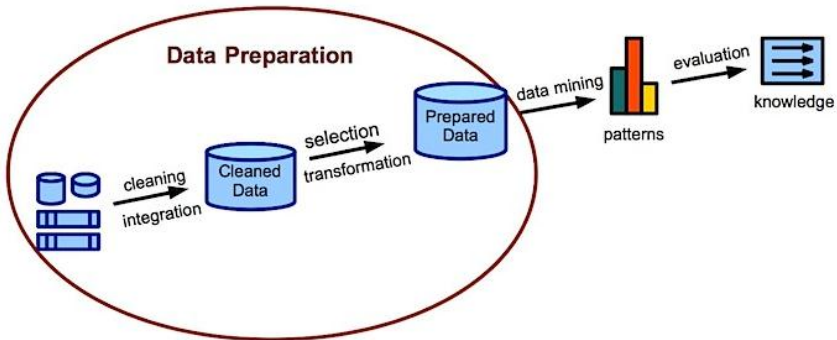
Access Data

Use MATLAB data stores to access data that normally does not fit into the memory of a single computer. Data stores support a variety of data types and storage systems



4.5 What is Data Preparation

Data preparation (or data preprocessing) in this setting suggests control of data into a shape proper for energize examination and taking care of. It is a system that incorporates an extensive variety of assignments and which can't be totally modernized. An impressive parcel of the data arranging practices are typical, repetitive and time consuming. It has been assessed that data plan speaks to 60%-80% of the time spent on a data mining wander.



Data planning is fundamental for effective information mining. Low quality information ordinarily result in mistaken and temperamental information mining comes about.

Data readiness enhances the nature of information and therefore enhances the nature of information mining comes about. The outstanding saying “trash in refuse out” is exceptionally important to this space.

Our point is to create instruments to encourage the undertakings of information arrangement.

Representing Data in MATLAB Workspace

Time-Domain Data Representation

Time-domain data comprises of at least one info variables $u(t)$ and at least one yield variables $y(t)$, examined as a component of time. On the off chance that there is no info variable, see Time-Series Data Representation.

Text Data Preparation

Import text data into MATLAB and preprocess it for analysis.

Functions

extractFiletext	Read text from PDF, Microsoft Word, HTML and plain text files
extractHTMLtext	Extract text from HTML
readPDFFormData	Read data from PDF forms
writeTextDocument	Write documents to text file

`extractFileText`

Read text from PDF, Microsoft Word, HTML and plain text files

Syntax

```
str = extractFileText(filename)

str = extractFileText(filename,Name,Value)
```

Extract Text Data from Text Files

Examples

Extract the text from sonnets.txt using extractFileText.

The file sonnets.txt contains Shakespeare's sonnets in plain text

```
str = extractFileText("sonnets.txt");
i = strfind(str,"I");
ii = strfind(str,"II");
start = i(1);
fin = ii(1);
extractBetween(str,start,fin-1)

ans =

'I
```

From fairest creatures we desire increase,
 That thereby beauty's rose might never die,
 But as the ripper should by time decease,
 His tender heir might bear his memory:
 But thou, contracted to thine own bright eyes,
 Feed'st thy light's flame with self-substantial fuel,
 Making a famine where abundance lies,
 Thy self thy foe, to thy sweet self too cruel:
 Thou that art now the world's fresh ornament,
 And only herald to the gaudy spring,
 Within thine own bud buriest thy content,
 And tender churl mak'st waste in niggarding:

Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

Extract Text Data from PDF

Extract the text from exampleSonnets.pdf using extractFileText.

The file exampleSonnets.pdf contains Shakespeare's sonnets in a PDF file

```
str = extractFileText("exampleSonnets.pdf");  
ii = strfind(str,"II");  
iii = strfind(str,"III");  
start = ii(1);  
fin = iii(1);  
extractBetween(str,start,fin-1)  
ans =  
"II"
```

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.

Preprocessing

It incorporates instruments for handling crude content from sources, for example, hardware logs, news encourages, studies, administrator reports and online networking. Utilize these instruments to remove content from well known record groups, preprocess crude content, separate individual words, change over content into numerical portrayals and manufacture factual models.

Regexprep

Replace text in words of documents using regular expression

Syntax

```
newDocuments = regexprep(documents,expression,replace)
```

Replace words that begin with “s”, end “e” and have at least one character between them. To match whole words, use “^” to match the start of a word and “\$” to match the end of the word.

```
documents = tokenizedDocument([ ...
    "an example of a short sentence"
    "a second short sentence"])
documents =
2x1 tokenizedDocument:
```

```
(1,1) 6 tokens: an example of a short sentence
```

```
(2,1) 4 tokens: a second short sentence
```

Replace

Find and replace substrings in documents

Syntax

```
newDocuments = replace(documents,old,new)
```

Replace Substrings in Documents

Replace words in a document array.

```
documents = tokenizedDocument([
    "an extreme example"
    "another extreme example"])
documents =
2x1 tokenizedDocument:
(1,1) 3 tokens: an extreme example
(2,1) 3 tokens: another extreme example
```

TALL

Create tall array

Syntax

```
t = tall(ds)
t = tall(A)
```

Make an information store for the `airlinesmall.csv` data set. Treat 'NA' values as missing information so they are supplanted with NaN values. Select a little subset of the factors to work with.

Program

```
>>varnames = {'ArrDelay', 'DepDelay', 'Origin',
    'Dest'} ;
ds = datastore ('airlinesmall.csv', 'TreatAsMissing',
    'NA', ...
    'SelectedVariableNames', varnames) ;
```

Use `tall` to make a tall exhibit for the information in the data store. Since the information in `ds` is unthinkable, the outcome is a tall table. In the event that the information isn't unthinkable, then `tall` creates a tall cell exhibit.

CHAPTER 1

An Introduction to Machine Learning

Machine learning belongs to a sub class field of a class of **Artificial Intelligence (AI)**. Machine learning performs effectively by considering an effective information structure in order to appropriately fit the information in models for individual comprehension.

In the field of software engineering, machine learning algorithms exhibit a desire performance due to computational methodologies. PC issues which are not customized are usually calculated customary with ascertain factors. In these cases, machine learning performs a calculation in computers by considering factual examination for estimating a particular information range. Further to this machine learning motivates the for input information processing machine learning encourages test information in robotize of information inputs.

Many innovative techniques have been derived through machine learning, including web-based social networking that provides tagging, photographs, and social networking factors. On the other hand, the optical character acknowledgment (OCR) facilitates changes in the pictures of mobile. General techniques derived from machine learning enable motion pictures for TV shows based on user inclinations. To make ease off buyers, self-driving autos are explored for figuring out machine information factors.

The field of machine learning exhibits a constant progress for desired performance with desired field creation. General classification of machine learning techniques offer an ordered part undertaking. This classification in machine learning is accomplished by the framework that is adopted and created in the network for machine learning.

Most commonly adopted techniques by large grasped machine learning procedures—supervised learning, trains counts in light of representation data and yield data that is set apart by individuals. Unsupervised learning

does not provide the computation with named data wherein remembering the true objective to empower it is to find structure inside its data.

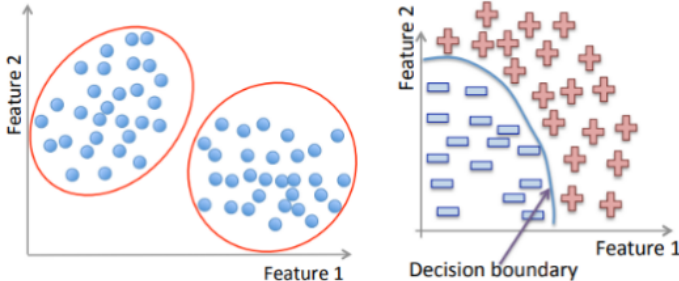
1.1 Basics of Machine Learning

Machine learning has been around for a long time now and every single social media client, have at some point in time been customers of machine learning innovation. One of the basic illustrations is confront acknowledgment programming, which is the ability to distinguish whether a computerized photo incorporates a given individual.

Definition

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

Basic Concepts



1.2 Machine Learning Types

Machine learning is of two types:

- Supervised learning
- Un-supervised learning

Supervised Learning

Machine learning algorithm has a major functionality of utilization of learning techniques through machines. In supervised approach, we are able to get the output variable data (Y) through the input variable (X) for

learning mapping function.

$$Y = f(X)$$

The primary objective here is to map function approximation for newly defined input data (X) and for output data prediction variables (Y) for a given data.

Supervised learning is obtained through training a dataset which is similar to the learning process via supervision of a teacher. We are already aware of the correct answer, through iterative prediction of training the data and rectify the process through teaching. In this learning, we will be stopped when desired acceptable performance is achieved in the machine.

Even supervised learning is subject to classification and regression as major problems for performance measurement. Let's now understand classification and regression in more detail:

- **Classification:** For this particular problem category, the output variable is classified as disease, no disease, blue, or red.
- **Regression:** When the output variable is obtained as a real value, regression problem will occur. For example: weight or dollars.

A few problems that are built with classification and regression require time series prediction and recommendation respectively. The most popular supervised machine learning algorithms are stated with problem solutions:

- Linear regression to solve regression problems
- Random forest to solve regression and classification problems
- Support vector machines to solve classification problems

Unsupervised Learning

If the data only consists of input data (X) without any output variables corresponding to the input variables, then, it belongs to unsupervised machine learning algorithm.

The primary goal of unsupervised learning is data distribution with underlying structure for more understanding of data.

This kind of data processing is known as unsupervised learning since it does not have the correct answers or a teacher for learning as in the case of supervised learning. In this, unsupervised learning algorithms are left in own devices for discovering and structure intersecting in the data.

Similar to supervised learning unsupervised learning are also grouped as association and clustering problems, let's understand what these are:

- **Clustering:** Clustering problem arises for discovering data with inherent groups similar to purchasing behavior of customer groups.
- **Association:** Learning problem with association rule is to discover rules for data at a larger portion just like people tend to buy the variable product X and also buy variable Y.

A few examples for unsupervised learning algorithm are:

- Clustering problem is resolved with the development of k-means
- To resolve association rule learning problem Apriori algorithm is developed

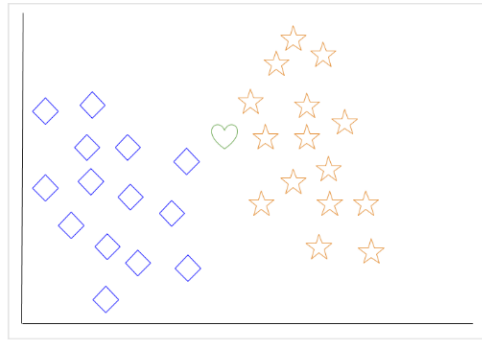
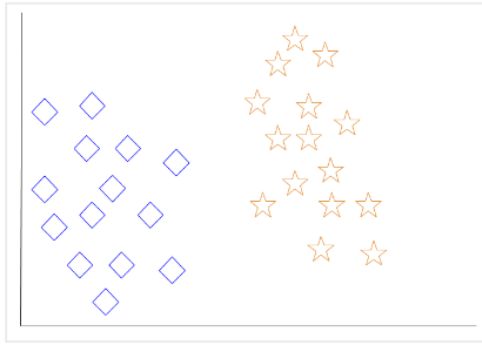
Approaches

Machine learning is immovably related to computational bits of knowledge. Hence, having an established data in estimations is significant for cognizance and using machine learning algorithms. It can be helpful to first describe association and backslide as they are routinely used methodologies for investigating the relationship among quantitative variables. Correlation is a measure of connection between two factors that are not allocated as either poor or independent. Regression at a key level is used to have a glance at the association between one dependent and one self-ruling variable. Since backslide bits of knowledge can be used to presume the dependent variable when the self-ruling variable is known, backslide engages desire capacities.

K-nearest neighbor

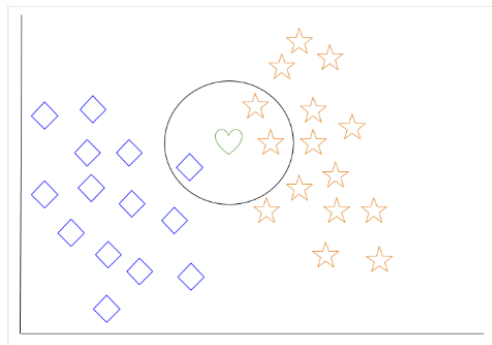
The K-nearest neighbor calculation is an example acknowledgment show that can be utilized for arrangement and in addition relapse. Frequently curtailed as k-NN, the k in k-nearest neighbor is a positive whole number, which is normally small. In either order or relapse, the information will comprise of the k-nearest preparing cases inside a space.

We will center around k-NN grouping. In this strategy, the yield is class participation. This will lead to another question—which class is the most basic one amongst its k-nearest neighbors? On account of $k = 1$, the question is doled out to the class of the single closest neighbor.



When we pick $k = 3$, the calculation will locate the three closest neighbors of the green heart to group it to either the precious stone class or the star class.

In our outline, the three closest neighbors of the green heart are one jewel and two stars. Therefore, the calculation will arrange the heart with the star class.



Among the fundamental machine learning calculations, k-nearest neighbor is thought to be a kind of apathetic learning as the speculation past the preparation information does not happen until an inquiry is made to the framework.

Decision Tree Learning

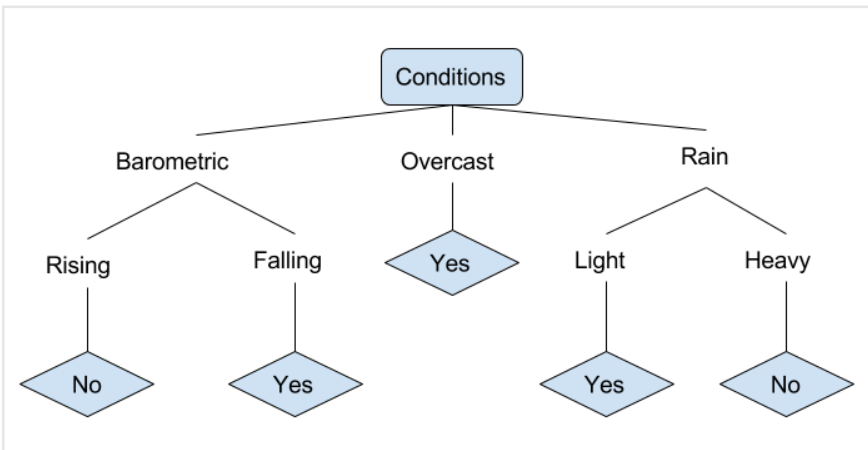
For general use, decision trees are used to apparently address decisions and show up or light up essential authority. When working with machine learning and data mining, decision trees are used as a judicious models. These models depict about data to choices about the data goal regard.

The target of decision tree learning is to influence a model that will foresee the estimation of a goal in perspective of information factors.

In the farsighted model, the data attributes that are settled through recognition are addressed by the branches, while the choices about the data target are addressed in the gets out.

When “taking in” a tree, the source data is parceled into subsets in perspective of a quality regard test, which is reiterated on each one of the construed subsets recursively.

Once the subset at a center point has the proportionate motivating force as its target regard has, the recursion method will be done. We should look at an instance of various conditions that can choose on the off chance that some person should go calculating. This fuses atmosphere conditions and furthermore barometric weight conditions.



In the disentangled choice tree over, a case is grouped by dealing it with the tree to the proper leaf hub. A genuine tree informational index would have significantly higher number of highlights than what is delineated above, yet connections ought to be direct to decide. When working with choice tree “taking in”, a few judgments should be made which include, what highlights to pick, what conditions to use for part, and understanding when the choice tree has achieved a reasonable completion.

Deep Learning

Deep learning tries to mimic how humans can process light and sound jars into vision and hearing. A significant learning configuration is persuaded by natural neural frameworks and contains various layers in a mimicked neural framework made up of hardware and GPUs.

Profound learning uses a course of nonlinear unit layers remembering the true objective to partition or change features (or depictions) of the data. The yield of one layer fills in as the commitment of the dynamic layer. In significant learning, computations can be either controlled and served to the bunch of data, or unsupervised and perform plan examination.

Among the machine learning figurings that are now being used and developed, significant learning ingests the most data and has had the ability to beat individuals in some scholarly assignments. In terms of these characteristics, significant learning has transformed into the approach with imperative potential in the artificial intellectual prowess space.

PC vision and talk affirmation have both recognized basic advances from significant learning approaches. IBM Watson is a prominent instance of a system that utilizes significant learning.

1.3 Selection of Appropriate Algorithm

fmincon Algorithms

For the **fmincon** function, five algorithms have been considered which are as follows:

- ‘interior-point’ (default)
- ‘trust-region-reflective’
- ‘sqp’
- ‘sqp-legacy’
- ‘active-set’

For resolving small, dense problems, interior-point algorithm has been utilized for large point handling problem. This algorithm contains several iteration factors to retain results NaN or Inf results. In case of large-scale application, algorithm is classified as large-scale and medium-scale approach is used to resolve large-scale issues.

The **fmincon** function is evaluated through interior-point algorithm.

The **sqp** function needs to retain all bounds with distinct iteration count. This function also recovers NaN or Inf results classified as large-scale and medium-scale algorithm.

The **sqp-legacy** function in MATLAB is same as the **sqp** function for slower and minimal memory utilization.

The **active-set** function takes steps to speed up the process. This function provides constraints in non-smooth manner and this also belongs to large and medium-scale algorithm.

The **trust-region-reflective** function in MATLAB provides gradient, linear equality, and bounds of the variables. These algorithms provide solutions to small and large sparse problems, also it belongs to large-scale algorithm.

fsolve Algorithms

The MATLAB function with fsolve are utilized based on following three algorithms:

- **trust-region-dogleg** (default)
- **trust-region**
- **levenberg-marquardt**

The solving of non-linear equation is performed. The **trust-region-dogleg** function is used in MATLAB for reducing sum of square in function.

In a similar manner, to solve sparse issue, the ‘trust-region’ algorithm is used. This belongs to the class of large-scale problem which utilizes the Jacobian function.

fminunc Algorithms

The other MATLAB variable, fminunc, uses two algorithm techniques as stated here:

- ‘quasi-newton’ (default)
- ‘trust-region’

The **fminunc** function uses gradient operations similar to trust-region of algorithm for obtaining true functional value. Specify Objective Gradient is utilized for this purpose. Else, in other case, quasi-newton algorithm will be utilized.

Least Squares Algorithms

Let us now consider the other function, **lsqlin**.

In MATLAB function **lsqlin** similar to above following two algorithms are stated:

- **interior-point**, the default
- **trust-region-reflective**

lsqcurvefit and lsqnonlin

In MATLAB function **lsqcurvefit** and **lsqnonlin** with below defined algorithms:

- **trust-region-reflective** (default)
- **levenberg-marquardt**

1.4 Linear Programming Algorithms

Linear programming uses three algorithms as mentioned here:

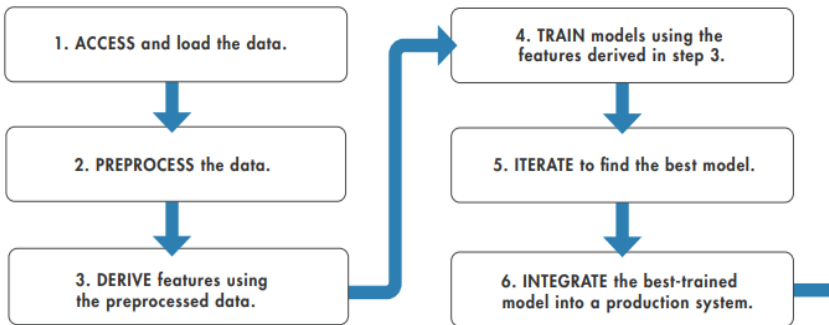
- **dual-simplex**, the default
- **interior-point-legacy**
- **interior-point**

The *dual-simple* algorithm or *interior-point* algorithms are considered.

These algorithms are used for fast and minimal memory utilization.

The interior-point-legacy algorithm is similar to the interior-point algorithm, however, interior-point-legacy has limitations of slower and minimal memory and robustness.

Workflow at a Glance in Machine Learning



Training a Model to Classify Physical Activities

This example is based on a cell application for health monitoring system. The accelerometer and gyroscope consists of sensor data in 3-axial factors. In responses, (or outputs), are the activities performed while walking, standing, running, climbing stairs, or lying down. We require the input data for training classification model to identify these activities. Since our goal is classification, we'll be applying supervised learning.

The trained model (or classifier) will be integrated into an application to help users track their activity levels throughout the day. To achieve this, we'll perform the following steps:

Step 1 – Load the Data

To load data from the accelerometer and gyroscope we do the following:

1. Sit down holding the phone, log data from the phone and store it in a text file labeled **Sitting**.
2. Stand up holding the phone, log data from the phone and store it in another text file labeled **Standing**.
3. Repeat the steps until we have data for each activity we want to classify. We store the labeled datasets in a text file. Flat file formats such as text or CSV are easy to work with and makes it straightforward to import data.

Machine learning calculations are not sufficiently shrewd to differentiate amongst clamor and profitable data. Before utilizing the information for preparing, we have to ensure it's spotless and finished.

Step 2 – Preprocess the Data

In MATLAB, we need to transfer and import data for plotting labeled dataset. Pre-processing of data can be presented as follows:

1. Look for outliers (data points that lie outside the rest of the data). We must decide whether the outliers can be ignored or they indicate a phenomenon that the model should account for. In our example, they can safely be ignored (it turns out that we moved unintentionally while recording the data).
2. Check for missing values (perhaps we lost data because the connection dropped during recording). We could simply ignore the missing values, but this will minimize the dataset. Alternatively, we could substitute approximations for the missing values by interpolating or using comparable data from another sample.
3. Remove gravitational effects from the accelerometer data so that our algorithm will focus on the movement of the subject, not the movement of mobile. In simple filter of highpass there will be biquad process.
4. Divide the data into two sets. We save a part of the data for testing (the test set) and use the rest (the training set) to build models. This is referred to as holdout and is a useful cross validation technique.

Step 3 – Derive the Features

Deriving features (also known as feature engineering or feature extraction) of machine learning is an important constraint.

It turns raw data into information that a machine learning algorithm can use. For the activity tracker, we want to extract features that capture the frequency content of the accelerometer data. These features will help the algorithm distinguish between walking (low frequency) and running (high frequency). We create a new table that includes the selected features.

1.4.1 Steps the Machine Learning Workflow using a Health Monitoring

Feature selection factors are used for the following features:

- Machine learning algorithm accuracy enhancement
- High-dimensional boost model performance enhancement
- Interpret ability model improvement

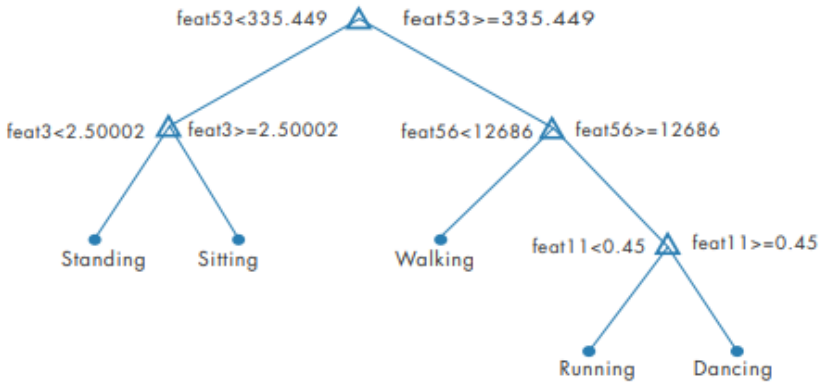
- Prevention of over fitting

The feature count could derive is limited only by your imagination. However, there are a lot of techniques commonly used for different types of data.

Data type	Feature selection task	Techniques
Sensor data	Extract signal properties from raw sensor data to create higher-level information	<p>Peak analysis—perform an fft and identify dominant frequencies.</p> <p>Pulse and transition metrics—derive signal characteristics such as rise time, fall time, and settling time.</p> <p>Spectral measurements—plot signal power, bandwidth, mean frequency, and median frequency.</p>
Image and video data	Extract features such as edge locations, resolution, and color	<p>Bag of visual words—create a histogram of local image features, such as edges, corners, and blobs.</p> <p>Histogram of Oriented Gradients (HOG)—create a histogram of local gradient directions.</p> <p>Minimum eigenvalue algorithm—detect corner locations in images.</p> <p>Edge detection—identify points where the degree of brightness changes sharply.</p>
Transactional data	Calculate derived values that enhance the information in the data	<p>Timestamp decomposition—break timestamps down into components such as day and month.</p> <p>Aggregate value calculation—create higher-level features such as the total number of times a particular event occurred.</p>

Stage 4 – Build and Train the Model

We begin with a fundamental choice tree. When fabricating a model, it's a smart thought to begin with something basic; it will be quicker to run and less demanding to translate.



To perceive how well it performs, we plot the disarray framework, a table that looks at the characterizations made by the model with the real class names that we made in stage 1.

	Sitting	Standing	Walking	Running	Dancing
TRUE CLASS	Sitting	Standing	Walking	Running	Dancing
Sitting	>99%	<1%			
Standing	<1%	99%	<1%		
Walking		<1%	>99%	<1%	
Running			1%	93%	5%
Dancing		<1%	<1%	40%	59%
	PREDICTED CLASS				

The disarray framework demonstrates that our model is experiencing difficulty recognizing strolling and running. Perhaps a choice tree doesn't work for this kind of information. We'll attempt a couple of various calculations.

We begin with **K-nearest neighbors (KNN)**, a basic calculation that stores all the preparation information, thinks about new indicates the preparation information, and returns the most regular class of the K-nearest focuses. That gives us 98% exactness contrasted with 94.1% for the basic choice tree. The perplexity grid looks better:

TRUE CLASS	Sitting	>99%	<1%			
	Standing	1%	99%	1%		
	Walking		2%	98%		
	Running		<1%	1%	97%	1%
	Dancing		1%	1%	6%	92%
		Sitting	Standing	Walking	Running	Dancing
		PREDICTED CLASS				

KNNs take a lot of memory to run, since they require all the preparation information to make an expectation. We attempt a direct discriminant display; however, that doesn't enhance the outcomes. At last, we attempt a multiclass bolster vector machine (SVM). The SVM does—we now get 99% precision.

TRUE CLASS	Sitting	>99%	<1%			
	Standing	<1%	>99%	<1%		
	Walking		<1%	>99%		
	Running			<1%	98%	2%
	Dancing		<1%	<1%	3%	96%
		Sitting	Standing	Walking	Running	Dancing
		PREDICTED CLASS				

We accomplished our objective by emphasizing on the model and attempting distinctive calculations. In the event that our classifier still couldn't dependably separate amongst strolling and running, we'd investigate approaches to enhance the model.

Stage 5 – Improve the Model

Disentangle

To begin with, we search for chances to decrease the quantity of highlights. Well known element decrease strategies include:

- **Correlation network:** Demonstrates the connection between factors, with the goal that factors (or highlights) that are not exceedingly related can be evacuated.
- **Principal segment investigation (PCA):** Disposes off excess by finding a blend of highlights that catches key qualifications between the first highlights and brings out solid examples in the dataset.
- **Sequential component diminishment:** Lessens includes iteratively on the model until there is no change in execution.

- Pruning branches from a choice tree
- Removing students from a gathering

Include Complexity

In the event that our model can't separate strolling from running since it is over-summing up, we need to discover approaches to make it all the more tweaked. To do this, we can do the following:

- **Use display blend:** Combine numerous, less difficult models into a bigger model that is better ready to speak to the patterns in the information than any of the less complex models could without anyone else.
- **Add more information sources:** Take a gander at the whirligig information and also the accelerometer information. The gyration records the introduction of the PDA amid movement. This information may give special marks to the diverse exercises; for instance, there may be a blend of increasing speed and revolution that is one of a kind to running. Once we've balanced the model, we approve its execution on the test information that we put aside amid preprocessing. On the off chance, that the model can dependably characterize exercises on the test information, we're prepared to move it to the telephone and begin following.

CHAPTER 2

Finding Natural Patterns in Data

2.1 Unsupervised Learning

Unsupervised learning corresponds to a machine learning algorithm for the reduction in datasets in reactions which are not named.

In unsupervised learning approaches clustering has been widely utilized for the examination of exploratory information with concealed information collection. Based on the measurements and display of the collection of information, unsupervised learning operates similar to Probabilistic or Euclidean separation.

Calculation of regular bunching needs to include the following steps:

- **Hierarchical Clustering:** The learning algorithm builds a group of multilevel performance into a tree like hierarchy of the cluster.
- **K-Means Clustering:** Information in the learning algorithm are segmented into groups under considerable k clustering in group with centroid.
- **Gaussian Models:** Multivariate provides a group with a blend of normal thickening parts.
- **Self-sorting Algorithm:** Neural network uses a topology for information convergence.
- **Hidden Markov model:** Utilizes observed information to recuperate the succession of information.

Unsupervised learning strategies utilizes a part of bioinformatics for succession investigation and hereditary bunching; in information digging for grouping and pattern mining, therapeutic imaging for picture division, and in PC vision for question acknowledgment.

Unsupervised learning is the preparation of a man-made brainpower (AI) algorithm using data that is neither characterized nor marked, enabling the calculation to follow up on that data without direction.

Unsupervised learning provides information in input (X) format without any yield function comparison.

In unsupervised learning the objective function is to illustrate a hidden structure or information dissemination for evaluating more information.

Unsupervised learning provides regular learning in a dissimilar manner without any provision of an answer to the instructor. To find desire evaluation calculation are illustrated left for identification to present information structure in intriguing.

To resolve the affiliation problem Unsupervised learning is classified into Clustering and Association rules.

- **Clustering:** Clustering provides a resolution in finding characteristic groups information. For example, in evaluating client information.
- **Association:** To resolve the learning affiliation problem we need to find a large part of information. An example of this is, instead of purchasing product X people purchase Y.

2.2 Clustering Strategies

2.2.1 Hard Clustering Calculations

Cluster Data evaluation using **Gaussian Mixture Model (GMM)**. The Gaussian Mixture Model is mainly utilized for information clustering. Generally, a GMM fixed bunch with relegating inquiry information focuses on the multivariate typical parts that augment the segment back likelihood of information. In GMM, a fixed value is assigned to question information for segment occupancy with the most astounding back likelihood. This technique is used for allotting an information precision point with hard clustering in one group.

The information clustering demonstrates a fitted model for clustering approach and the probability of back part to evaluate Hard Clustering data with the Cluster Gaussian Mixture.

2.2.2 Soft Clustering Calculations

GMM grouping is more adaptable in light of the fact for *fuzzy* or *soft* clustering approach. Delicate bunching techniques allot point of information

in each group point. Evaluation of information shows affiliation of quality information from the group. Rather than hard bunching techniques, delicate grouping strategies are adaptable, in that they can relegate an information point up to an excess of one bunch. When grouping with GMMs, the score is the back likelihood. In case of delicate grouping, utilization of GMM is generally in relation to Gaussian Mixture Data with Soft Clustering.

GMM clustering provides a group suit with different sizes and a structural relationship along them. In these lines GMM clustering provides more appropriate information, an example of which is k-means clustering. In grouping strategies, we should indicate the quantity for wanted bunches as previously modeled of fitting. Quantity bunches determine quantity parts of the GMM.

In GMMs, for effective practices consider the **Part Covariance Structure** that determines edges or a full covariance vector, which in all parts are similar to covariance lattice.

Covariance Structure Option

Loading data set of Fisher's Iris. Consider sepal measurement for clustering.

Program

```
Load fisheriris;
X = meas(:, 1:2) ;
[n ,p] = size (X) ;
rng (3) ; % For reproducibility

figure ;
plot (X ( : , 1) , X (: , 2) , ' . ' , 'Markersize' ,
15) ;
title ( 'Fisher ' ' s Iris Data Set ' ) ;
xlabel ( ' Sepal length (cm) ' ) ;
ylabel ( ' Sepal width (cm) ' ) ;
```

Output



The quantity k is defined for components where GMM decides the count of sub-population or groups. From the above figure we need to conclude two, three or more fitting in parts. With the increase in k in an unpredictable manner the GMM is evaluated. Every part compiled with framework of covariance.

Geometrically, an ellipsoid structure provides a covariance with the certainty of ellipsoidal with a group or sub population. The covariance lattices are indicated from edge to edge segmentation in complete or every parts of same grid covariance. The determination thus provides a mixing of shape and ellipsoid introduction.

GMMs fit information to inspect impacts for determining a complete mixer of covariance structure alternatives with the ellipsoidal state. To determine the blends of name-esteem, adopt a combination of **Covariance Type** and **Shared Covariance**.

A Covariance structure is used in all parts of information determination. In the outline, determined for classification of three parts as well as in ellipsoidal structure where GMMs fixes the cluster framework for particular plane for extreme estimation.

Acquired score indicates 99% likelihood edge in every certainty district. For this decides major and minor edges of ellipsoids.

Shading circle, utilize comparative shading to its bunch.

Program

```

k = 3;
Sigma = { 'diagonal' , 'full' } ;
nSigma = numel (Sigma) ;
SharedCovariance = {true, false} ;
SCText = { 'true' , 'false' } ;
nSC = numel (SharedCovariance) ;
d = 500;
x1 = linspace (min (X ( : , 1) ) - 2 , max (X ( : , 1)
) + 2, d) ;
x2 = linspace (min (X ( : , 2) ) - 2 , max (X ( : , 2)
) + 2, d) ;
[x1grid, x2grid] = meshgrid (x1, x2) ;
X0 = [x1grid ( : ) x2grid ( : ) ] ;
threshold = sqrt ( chi2inv (0.99, 2) ) ;
options = statset ( 'MaxIter ' , 1000 ) ; % Increase
number of EM iterations

figure;
c = 1;
for i = 1: nSigma;
    for j = 1: nSC;
        gmfit = fitgmdist (X, k, 'CovarianceType', Sigma{i},
        . . .
        'SharedCovariance', SharedCovariance{j},
'Options', options) ;
        clusterX = cluster (gmfit , X) ;
        mahalDist = mahal (gmfit , X0) ;
        subplot (2, 2, c) ;
        h1 = gscatter (X ( : , 1) , X ( : , 2 ) , clusterX) ;
        hold on;
        for m = 1 : k;
            idx = mahalDist (: , m)<=threshold;
            Color = h1(m) .Color*0.75 + -0.5* (h1(m) .Color - 1)
;
            h2 = plot (X0(idx,1) ,X0(idx,2) , '\.', 'Color',
Color,'MarkerSize',1)
            uistack(h2, 'bottom') ;

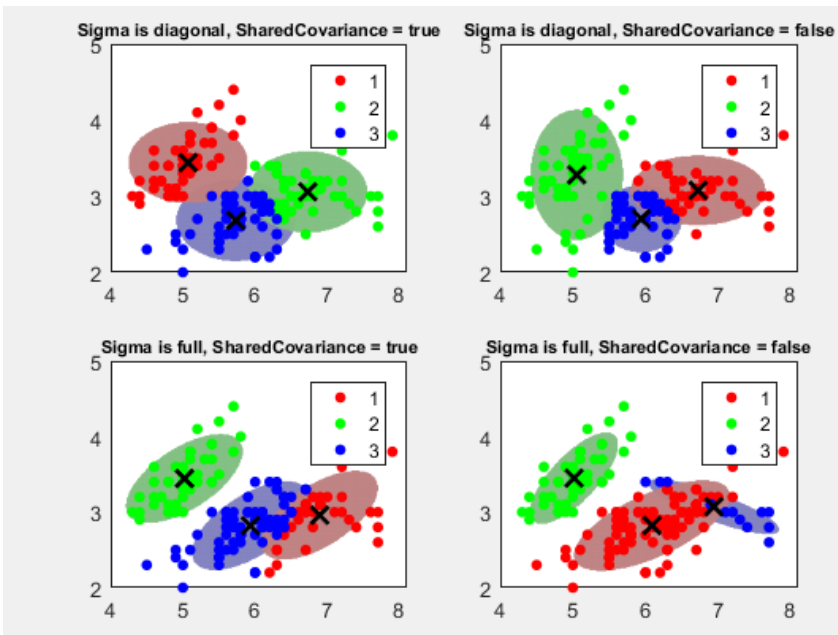
```

```

end
plot (gmfit.mu (:,1) ,gmfit.mu (:,2) , 'kx' ,
'LineWidth',2, 'MarkerSize',10)
title (sprintf('\Sigma is %s, SharedCovariance = %s' ,
. . .
Sigma {i}, SCText{j}, 'FontSize',8)
legend (h1, { '1', '2', '3' } ) ;
hold off
c = c + 1;
end
end

```

Output



The likelihood limit for the certainty district decides the minor and major framework, in which the covariance district compose decides framework introduction.

Corner to corner covariance frameworks demonstrate that indicators are not correlated. The minor and major framework of circles is calculated

in a parallel manner or opposite for x and y axis. For a particular build to aggregate p for parameter number, indicators of quality segment more stingy covariance determination for the complete operation.

Covariance grids take into account related indicators with no confinement introduction of the circles with respect to axis in x and y direction. Every segment expands by an aggregate parameter number, however it catches relationship structure indicators which lead to overfitting.

A network with covariance in split provides demonstration in lattice of covariance. The oval shape is similar to previously defined function. More stingy detail of grounds in unshared data are presented in a parameter with the aggregator count with covariance quality in one segment.

Covariance lattices show complete segments on their own network structure. The shape of the circle with an introduction may vary for a detailed quality parameter in k-means of parameters of covariance for segments.

This figure additionally demonstrates how a cluster group generally safeguards a group arrangement. That is, whether or not you bunch a few fitted **gmdistribution** models, the 'cluster' might dole out various group names for comparative segments.

Starting Conditions

Expectation-Maximization (EM) provides GMM fitness. In similar manner k-means provides clustering calculation, EM introductory conditions to focalize neighborhood ideal. To determine beginning estimation parameters, indicate an introductory bunch for information focuses or anything haphazardly picked for utilization of k-implies ++ calculation.

The calculation fits GMM for information delicate beginning conditions. In this case, for evaluating GMMs fitness indicate a distinctive initial condition. In particular, indicated at first the information guides have a place toward the main bunch, two arrangements of arbitrary, introductory assignments, and use k-implies ++ to get starting group focuses.

For all cases, indicate three parts; unshared and full covariance frameworks, similar introductory blend extent, and similar starting covariance lattices. For solidness when you attempt distinctive arrangements of starting esteems, increment the quantity of EM calculation emphases.

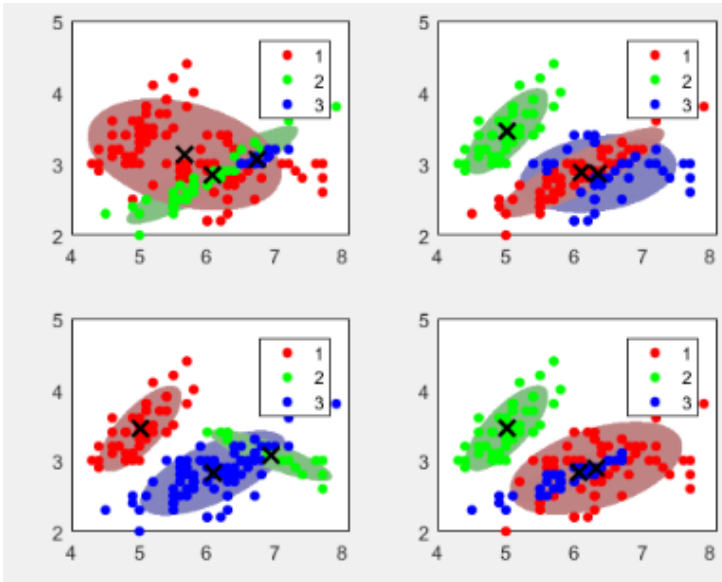
Program

```

cluster0 = { [ones (n-8, 1) ; [2; 2; 2; 2] ; [3; 3; 3;
3] ] ; . . .
    randsample (1:k, n, true) ; randsample (1:k, n,
true) ; 'plus' } ;
converged = nan (4, 1) ;
figure;
for j = 1 : 4 ;
    gmfit = fitgmdist (X, k, 'CovarianceType', 'full', . .
.
    'SharedCovariance', false, 'Start', cluster0 {j} ,
. . .
    'Options' , options) ;
clusterX = cluster (gmfit,X) ;
mahalDist = mahal (gmfit,X0) ;
subplot (2, 2, j) ;
h1 = gscatter (X(:, 1) ,X(:, 2) , cluster);
hold on;
nK = numel (unique (cluster) ) ;
for m = 1:nK;
    idx = mahalDist (: ,m)<=threshold;
    Color = h1(m) .Color*0.75 + -0.5* (h1(m) .Color -
1) ;
h2 = plot (X0 (idx, 1),X0 (idx, 2) , ' . ' , 'Color' ,
Color, 'MarkerSize' , 1) ;
uistack (h2, 'bottom' ) ;
end
plot (gmfit . mu (: , 1) , gmfit . mu (: , 2 ) , 'kx' ,
'LineWidth' , 2 , 'MarkerSize' , 10)
legend (h1,{ ' 1 ' , ' 2 ' , ' 3 ' } ) ;
hold off
converged (j) = gmfit . Converged;
end
sum (converged)

```


Output



All calculations focalize every one of the diverse arrangements of beginning bunch assignments for the information guides leading toward an alternate, fitted group task. For name - esteem of 'Replicates' argument for positive or whole number, for predefined number of time calculation. The higher probability `fitgmdist` fit yield.

Regularization Parameter

On the off chance that, for instance, you have a larger number of indicators than information focuses, at that point you can regularize for estimation steadiness.

Now and again, amid an EM cycle, a fitted covariance lattice can turn out to be badly molded, that is, the probability is getting away to unending. This can happen if:

- There are a bigger number of indicators than information points.
- You determine to fit with an excessive number of parts.
- Factors are exceptionally related.

To conquer this issue, determine a small positive number for utilizing **Regularize** match argument. The function `fitgmdist` provides a

predetermined small positive number integer to edge components of covariance structure, for guaranteed frameworks in clear manner. Regularization decreases the maximum probability esteem. At this points of interest, see `fitgmdist`.

Group Gaussian Mixture Data Using Hard Clustering

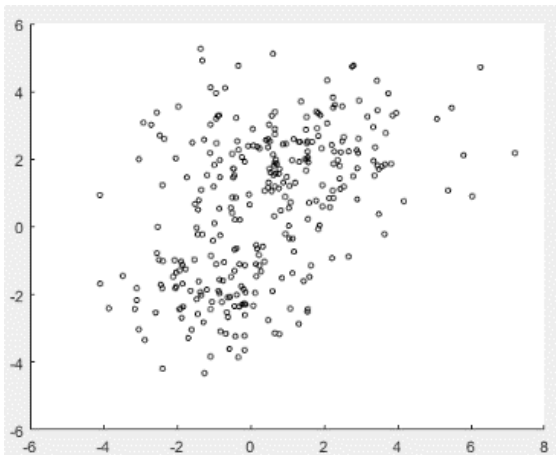
This case demonstrates the execution of hard grouping on mimicked information from a blend of Gaussian dispersions. Gaussian blend models can be utilized for grouping information, by understanding multivariate segments in fitted model to speak bunches. Reproduce Data from a Mixture of Gaussian Distributions. Reproduce information with a blend of bivariate Gaussian circulations function `mvnrnd`.

Program

```
rng ( 'default' ) % For reproducibility
mu1 = [1 2] ;
sigma1 = [3 .2; .2 2] ;
mu2 = [-1 -2] ;
X = [mvnrnd (mu1, sigma1, 200) ;mvnrnd (mu2, sigma2,
100) ] ;
n = size (X, 1) ;
```

```
figure
scatter (X(:, 1) , X(:, 2) , 10, 'ko' )
```

Output



Fit the Simulated Data to a Gaussian Mixture Model

Gaussian Mixture Model (GMM) with two-fit segmentation. With the right number of segment utilization. Practically speaking, genuine information, choice require contrasting model and distinctive quantities in segments.

Program

```
options = statset ( ' Display ' , ' final ' ) ;
gm = fitgmdist (X, 2, 'Options' , options )
33 iterations, log-likelihood = -1210.59
```

gm =

Gaussian mixture distribution with 2 components in 2 dimensions

Component 1:

Mixing proportion: 0.629379

Mean: 1.0758 2.0426

Component 2:

Mixing proportion: 0.370621

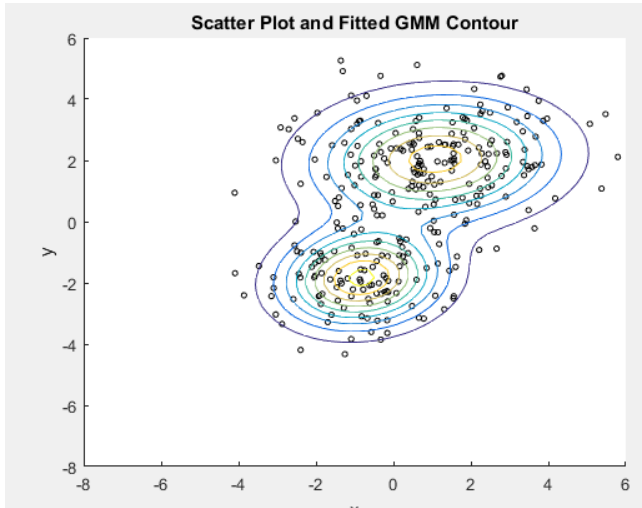
Mean: -0.8292 -1.8482

Plot the evaluated likelihood thickness that forms from the two-segment blend circulation. The two bivariate ordinary parts cover, yet their pinnacles are particular. This recommends the information could sensibly be isolated into two groups.

Program

```
hold on
ezcontour (@ (x, y)pdf (gm, [x y]) , [-8 6] , [-8 6]
)
title ( 'Scatter Plot and Fitted GMM Contour' )
hold off
```

Output



Cluster the Data Using the Fitted GMM

Cluster implements **hard grouping**, a strategy that allots every datum point to precisely one bunch. For GMM, the cluster assigns each point to one of the two blend segments in the GMM. The focal point of each bunch is the comparing blend part mean.

Parcel the information into bunches by passing the fitted GMM and the information to cluster.

Program

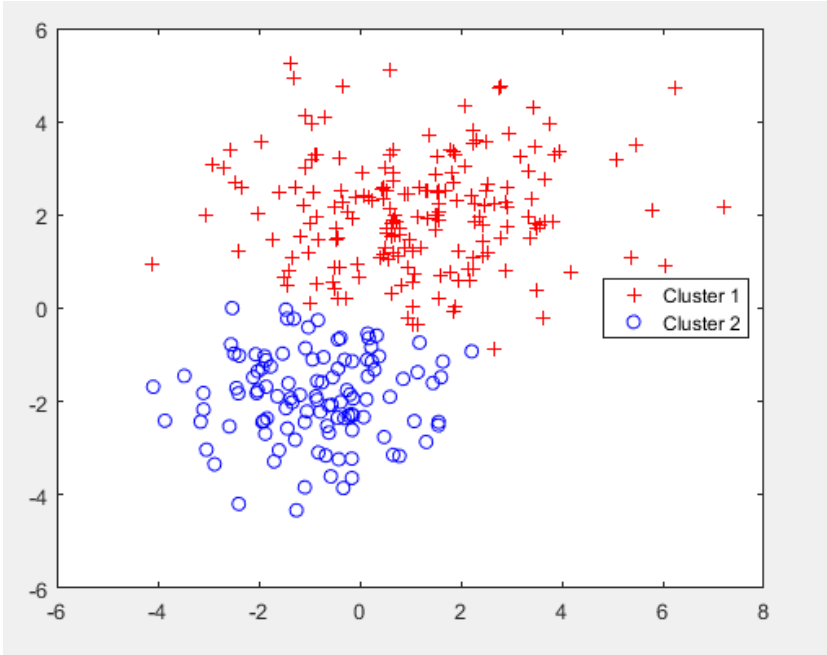
```
ezcontour (@ (x, y) pdf (gm, [x y] ) , [-8 6] , [-8 6]
)
title ( 'Scatter Plot and Fitted GMM Contour' )
hold off

idx = cluster (gm, X) ;
cluster1 = (idx == 1) ; % |1| for cluster 1 membership
cluster2 = (idx == 2) ; % |2| for cluster 2 membership
```

figure

```
gscatter (X (: , 1) ,X (: , 2) , idx, 'rb' , '+o' )
legend ( 'Cluster 1', 'Cluster 2' , 'Location', 'best'
)
```

Output



Each group compares bivariate ordinary parts with blend distribution. Cluster information bunch in view of a group participation score. Each bunch enrollment score is an evaluated back likelihood that has an originating information point from the relating component. Cluster assigns each point to the blend segment comparing to the most elevated back likelihood.

You can appraise group enrollment back probabilities by passing the fitted GMM and information to either group and demand to restore the third yield contention.

Estimate Cluster Membership Posterior Probabilities

Program

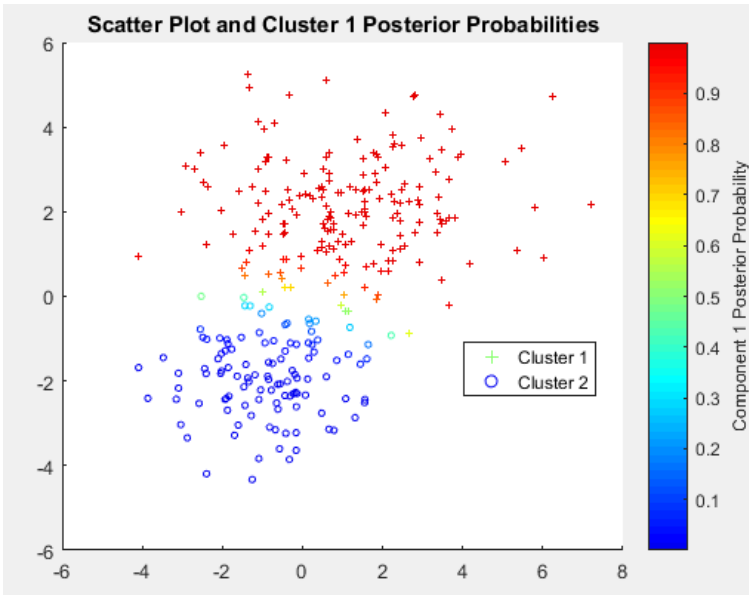
```

P = posterior (gm,X) ;

figure
scatter (X ( cluster1 , 1 ) , X ( cluster1 , 2 ) ,
10 , P ( cluster1 , 1 ) , '+' )
hold on
scatter (X ( cluster2 , 1 ) , X ( cluster2 , 2 ) , 10
, P ( cluster2 , 1 ) , 'o' )
hold off
clrmap = jet (80) ;
colormap (clrmap (9 : 72 , : ) )
ylabel ( colorbar , 'Component 1 Posterior
Probability ' )
legend ( 'Cluster 1', 'Cluster 2' , 'Location', 'best'
)
title ( 'Scatter Plot and Cluster 1 Posterior
Probabilities' )

```

Output



Assign New Data to Clusters

You can likewise utilize the cluster method to appoint new information focuses to the blend parts found in the first information. Reproduce new information from a blend of Gaussian conveyances. As opposed to using `mvnrnd`, you can make a GMM with the genuine blend segment means and standard deviations using `gmdistribution` and after that pass the GMM to `random` to mimic information.

Program

```

Mu = [mu1; mu2] ;
Sigma = cat (3, sigma1, sigma2) ;
p = [0.75 0.25] ; % Mixing proportions

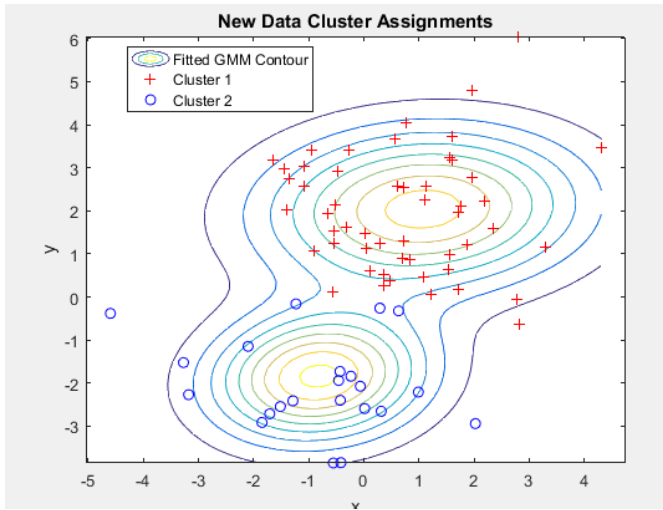
gmTrue = gmdistribution (Mu, Sigma, p) ;
X0 = random (gmTrue, 75) ;

[idx0,~, P0] = cluster (gm,X0) ;

figure
excontour (@ (x, y) pdf (gm, [x y] ) , [min (X0 ( :
, 1) ) max (X0 ( : , 1) )] , . . .
[ min (X0 ( : , 2) ) max (X0 ( : , 2) ) ] )
hold on
gscatter (X0 (: , 1) , X0 (: , 2) , idx0 , 'rb' ,
'+o' )
legend ( 'Fitted GMM Contour' , 'Cluster 1', 'Cluster
2' , 'Location', 'best' )
title ( 'New Data Cluster Assignments ' )
hold off

```

Output



Cluster Gaussian Mixture Data Using Soft Clustering

Cluster estimates bunch enrollment back probabilities and afterward doles out each point to the group, relating to the most extreme back likelihood. Delicate bunching is an elective grouping strategy that permits a few information focuses to have a place with different bunches.

To execute delicate bunching:

1. Dole out a bunch participation score to every datum point that portrays how comparable each point is to each group's prime example. For a blend of Gaussian dispersions, the group model is relating part mean and the segment can be the evaluated bunch enrollment back likelihood.
2. Rank the focuses by their group enrollment score. Examine the scores and decide group participations. For calculations that utilization back probabilities as scores, an information point is an individual from the group relating to the most extreme back likelihood. In any case, if there are different groups with relating back probabilities that are near the most extreme, at that point the information point can likewise be an individual from those bunches. It is great practice to decide the limit on scores that yield numerous bunch participations before grouping.

3. Mimic information from a blend of two bivariate Gaussian circulations.

Program

```
rng (0, 'twister' ) % For reproducibility
mu1 = [1 2] ;
sigma1 = [3 .2; .2 2] ;
mu2 = [-1 -2] ;
sigma2 = [2 0; 0 1] ;
X = [mvrnd (mu1, sigma1, 200) ;mvrnd (mu2, sigma2,
100) ] ;
```

Fit a two-segment Gaussian blend demonstrate (GMM). Since there are two parts, assume that any information point with group enrollment back probabilities in the interim [0.4,0.6] can be an individual from the two bunches.

```
gm = fitgmdist (X, 2) ;
threshold = [0.4 0.6] ;
```

Gauge segment part back probabilities for all information focuses utilizing the fitted GMM **gm**. These speak to group participation scores.

```
p = posterior (gm, x);
```

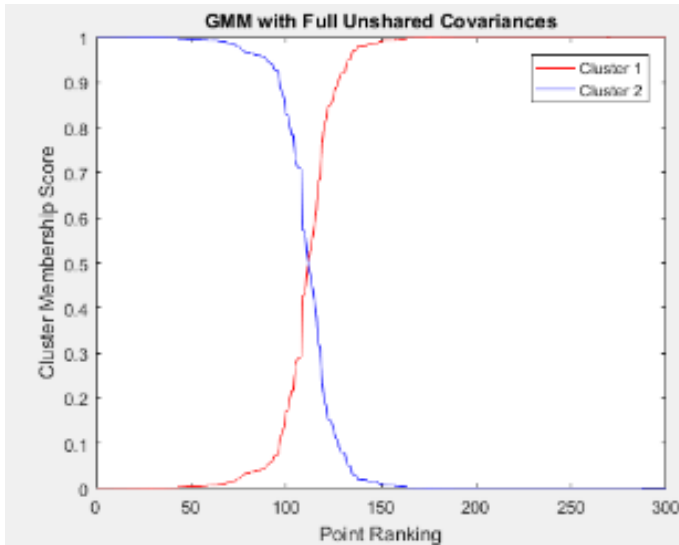
For each group, rank the participation scores for all information focuses. For each bunch, plot every datum guides participation score with deference toward its positioning in respect to every other datum focuses.

Program

```
n = size (X,1) ;
[~, order] = sort (P (: , 1) ) ;

figure
plot (1:n, P (order, 1) , 'r-' , 1:n, P (order,2) ,
'b-' )
legend ( { 'Cluster 1' , 'Cluster 2' } )
ylabel ( 'Cluster Membership Score' )
xlabel ( 'Point Ranking' )
title ( 'GMM with Full Unshared Covariances' )
```

Output



In spite of the fact that a reasonable division is difficult to find disperse plot information, the participation shows fitted dissemination complete the job of isolating information gathering.

Plot the information and relegate bunches by most the extreme back likelihood. Distinguish focuses that could be in either group.

```
idx = cluster (gm, x);
idxBoth = find(P(:, 1) >= threshold(1) & P(:,
1) <= threshold (2));
numInBoth = numel(idxBoth)
numInBoth =
    7
```

Program

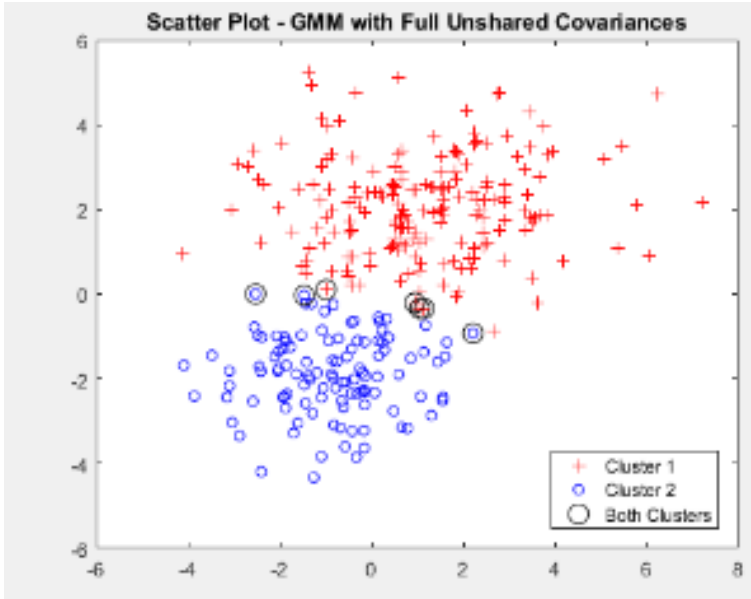
```
figure
gscatter (X (: , 1) , X (: , 2) , idx, 'rb' ,
'+o' , 5)
hold on
plot (X(idxBoth, 1) , X (idxBoth, 2) , 'ko' ,
'MarkerSize' , 10)
legend ( {'Cluster 1' , 'Cluster 2' , 'Both
Clusters' } , 'Location' , 'SouthEast' )
```

```

title ( 'Scatter Plot - GMM with Full Unshared
Covariances' )
hold off

```

Output



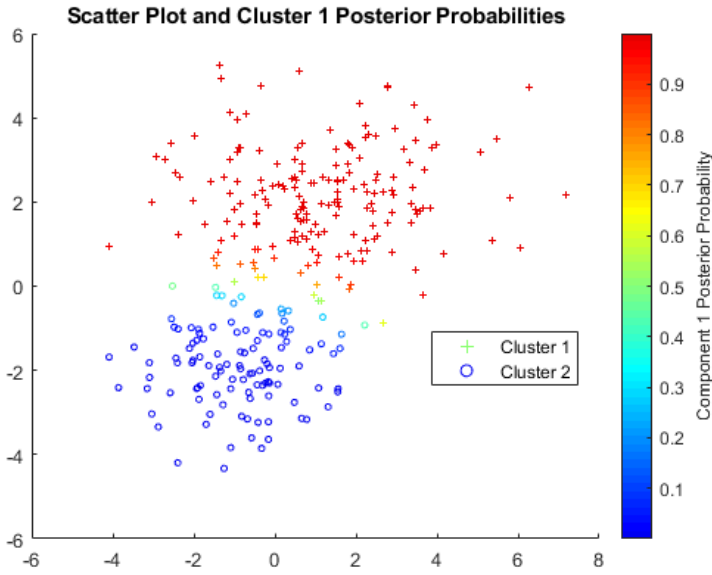
2.3 Cluster Evaluation and Interpretation

2.3.1 Common Dimensionality Reduction Techniques for Improving Model Performance

Cluster Analysis

Unsupervised learning strategies discover common clustering and examples for information group examination. Additionally called division investigation or scientific categorization investigation, allotments test information into bunches or clusters. Bunches are framed to such an extent that articles in a similar group are fundamentally the same and questions in various groups are exceptionally unmistakable.

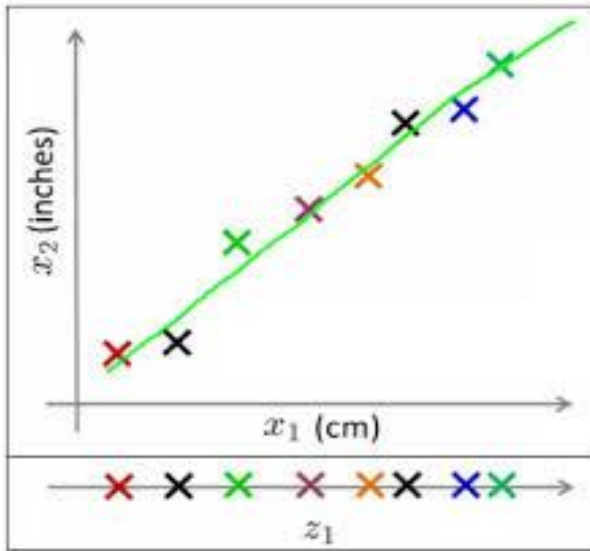
Measurements and Machine



Dimensionality Reduction

Measurement Reduction refers to the way toward changing over an arrangement of information having vast dimensions into information with lesser measurements ensuring that it conveys comparable data concisely. These procedures are normally used while solving machine learning problems to obtain better highlights for an order.

We should take a gander at the picture demonstrated as follows. It demonstrates 2 measurements x_1 and x_2 , which give us a chance to state estimations of a few object in cm (x_1) and inches (x_2). Presently, if you somehow happened to utilize both these measurements in machine learning, they will pass on comparative data and present a great deal of commotion in framework, so you are better of simply utilizing one dimension. Here we have changed the measurement of information from 2D (from x_1 and x_2) to 1D (z_1), which has made the information generally simpler to explain.



In comparative ways, we can decrease n measurements of informational index to k measurements ($k < n$). These k measurements can be straightforwardly recognized (separated) or can be a blend of measurements (weighted midpoints of measurements) or new dimension(s) that speak to existing various measurements well.

A standout amongst the most well-known utilization of this method is **Image handling**. You may have come across the Facebook application – “Which Celebrity Do You Look Like?”. Here’s the answer: To distinguish the coordinated big-name picture, we utilize pixel information and every pixel is identical to one measurement. In each picture, there are a high number of pixels i.e. high number of measurements. What’s more, every measurement is vital here. You can’t overlook measurements randomly to make a better feeling of your general informational collection. In such cases, dimension decrease strategies help you locate the huge dimension(s) utilizing different method(s). We’ll examine these techniques right away.

CHAPTER 3

Building Classification Methods

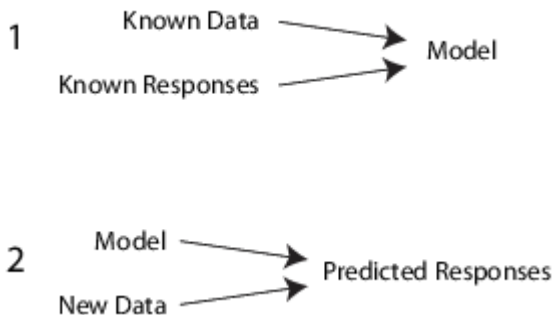
3.1 Supervised Learning

Work Flow and Algorithm for Supervised Learning

Definition of Supervised Learning?

Machine learning is a model which forecasts a view of the result with minimal vulnerability. As versatile calculations distinguish designs in information, a PC “learns” from the perceptions. At the point when presented to more perceptions, the PC enhances its prescient execution.

Machine learning provides calculation with an arrangement of information for yielding information and model training that forecast a sensible reaction of information data.



Supervised learning can be divided into two distinct methods: classification and regression.

In classification, the objective of the information class is allotted (or labeled) with certain limited class of information perception ordering. In this case, reaction consist of factor information. Applications incorporate spam channels, commercial suggestion frameworks, and picture & discourse

acknowledgment. Anticipating whether a patient will show at least a bit of kindness assault inside a year is an arrangement issue with conceivable classes whether it belongs to true or false. Grouping calculations normally ostensible reaction esteems. Be that as it may, a few calculations can suit class of ordinary information (function of fitcecoc).

In regression analysis, the objective is to foresee consistent estimation for a perception. For this, reactions factor with genuine numbers. Here applications incorporate gauging costs, vitality utilization, or malady rate.

Supervised learning adopts the following steps:

- Preparation of Data
- Appropriate algorithm selection
- Fitness of Model
- Selection of Validation Method
- Selection of Fitted and Update Satisfied Condition

Prepare Data

All regulated learning strategies begin with an info information lattice, typically d as X, or indicator. Deals with sections of X in NaN values. Generally, Toolbox related to Machine Learning algorithm regulated calculations to manage NaN values, with disregarding or by overlooking column with NaN information.

To utilize different information writings for reaction data Y, every component denotes Y where the reaction of relating column is X. Perceptions of Y with disregarded information missing count.

In regression value, Y need to be of numeric vector with indistinguishable count of components from quantity of X columns.

In classification, Y is based on following information compositions.

Data Types	Value of Missing
Numeric	NaN
Categorical	<undefined>
Character Array	Row of spaces
Cell Array of character vectors	, ,
Logical Vector	(cannot represent)

Model Fitness

Model function for fitness is based in the algorithm selection.

Classification Tree	Fitctree
Regression Tree	Fitrtree
Discriminant Analysis	Fitcdiscr
<i>k</i> -Nearest Neighbors	Fitcknn
Naive Bayes	Fitcnb
Support Vector Machines (SVM)	Fitsvm
SVM of regression	Fitrsvm
Multiclass models or other classifiers	Fitcecoc
Classification Ensembles	Fitc ensemble
Regression Ensembles	fitr ensemble
Classification or Regression Tree (e.g., Random Forests [1]) in Parallel	Tree Bagger

Validation Method Selection

In validation model three fundamental techniques are used to inspect exactness of the subsequent fitted model which are presented as below:

Look at the re-substitution mistake. For cases, see:

- Classification Tree Re-substitution Error
- Regression Tree for Cross Validation
- Quality of ensemble classifier

Example: Re-substitution Error of a Discriminant Analysis Classifier

Order Tree Re-substitution Error

1. For loading Fisher's iris data related in MATLAB following function is utilized.

```
>> load fisheriris
```

2. Entire data loaded are trained using classification tree built in default.

```
>> Mdl = fitctree (mead, species);
```

3. Examine the re-substitution error.

```
>> resuberror = resubLoss (Mdl)
resuberror = 0.0200
```

Cross Validate a Regression Tree

This illustration demonstrates to look at the re-substitution and cross-approval precision of a relapse tree for foreseeing mileage in light of carsmall data. This dataset is loaded for quickening, uprooting, torque and weight as indicators of MPG.

```
>> load carsmall
X = [Acceleration Displacement Horsepower Weight];
```

Regression tree growth through observations.

```
>> rtree = fitrtree (X, MPG);
>> resuberror = resubLoss (rtree)
```

Compute the in-sample error.

```
resuberror = 4.7188
```

Estimate the cross-validation MSE.

```
>> rng 'default';
cvrtree = crossval (rttree);
cvloss = kfoldLoss (cvrtree)
```

```
cvloss =
    23.8065
```

Selection of Algorithm

The characteristics of an algorithm differ in different aspects, which are stated as follows:

- Training of data speed
- Utilization of Memory
- New data accuracy prediction
- Transparency or interpretability,
- Which reveals ease of algorithm understanding and prediction.
- Use Fitted Model for Predictions

3.2 Supervised Machine Learning

The main functionality of a machine learning algorithm is the utilization of learning techniques through machines. In a supervised approach we

are able to get the output variable data (Y) through input variable (X) for learning mapping function.

$$Y = f(X)$$

The primary goal is to approximate the mapping function with the newly defined input data (X) for output data prediction variables (Y) for given data.

In supervised learning, learning is obtained by training a dataset similar to the learning process via the supervision of a teacher. We are already aware of the correct answer, through the iterative prediction of training data and rectify the process through teaching. In this case, learning will be stopped when a desired acceptable performance is achieved in the machine.

Even supervised learning subjected to classification and regression has a major problem for performance measurements.

- **Classification:** For this particular problem category the output variable is classified as “disease” or “no disease” or “blue” or “red”.
- **Regression:** When output variable is obtained as real value regression problem will occur, Example for this are “weight” or “dollars”.

Fewer problems built with classification and regression require time series prediction and recommendation respectively.

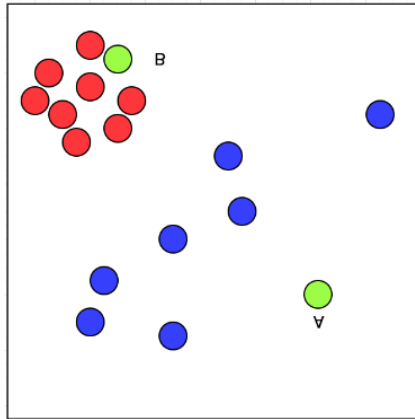
Most popular supervised machine learning algorithms are stated with problem solution:

- Linear regression to solve regression problems.
- Random forest to solve regression and classification problems.
- Support vector machines to solve classification problems.

Supervised Learning has Input and Correct Output

Let us take a look at an example of supervised learning. We have data regarding an opinion of a particular movie whether a person would “like” or “not like” a particular movie. This data is obtained through a sequential interview and with the data collected based on this we can be able to predict if the movie will be a “hit” or “not”.

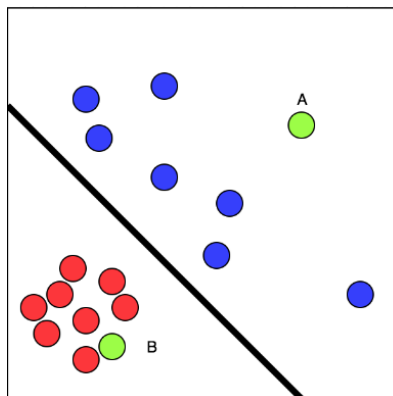
Raw Data



The picture above deals with the number of restaurants I visited. In this picture, visited restaurants are marked as red circle and restaurant which I didn't visit are denoted by blue circle.

Now, I have two options A and B are marked in green color to visit, Can you predict which one I select? Yes, you can simply classify the linear data in two parts with a line for separating the red and blue circle. For a clearer understanding look at the picture below:

Learned by Supervised Learning



It is simply easy, you can tell with assured confidence I will visit restaurant B more than A. This is the kind of process that goes on with supervised machine learning approach.

3.3 Unsupervised Machine Learning

If data only consists of input data (X) without any output variables corresponding to those input variables, then it belongs to unsupervised machine learning algorithm.

The primary goal of unsupervised learning is data distribution with an underlying structure for better understanding of data.

This kind of data processing is known as unsupervised learning since like supervised learning it does not have a correct answer and teacher for learning. In this unsupervised learning, the algorithm is left in own devices for discovering and structure intersecting in the data.

Similar to supervised learning unsupervised learning is also grouped as association and clustering problems.

- **Clustering:** Clustering problem arises for discovering data with inherent groups similar to purchasing behavior of customer group.
- **Association:** Learning problem with the association rule is to discover rules for data at larger portion like people who tend to buy the variable product X also tend to buy variable Y .

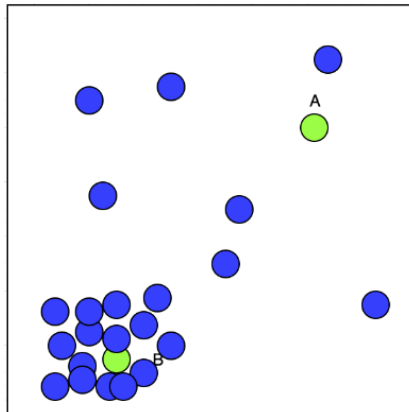
Fewer examples for unsupervised algorithm learning are:

- Clustering problem is resolved with the development of k-means.
- To resolve association rule learning problem Apriori algorithm is developed.

Unsupervised Learning has Inputs

Suppose a taxi driver has the option of either accepting or rejecting the bookings. For this case, in below picture the blue circles plotted denote the accepted booking.

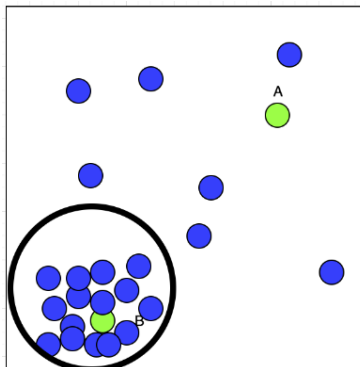
Raw Data for Unsupervised Learning



Now, the taxi driver has two booking at place A and place B; could you tell which one he will select? To predict the acceptance cluster, we see that more booking has been accepted by the driver in the area at lower left corner as shown in below picture.

Unsupervised Learning

Unsupervised Learning



The major difference between supervised and unsupervised machine learning algorithm is stated as below:

- In supervised machine learning, algorithm datasets are trained with labels which is added by the engineer or data scientist to make them understand important features.
- In Unsupervised machine learning, algorithms training datasets are unlabeled for detection of feature importance based on own inherent data patterns.

3.4 Semi-supervised Learning

When abundant input data (X) and fewer labeled data are available (Y) then it belongs to semi-supervised learning algorithm. This kind of learning hangs between either supervised or unsupervised learning. The foremost example for this kind of learning is archive image; where only a few images are archived with label like dog, cat etc.

Many world machine learning issues fall under this space. As a result, it can be a high-priced or long knowledge label which requires consultant domain access. Whereas unlabeled knowledge is affordable and simple for storing data.

Through input variables structure can be learned in unsupervised learning techniques to get and learn the structure within the input variables.

Additionally, utilization of supervised learning provides better predictions for unlabeled knowledge, knowledge retrieved to the supervised learning algorithmic rule as coaching knowledge model for prediction of new unseen knowledge.

3.4.1 Understanding Semi-supervised Learning

Semi-supervised learning (SSL) belongs to one among the unreal intelligence (AI) strategies that became in style within the previous few months. firms like Google are advancing the tools and frameworks relevant for building semi-supervised learning applications. Google Expander could be a nice example of a tool that reflects the advancements in semi-supervised learning applications.

Conceptually, semi-supervised learning will be positioned between unsupervised and supervised learning models. A semi-supervised learning downside starts with a series of tagged information moreover as some knowledge point where labels don't seem to be better-known. Semi-supervised learning model is the classification of unlabeled knowledge victimization to a tagged info set.

Semi-supervised learning algorithms square measure are trained on a mix of tagged and unlabeled knowledge. which can be helpful for a number of reasons. First, the method of labeling huge amounts of information for supervised learning is commonly prohibitively long and high-priced. What's "a lot of"? an excessive amount of labeling that will impose human biases on the model. This means together with various unlabeled knowledge throughout; the coaching method really tends to enhance the accuracy of the ultimate model whereas reducing the time and value spent building it.

Semi-supervised learning could be a win-win in cases like web page classification, speech recognition, or maybe for genetic sequencing. All told of those cases, knowledge scientists will access giant volumes of unlabeled knowledge, however the method of real distribution of superintendence info to any of it might be an insurmountable task.

Some AI practitioners see semi-supervised learning as a style of supervised learning with further info. At the end, the goal of semi-supervised learning models is to seem identical as supervised ones: to predict a target worth for a selected input file set. Alternatively, different segments of the AI community see semi-supervised learning as a style of unsupervised learning with constraints. You'll choose your favorite faculty of thought.

Semi-Supervised Learning within the world

Semi-supervised learning models have become wide applicable in situations across an oversized type of industry. Let's explore a few of the foremost well-known examples:

- **Speech Analysis:** Speech analysis could be a classic example of the worth of semi-supervised learning models. Labeling audio files usually could be a terribly intensive tasks that needs plenty of human resources. Applying SSL techniques will extremely facilitate to enhance ancient speech analytic models.
- **Protein Sequence Classification:** Inferring the operate of proteins usually needs active human intervention.
- **Web Content Classification:** Organizing the data obtainable from billions of web content can advance completely different segments of AI. Sadly, that task usually needs human intervention to classify the content.

There square measure many different situations for SSL models. However, not all AI situations will directly be tackled victimization SSL. There

square measure a number of essential characteristics that ought to be a gift on a retardant to be an effectively soluble victimization SSL.

A Semi-Supervised learning technique

We have made immense progress in the determination of supervised machine learning issues. That additionally implies that we want plenty of information to create our image classifiers or sales forecasters. The algorithms search for patterns through the information again and again.

But, that's not how a human mind learns. An individual's brain doesn't need a lot of knowledge for coaching with multiple iterations of browsing a similar image in order to understand a subject. All it wants could be a few guiding points to coach itself on the underlying patterns. Clearly, we tend to square measure missing one thing in the current machine learning approach.

Using classification as AN example, let's compare how these 3 approaches add practice:

- **Supervised classification:** The algorithmic rule learns to assign labels to forms of webpages supported by the labels that were input by an individual throughout the coaching method.
- **Unsupervised clustering:** The algorithmic rule appearance has inherent similarities between webpages to position them into teams.
- **Semi-supervised classification:** Labeled knowledge is employed to assist identify that their square measure specific teams of webpage varieties gift within the knowledge and what they might be. The algorithmic rule is then trained on unlabeled knowledge to outline the boundaries of these webpage varieties and will even establish new forms of webpages that were some within the existing human-input labels.

3.5 Reinforcement Learning

Introduction

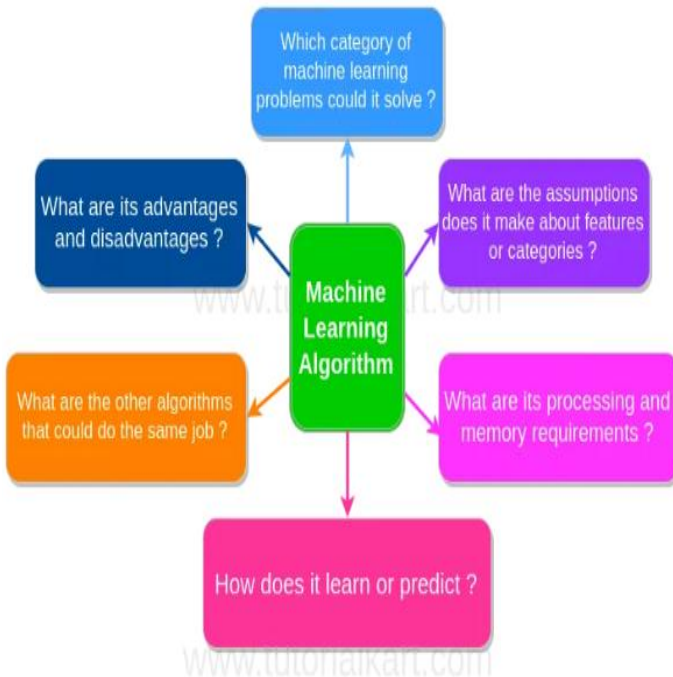
One of the foremost basic question for scientists across the world has been – “How to find out a brand-new skill?”. the will to perceive|to know|to grasp the solution is apparent – if we are able to understand this, we are able to modify the human species to try and do things we'd not have thought before. Alternately, we are able to train machines to try to do a lot

of “human” tasks and make true computing.

While we tend to not have a whole answer to the higher than question, nevertheless, there square measure a number of things that square measure clear. No matter the ability, we tend to 1st learn by interacting with the setting. Whether or not we tend to square measure learning to drive a automotive or whether or not it a baby learning to run, the training relies on the interaction with the setting. Learning from interaction is the fundamental underlying construct for all theories of learning and intelligence.

Human has ability to solve a broad range of problems starting from low-level motor control to higher-level cognitive tasks. Our final aim is to develop artificial agents with DeepMind for achieving generality and performance similarity. As like human our learning agent is able achieve strategies successfully for long-term rewards. In this scenario the trial-and-error approach with solely reward and punishment is stated as **reinforcement learning (RL)**.

3.6 Some Important Consideration in Machine Learning



3.7 Training and Validation

Cross-Validation Model to improve Assessing and Performance of Prediction

Cross-validation appraisal system is used for assessing a forecasting value of machine learning execution time for unprepared or completed data set. Here, an appropriate subset usage and dataset calculation are performed for testing information. Since cross-approval does not utilize the majority of the information to assemble a model, it is a generally utilized technique to avoid over fitting amid preparing.

Cross-validation performs original dataset partitioning at random as a *training set* and a *testing set*. In a supervised dataset, training is utilized to measure training performance. Effective performance indicators for cross-validation provide an average, and several times help in error calculation.

A few techniques involved in cross-validation are stated as follows:

- **K-fold:** In this subset, a choice is made randomly in to k subset in equal size. Remaining subset are validated with one subset for training model. For exact validation k -fold is presented for each subset to validate model.
- **Holdout:** Two subsets or folds are classified with equal length for validation and training purpose.
- **Leave-out:** Classification of k - mean approaches with same number of observations with cross-validation approach.
- **Repeated random sub-sampling:** For random partitioning of data Monte Carlo is performed for aggregation runs.
- **Stratify:** Classification of data, as training and testing are divided roughly into proportions of dataset of response and target.
- **Re-substitution:** Does not divide the information; but utilizes the preparation information for approval. Frequently delivers excessively idealistic evaluations for execution and must be maintained at a strategic distance from, if there is adequate information.

Cross-approval can be a computationally serious task for preparing and the approval is completed a few times. Since each parcel set is autonomous, this examination can be performed in parallel to accelerate the procedure.

3.8 Classification of Methods

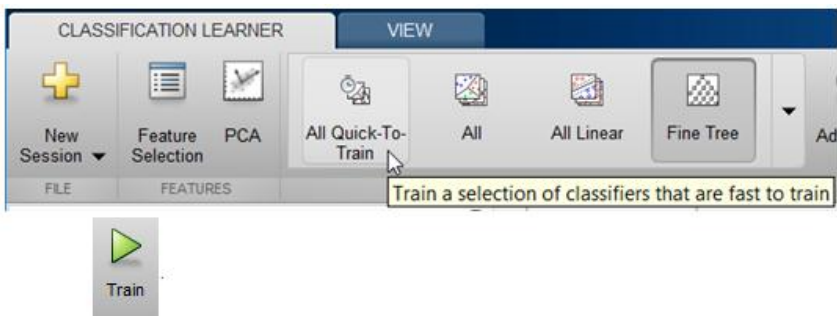
3.8.1 Training of Automated Classifier

Classifier learner is used to train data automatically for various classification models of data.

At once multiple models are utilized for automated training. For effective model interactive performance, model selection is explored.

In case classifier type is known to you already than train classifier individually. Refer **Classifier Training** by following the below steps:

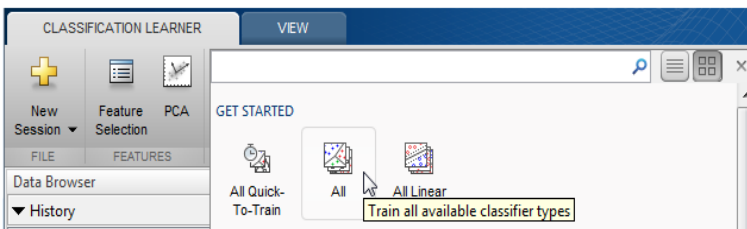
1. In the **Apps** tab of **Machine Learning** group select **Classification Learner**.
2. Then select **New Session** for selecting data from file or workspace. Select the response variable and predictors for the refer data selection and classification model selection.
3. In **Classification Learner** provides section **Model Type** and select **All Quick-To-Train**. This enables a model for fit and fast training option of data set.



4. Click **Train**. In history list model type a selection appears for complete training. For effective and more **Accuracy**, percentage values are highlighted.

▼ History		
1.1	☆ Tree Last change: Fine Tree	Accuracy: 94.0% 3/3 features
1.2	☆ Tree Last change: Medium Tree	Accuracy: 94.0% 3/3 features
1.3	☆ Tree Last change: Coarse Tree	Accuracy: 94.0% 3/3 features
1.4	☆ KNN Last change: Fine KNN	Accuracy: 90.0% 3/3 features
1.5	☆ KNN Last change: Medium KNN	Accuracy: 94.7% 3/3 features
1.6	☆ KNN Last change: Coarse KNN	Accuracy: 68.7% 3/3 features
1.7	☆ KNN Last change: Cosine KNN	Accuracy: 84.0% 3/3 features
1.8	☆ KNN Last change: Cubic KNN	Accuracy: 94.7% 3/3 features
1.9	☆ KNN Last change: Weighted KNN	Accuracy: 93.3% 3/3 features

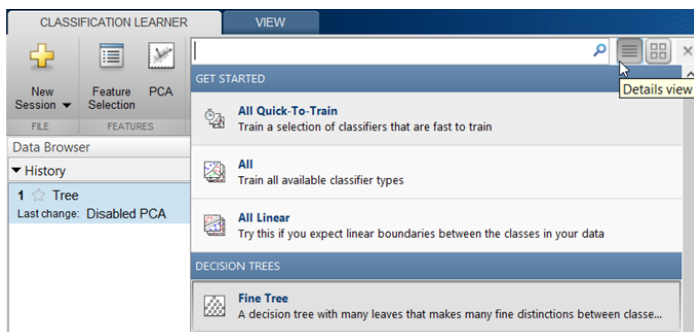
5. In history list select the required models for providing plots. In next stage, refer to **Classifier Training Manual** and **Model classification Improvement**.
6. For data set you need to provide model classifier of data set by clicking **All tab** and then click **Train**.



3.8.2 Manual Classifier Training

In the event that you need to investigate singular model writes, or in the event that you definitely recognize what classifier compose you need, you can prepare classifiers each one in turn, or prepare a gathering of a similar sort.

1. Selection of Classifier: On the **CLASSIFICATION LEARNER** tab, with **Model Type** section, click a classifier to write. To see all accessible classifier alternatives, tap the bolt on the rightmost corner of the **Model Type** section to expand the rundown of classifiers. The choices in the **Model Type** gallery provide various settings during the beginning stages, appropriate for a scope of various order issues. To peruse a portrayal of every classifier, change to the points of interest:



For more information on each option, see **Choose Classifier Options**.

2. After selecting a classifier, click **Train**. Repeat to try different classifiers.

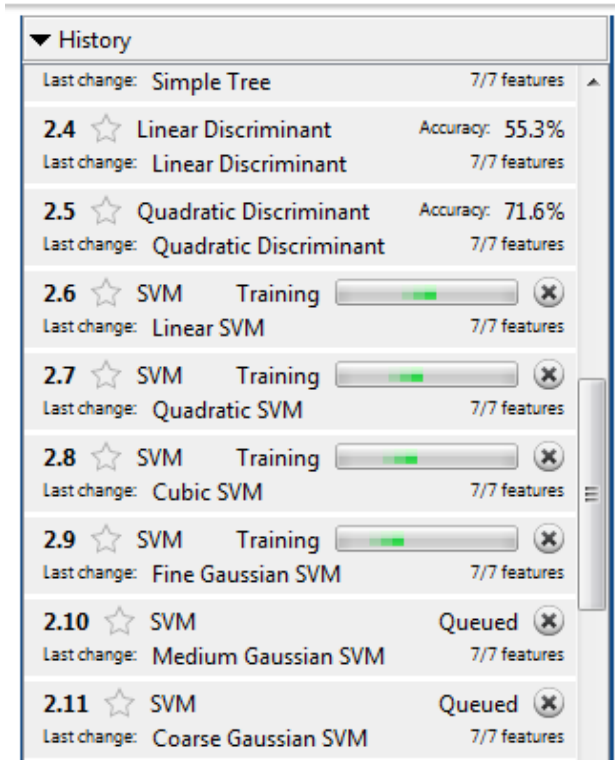


3. In case you want to check all types of models select the **All** option in **Model Type** gallery.

3.8.3 Parallel Classifier Training

For parallel preparation of models, utilize Classification Learner in the event that **Toolbox in Parallel** Computing. For classifier preparation, the application consequently begins laborers in parallel, for parallel preference it is in default sequentially in a parallel manner. In the event that a pool is as of now open, the application utilizes it for preparing. Parallel preparing enables you to prepare numerous classifiers on the double and keep working.

1. On the first occasion when you click Train, a trade application provides a parallel pool authorities. After opening of the pool, you can set up various classifiers immediately.
2. Parallel manner classifiers are prepared for improvement markers for individual preparation and lined model rundown history, which can be scratched off in the event that you need models. Amid preparing, analyze the results and plots from model, then start preparing them in classifiers.



For training in parallel select the tab **Use Parallel** in upper strip of the app.



3.9 Algorithm for Classification

3.9.1 Classification Algorithm in General

Classification belongs to a kind of administered machine learning where the calculation “learns” to arrange perceptions in marked information cases. To investigate grouping models intelligently, utilize the **Classification Learner** app. More noteworthy is its adaptability indicator which highlights information relating to reactions or names to a calculation fitting capacity in the summon line interface.

In machine learning and insights, characterization is a managed learning approach in which the PC program gains from the information input given to it and afterward utilizes this in figuring out how to group new perception. This informational collection may be bi-class (like distinguishing whether the individual is male or female or that the mail is spam or non-spam) or multi-class as well. A few cases of grouping issues are: discourse acknowledgment, penmanship acknowledgment, bio metric distinguishing proof, report order and so forth.

3.9.2 Common Classification Algorithm

Logistic Regression

Logistics regression is the go-to direct grouping calculation for two-class issues. It is anything but difficult to execute, straightforward, and gets extraordinary outcomes on a wide assortment of issues; notwithstanding when the desires the technique has of your information are damaged.

Calculated Machine Learning Regression characterization utilizes a calculation that foresees the likelihood clear cut variable. For a strategic relapse, needy variables are located in parallel with information coded as 1 (yes, achievement and so on.) or 0 (no, disappointment and so forth.). As such, the strategic relapse demonstrates the prediction $P(Y=1)$ with X as component.

Assumption of Logistic Regression

- Parallel calculated relapse requires the needy variable to be double.
- For a twofold relapse, the factor level 1 of the reliant variable ought to speak to the coveted result.
- Only the important factors ought to be incorporated.

- The autonomous factors ought to be free of each other. That is, the model ought to have practically zero multi co-linearity.
- The autonomous factors are straightly identified with the log chances.
- Logistic relapse requires very extensive example sizes.

Remembering the above presumptions, we should take a gander at our dataset.

Logistic Regression with Stochastic Gradient Descent

Logistic regression is named for the capacity utilized center strategy, that is its calculated capacity. Calculated relapse utilizes a condition as the portrayal, particularly like direct relapse. Information esteems (X) are joined directly utilizing weights or coefficient esteems to foresee a yield esteem (y). In key contrast from straight relapse, we yield an esteem demonstrated paired value (0 or 1) as opposed to numeric esteem.

$$\hat{y} = e^{(b_0 + b_1 * x_1)} / (1 + e^{(b_0 + b_1 * x_1)})$$

Where e is represented as Euler's number count or common log value, \hat{y} refers to the output of the predicted value, b_0 defines predisposition to capture term variable and b_1 defined as the coefficient of a single information esteem (x_1).

The \hat{y} expectation provides a genuine incentive in the vicinity of 0 and 1 that should be adjusted to a number esteem and mapped to an anticipated class esteem.

Every segment in your information has a related b coefficient (a steady genuine esteem) that must be gained from your preparation information. The genuine portrayals of the model for storing memory in a document format coefficients condition (the beta esteem or b 's). The calculated coefficient relapses the calculation to be assessed in the preparation information.

Stochastic Gradient Descent

Gradient Descent provides an optimal way toward limiting capacity with inclinations towards cost work.

It includes a type of cost belonging and additional subsidiary required for an angle in given point and a path to move toward points, e.g. the base value of downhill. Machine learning utilize a method for assessment and coefficient update in each emphasis of stochastic slope drop limit blunder to prepare the information model.

In the advancement calculation, individual preparation case appeared in the model in turn. The model influences a forecast for a preparation to occur, the blunder is ascertained, and the model is refreshed with a specific end goal to lessen the mistake for the following expectation.

This strategy utilized for arrangement location in the model, coefficients the resultant outcomes' littlest mistake preparation information model. After every cycle, machine learning coefficients (b) dialect is refreshed utilizing condition:

$$b = b + \text{learning_rate} * (y - \text{yhat}) * \text{yhat} * (1 - \text{yhat}) * x$$

In the above equation **b** is defined as the optimized weight or coefficients; **learning_rate** states the configuration of learning rate (e.g. 0.01); **(y - yhat)** provides training data model for error prediction weight attributes; **yhat** represent coefficients of prediction and **x** defines the input value .

```
%% Logistic Regression
```

```
%% Initialization
```

```
clear ; close all; clc
```

```
%% Load Data
```

```
% The first two columns contains the exam scores and the third column  
% contains the label
```

```
data = load('inputTrainingSet1.txt');
```

```
X = data(:, [1, 2]); % X is a #OfExamScores x 2 matrix
```

```
y = data(:, 3); % y is a #OfExamScores x 1 matrix
```

```
% It's always a good idea to first plot the problem to solve  
fprintf(['Plotting data with + indicating (y = 1) examples and o ' ...  
        'indicating (y = 0) examples.\n']);
```

```
%plotData(X, y);
```

```
plot(X, y);
```

```
% Put some labels
```

```
hold on;
```

```
% Labels and Legend
```

```
title('Acceptance of Previous Students Based on Scores on 2 Exams')
```

```
xlabel('Exam 1 score')
```

```
ylabel('Exam 2 score')
```

```
% Specified in plot order
```

```
legend('Admitted Students', 'Not admitted Students')
```

```
hold off;
```

```

fprintf('\nProgram paused. Press enter to continue.\n');
pause;

%% ===== Compute Cost and Gradient =====
% implement the cost and gradient for logistic regression

% Setup the data matrix appropriately, and add ones for the intercept term
[m, n] = size(X);

% Add intercept term to x and X_test
X = [ones(m, 1) X];

% Initialize fitting parameters
initial_theta = zeros(n + 1, 1);

% Compute and display initial cost and gradient
[cost, gradient] = costFunction(initial_theta, X, y);

fprintf('Cost at initial theta (zeros): %f\n', cost);
fprintf('Gradient at initial theta (zeros): \n');
fprintf(' %f \n', gradient);

fprintf('\nProgram paused. Press enter to continue.\n');
pause;

%% ===== Optimizing using fminunc =====
% use a built-in function (fminunc) to find the
% optimal parameters theta. Octave's fminunc is an optimization
% solver that finds the minimum of an unconstrained function.
% For logistic regression, you want to optimize the cost
% function J(theta) with parameters theta.

% Set options for fminunc
options = optimset('GradObj', 'on', 'MaxIter', 400);

% Run fminunc to obtain the optimal theta
% This function will return theta and the cost
[theta, cost] = fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);
% NOTE: that by using fminunc, you do not have to write any loops yourself,
% or set a learning rate like you did for gradient descent. You ONLY need
% to provide a function calculating the cost and the gradient.

% Print theta to screen
fprintf('Cost at theta found by fminunc: %f\n', cost);
fprintf('theta: \n');
fprintf(' %f \n', theta);

% Plot Boundary
plotDecisionBoundary(theta, X, y);

% Put some labels
hold on;

```

```

% Labels and Legend
xlabel('Exam 1 score')
ylabel('Exam 2 score')

% Specified in plot order
legend('Admitted', 'Not admitted')
hold off;

fprintf('\nProgram paused. Press enter to continue.\n');
pause;

%% ===== Part 4: Predict and Accuracies =====
% After learning the parameters, you'll like to use it to predict the outcomes
% on unseen data. In this part, you will use the logistic regression model
% to predict the probability that a student with score 45 on exam 1 and
% score 85 on exam 2 will be admitted.
%
% Furthermore, you will compute the training and test set accuracies of
% our model.
%
% Your task is to complete the code in predict.m

% Predict probability for a student with score 45 on exam 1
% and score 85 on exam 2

prob = sigmoid([1 45 85] * theta);
fprintf(['For a student with scores 45 and 85, we predict an admission ' ...
        'probability of %f\n\n'], prob);

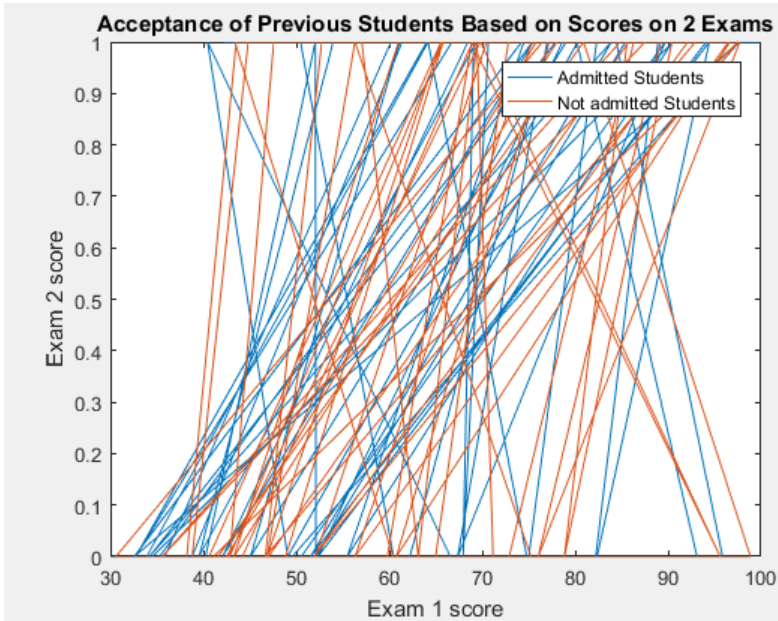
% Compute accuracy on our training set
p = predict(theta, X);

fprintf('Train Accuracy: %f\n', mean(double(p == y)) * 100);

fprintf('\nProgram paused. Press enter to continue.\n');
pause;

```

Output



K-Nearest Neighbors

The k-Nearest Neighbors (or kNN for short) provides calculation in a simpler manner utilized to comprehend and actualize. The usage will be particular for order issues and will be shown utilizing the Iris blossoms characterization issue.

What is k-Nearest Neighbors

The model for kNN is the whole preparing dataset. At the point when a forecast is required for an inconspicuous information occasion, the kNN calculation will look through the preparation dataset for the k-most comparable occurrences. The expectation characteristic of the most comparable occurrences is condensed and returned as a forecast for the inconspicuous occasion.

The likeness measure is subject to the sorting of information. For genuine esteemed information, the Euclidean separation can be utilized. For different sorts of information, for example, clear cut or double information, Hamming separation can be utilized.

Performance of k-Nearest Neighbors Works

The kNN calculation has a place with the group of case based, focused learning and languid learning calculations.

Case based calculations are those calculations that model the issue utilizing information cases (or columns) with a specific end goal to settle on prescient choices. The kNN calculation is an extraordinary type of occasion-based strategy since all preparation perceptions are held as a major aspect of the model.

It is an aggressive learning calculation, since its inner utilizations rival between demonstrate components (information examples) keeping in mind the end goal to settle on a prescient choice. The target closeness measure between information cases causes every datum example to contend to “win”; or most likely be a given inconspicuous information occasion and add to an expectation.

Lethargic learning alludes a way for calculating a forecasting assembled model. On the grounds that it just works in the end. It has an advantage of information importance for the desired information also known as restricted model.

Lastly kNN, in light of the fact that it doesn't expect anything about the information other than a separation measure, is capable in figuring reliably between any two cases. In that capacity, it is called non-parametric or non-straight as it doesn't expect a practical frame.

Program

```

% This program is an example for implementaion of KNN Algorithm.
% Inputs are temperature, wind, humidity and precipitation values taken
% from http://www.timeanddate.com/weather/india/new-delhi/hourly for 23
|
k = 7; % Number of nearest neighbours we want to evaluate to find the result of current search query.

qTemp    = 30;
qWind    = 12;
qHumidity = 72;
query    = [qTemp qWind qHumidity];

temp     = xlsread('E:\Data Analytics\Data science\pauls\pattern\Machine_learning\MachineLearning\Machi
wind     = xlsread('E:\Data Analytics\Data science\pauls\pattern\Machine_learning\MachineLearning\Machi
humidity = xlsread('E:\Data Analytics\Data science\pauls\pattern\Machine_learning\MachineLearning\Machi
precipitation = xlsread('E:\Data Analytics\Data science\pauls\pattern\Machine_learning\MachineLearning\Machi

%Start(0) Calculating Euclidean Distance between the query points and
%previous data.

z1 = (query(1) - temp).^2;
z2 = (query(2) - wind).^2;
z3 = (query(3) - humidity).^2;

euclideanDistance = z1 + z2 + z3; %Euclidean distance in square units

%End (0)

distance = [euclideanDistance precipitation]; % Appending the output (precipitation) vector to distance.
sortedDistance = sortrows(distance); % sort the distance vector based on the proximity to search query.

precipitation = mode(sortedDistance(1:k,2)); % Find the most frequently occuring class value( here, precipitation)

if(precipitation == 1)
    disp('High')
else
    disp('Low')
end

```

The above algorithm has the following results

Output

```

>> KNN
Low
SVM

```

This is an implementation of the SVM algorithm. To do this, I solve the dual L1-regularized and kernelized optimization problem via classic QP using CVX and (in the future) via the SMO algorithm.

Neural Networks

Pattern Recognition

In neural network, training for inputs and classes are stated as follows:

Notwithstanding capacity fitting, neural systems are additionally great at perceiving designs.

For instance, assume you need to arrange a tumor as generous or threatening, in light of the consistency of cell measure, cluster thickness, mitosis and so on. You have 699 cases for which you have 9 things of information and the right order as favorable or dangerous.

Likewise with work fitting, there are two approaches to take care of this issue:

Utilization of nprtool GUI, depicted with in use of Neural Network Pattern Recognition App.

Utilize an order line arrangement, as depicted in Using Command-Line Functions.

Defining a Problem

To define an example acknowledgment issue, mastermind an arrangement of Q input vectors as segments in a lattice. At that point, mastermind another arrangement of vector as Q to demonstrate the information vector allocated class.

At this point it contains two information classes; where every target in scalar form provides an incentive as 1 or 0, showing that class comparison input has a place. For example, you can characterize the two-class restrictive or arrangement issues to be taken after:

```
inputs = [0 1 0 1; 0 0 1 1];  
targets = [1 0 0 1; 0 1 1 0];
```

Input is classified into classes with N different values, where target elements have N vector. The target vector contains two elements such as 0 and 1. In following example classification problem defines cube division in 3 classes:

- The class of origin with input vector first.
- In second the class, origin are farthest from the corner with the input vector at last.
- In third the class all points lie in the same line.

```
inputs = [0 0 0 0 5 5 5 5; 0 0 5 5 0 0 5 5; 0 5 0 5 0 5 0 5];
targets = [1 0 0 0 0 0 0 0; 0 1 1 1 1 1 1 0; 0 0 0 0 0 0 0 1];
```

Generally the classification problem belongs to either format classes. For this, the target value consists of two-element vector (0 or 1), where it has any one element either 0 or 1.

Using Command-Line Functions

```
% Solve a Pattern Recognition Problem with a Neural Network
% Script generated by NPRTOOL
%
% This script assumes these variables are defined:
%
%   cancerInputs - input data.
%   cancerTargets - target data.

inputs = cancerInputs;
targets = cancerTargets;

% Create a Pattern Recognition Network
hiddenLayerSize = 10;
net = patternnet(hiddenLayerSize);

% Set up Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)
```

```

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
% figure, plotperform(tr)
% figure, plottrainstate(tr)
% figure, plotconfusion(targets,outputs)
% figure, ploterrhist(errors)

```

1. In MATLAB, the scripts are already loaded with input and target vector in workplace. In case those vectors are loaded previously then with the following function it can be loaded:

```
[inputs, targets] = cancer_dataset;
```

2. Make the system. Default arrange work fitting (or relapse) problems; patternnet is a feedforward coordinate to move the work concealed layer and softmax to move work yield layer. Relegated ten neurons (fairly discretionary) shrouded layer in the past area.
 - The organized result has two yield neurons, in light of the fact of 2 target element esteems (classes) related to information vector.
 - Every neuron in the vector communicates with class.
 - An information vector with suitable class will be connected with the system, and relating neurons need to provide the resultant vector as a 1 with alternate neurons yield a 0.

In system make following changes in script

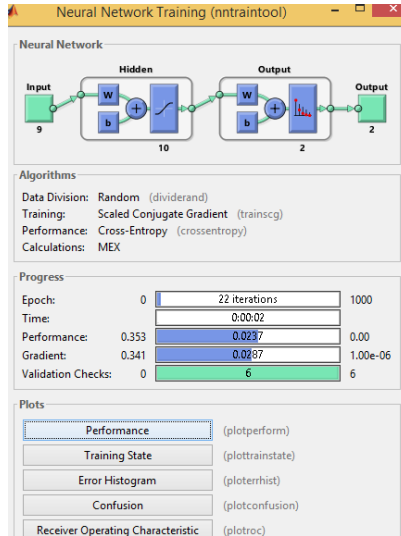
```
hiddenLayerSize = 10;
net = patternnet (hiddenLayerSize);
```

1. Set up the division of data

```
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
```
2. Network Training: The example acknowledgment organized utilizes the default Scaled Conjugate Gradient (traincsg) calculation for preparing. To prepare the system enter this summon.

```
[net, tr] = train(net, inputs, targets);
```

In the data training phase, for better fitting of variables open a window for preparation, this enable you to look into the progress of preparation. In case if you want to stop training select **Stop Training** at any stage.



This preparation halted when the approval blunder expanded for six cycles, which happened at emphasis 24.

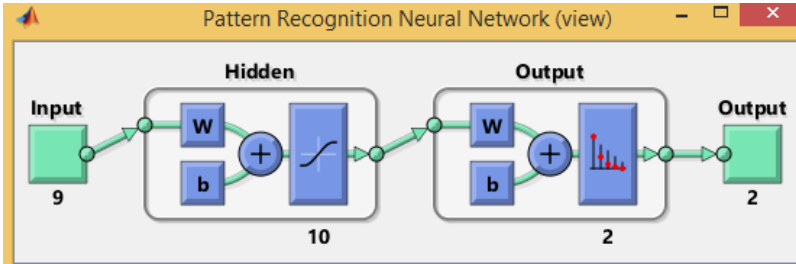
5. System Testing: After the system has been prepared, you can utilize it to figure the system yields. The accompanying code figures the system yields, mistakes, and general execution.

```
outputs = net(inputs);
errors = gsubtract(targets, outputs);
performance = perform(net, targets, outputs)
```

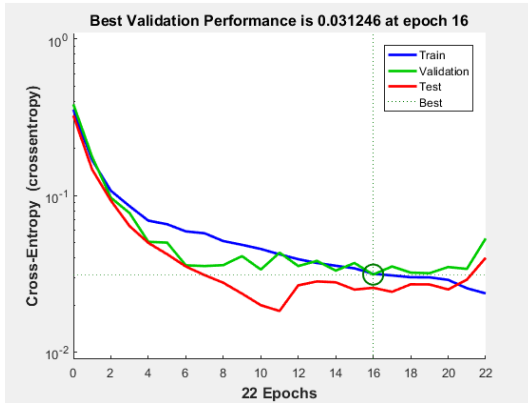
It is likewise conceivable to figure the system execution just on the test set, by utilizing the testing files, which are situated in the preparation record.

```
tInd = tr.testInd;
tstOutputs = net(inputs(:, tInd));
tstPerform = perform(net, targets(:, tInd), tstOutputs)
```

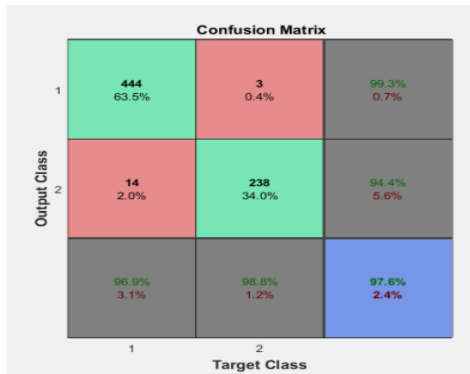
6. Network diagram view
view(net)



- Plotting performance of testing, training, and validation.
`figure, plotperform(tr)`



- Using a confusion matrix plot the values to evaluate error in final stage of training network.
`figure, plotconfusion (targets, outputs)`



The corner to corner cells provides the quantity of significantly clustered information and through cell inclination provides wrongly classified data cases. In the above figure the blue cell presents the accuracy of aggregation and the green denotes accurately arranged data. Finally, the red color cell represents mis-classified or wrongly classified data. On the off chance that you require significantly more precise outcomes, which is accompanied with significant methodologies:

- Modify the underlying system weights and inclinations of qualities with init and prepare once more.
- Increase shrouded neurons quality.
- Prepare vectors quality increment.
- Information esteems quality enhancement, in cases where more pertinent data is accessible.
- Try an alternate preparing calculation (see “Training Algorithms”).

For this situation, the system reaction is attractive and you would now be able to put the system to use on new sources of information.

For getting detailed information related to order line activities, few assignments can be attempted:

- For data preparation create a window to plot (for example perplexity plot) which is used effectively.
- Line summon plotting with function capabilities `plotroc` and `plottrainstate`.

Likewise, see the propelled content for more alternatives, when preparing from the charge line.

Each time a neural system is prepared, it can bring about an alternate arrangement because of various starting weight and inclination esteems and distinctive divisions of information into preparations, approvals and test sets.

Naïve Bayes

Naive Bayes model combined with predictors of Gaussian, multinomial or kernel predictors.

Naive Bayes models expects perceptions with multivariate dispersion for class enrollment, yet indicator or highlights making perception are free.

This system follows a total list of capabilities with the end goal that a perception is an arrangement of multinomial tallies.

To prepare an innocent Bayes show, use `fitcnb` in the order line interface. In the wake of preparing, anticipate marks or gauge back model probability and information prediction model.

The guileless Bayes classifier is intended for utilization when indicators are free of each other inside each class, yet it seems to function admirably by and by notwithstanding when that autonomy supposition isn't substantial. It groups information in two stages:

1. Training stage: Using the preparation information, the strategy evaluates the parameters of a likelihood appropriation, expecting indicators are restrictively autonomous class.
2. Prediction stage: In concealed test information, strategy figures the back likelihood example having a place with each class. The strategy at that point characterizes the test information concurring the biggest back likelihood.

The class-restrictive freedom supposition extraordinarily improves the preparation venture and gauge the one-dimensional class-contingent thickness of every indicator independently. While the class-contingent autonomy indicator isn't valid, when all is said in done inquire about demonstrates that this hopeful suspicion functions admirably by and by. This suspicion of class-restrictive freedom of the indicators enables the innocent Bayes classifier to evaluate the parameters required for the precise order utilizing minimal preparing information with numerous different classifiers. This makes it especially viable for informational indexes containing numerous indicators.

Supported Distributions

The preparation venture in gullible Bayes arrangement depends on estimating $P(X|Y)$, with predictors X which lies in class Y . Naive Bayes order model **Classification NaiveBayes** and preparing function `fitcnb` provide a bolster for a typical (Gaussian), piece, multinomial and multivariate, multinomial indicator contingent conveyances. To determine dispersions for the indicators, utilize the Distribution Names name-esteem matching the contention of `fitcnb`. You can indicate one sort of appropriation for all indicators by providing the character vector relating to the circulation name; or determine distinctive disseminations for the indicators by providing a length D cell exhibit of character vectors, where

D is the quantity of indicators (that is, the quantity of segments of X).

Normal (Gaussian) Distribution

The ‘normal’ distribution (determine using ‘normal’) has proper indicators with ordinary conveyances class. Every indicator demonstrates a typical dispersion, the innocent Bayes classifier assesses different ordinary appropriation in every class by processing mean and standard deviation of information class creation.

Kernel Distribution

The ‘kernel’ distribution (determine using <kernel>) suitable indicators consistent appropriation. Solid presumption, for example, an ordinary appropriation and you can utilize situations circulation where an indicator might skew or modes. Every indicator for which you display piece conveyance, guileless Bayes classifier figures a different part thickness gauge for each class in light of the preparation information for that class. As a matter of course the portion is the typical part of the classifier that chooses width naturally for class and indicator. Product underpins determine diverse bits for every indicator and distinctive widths for every indicator or class.

Multivariate Multinomial Distribution

The multivariate, multinomial conveyance (indicate ‘mvmn’) an indicators perceptions clearly. Innocent Bayes classifier development utilizing multivariate multinomial indicator is depicted beneath. An illustrations perception marked 0, 1 or 2, indicate the climate example directed.

1. Particular classifications speak to the perceptions of the whole indicator. For instance, particular classifications (or indicator levels) may incorporate bright, rain, snow, and overcast.
2. For the perceptions reaction class, isolate perceptions named 0 from perceptions named 1 and 2 and perceptions named 1 from perceptions named 2.
3. For each response class, fit a multinomial model using the characterization relative count and the total number of observations. For example, for observations named 0, the evaluated probability splendid $psunny_0 = (\text{where brilliant discernments with name } 0) / (\text{number of recognitions check } 0)$ and tantamount interchange of classes and response names.

The class-restrictive, is where multinomial irregular factors include a multivariate multinomial arbitrary variable. It includes some unique properties straightforward from Bayes classifiers for multivariate multinomial.

- Every indicator displays a multivariate multinomial dispersion, credulous Bayes classifier.
- Records different arrangement-particular indicator levels for every indicator
- Computes different arrangement probabilities indicator levels in each class.

In programming underpins display nonstop indicators as multivariate multinomial. For a given situation, indicators are the unmistakable events in an estimation. An indicator represents numerous indicator levels. It is great practice to discretize such indicators.

On the off chance that an observation is an arrangement of achievements for different classes (spoken to by the greater part of the indicators) out of a settled number of free trials, at that point we determine that the indicators involve a multinomial dispersion. For points of interest, Refer Multinomial Distribution.

Multinomial Distribution

The multinomial conveyance (determined using ‘DistributionNames’, ‘mn’) is fitting when, given the class each observation is a multinomial arbitrary variable. That is, perception, or row, j of the indicator data X represents D categories, where x_{jd} is the quantity of achievements for classification (i.e., predictor) d in $n_j = \sum_{d=1}^D x_{jd}$ independent trials. The means to prepare a guileless Bayes classifier are sketched out straightaway.

1. For each class, fit a multinomial dispersion for the indicators given its class. We do this by:
 - a. Aggregating the weighted, classification checks over all perceptions. Moreover, the product actualizes added substance smoothing.
 - b. Estimating the D category probabilities inside each class utilizing the totaled classification checks. These classification probabilities form the likelihood parameters out of the multinomial dispersion.

2. Let another perception have an aggregate tally of m . At that point, the innocent Bayes classifier:
 - a. Sets the aggregate check parameter of every multinomial dissemination to m .
 - b. For each class, gauges the class back likelihood utilizing the evaluated multinomial conveyances.
 - c. Predicts the perception into the class comparing with the most elevated back likelihood.

Consider the alleged sack of-tokens display, where there is a sack containing various tokens of different sorts and extents. Every indicator speaks to a particular kind of token taken care of; a perception has n independent draws (i.e., with substitution) of tokens from the sack and the information is a vector of checks, where the element d is the circumstances where token d appears.

A machine-learning application is the development of an email spam classifier, where every indicator speaks to a word, character, or expression (i.e., token), a perception is an email and the information included of the tokens in the email. One indicator may check the quantity of outcry focuses, another might tally the circumstances “cash” that shows up and another might tally the circumstances the beneficiary’s name shows up. In Bayes display for further supposition, aggregate token count (or record length aggregated) is of reaction class.

The different properties of guileless Bayes classifiers that utilization multinomial perceptions include are mentioned below:

·Classification depends on the relative frequencies of the classes. If $n_j = 0$ for observation j , at that point arrangement isn’t workable for that perception.

- The indicators are not restrictively autonomous since they should aggregate to n_j .
- Naive Bayes isn’t proper when n_j provides data about the class. That is, this classifier requires that n_j is free of the class.
- If you determine that the indicators are restrictively multinomial, at that point the product applies this detail to all indicators. As such, you can’t include ‘mn’ in a cell exhibit when specifying ‘DistributionNames’.

In the event that a predictor is straight out, i.e., it is multinomial inside a reaction class, at that point determine that it is multivariate multinomial. For subtle elements, see Multivariate Multinomial Distribution.

fitcnb

Naive Bayes model multiclass training

Train Naive Bayes Classifier

In first stage, load the dataset of Fisher's iris.

Program

```
loadfisheriris
X = meas( : , 3:4 ) ;
Y = species;
tabulate (Y)
```

```
>>naive_bayes
```

Value	Count	Percent
setosa	50	33.33%
versicolor	50	33.33%
virginica	50	33.33%

The programming can order information with an excess of two classes utilizing credulous Bayes techniques.

Prepare a gullible Bayes classifier. It is a great practice to determine the class arrangement.

```
Mdl =
```

```
ClassificationNaiveBayes
    ResponseName: 'Y'
    CategoricalPredictors: []
    ClassNames: {1x3 cell}
    ScoreTransform: '...'
    NumObservations: 150
    DistributionNames: {1x2 cell}
    DistributionParameters: {3x2 cell}
```

Properties, Methods

Mdl is a trained Classification Naive Bayes classifier.

As a matter of process, the product models the indicator conveyance inside each class utilizing a Gaussian dispersion having some mean and standard deviation. Utilize spot documentation to show the parameters of a specific Gaussian fit, e.g., show the fit for the main element inside setosa.

Program

```
setosaIndex = strcmp (Mdl . ClassNames, ' setosa ' ) ;
estimates = Mdl . DistributionParameters{setosaIndex,
1}

estimates =
    1.4620
    0.1737
```

In the above program 1.4620 is obtained as mean value and 0.1737 is the standard deviation.

Gaussian contours plot

Program

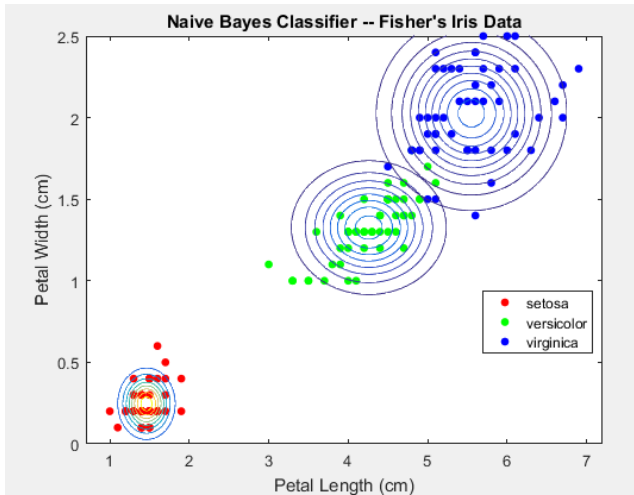
```
figure
gscatter (X( : , 1 ) ,X( : , 2 ) ,Y) ;
h = gca;
xlim = h.XLim;
ylim = h.YLim;
hold on
Params = cell2mat (Mdl . DistributionParameters) ;
Mu = Params (2* (1:3)-1, 1:2) ; % Extract the means
Sigma = zeros (2, 2, 3) ;
for j = 1:3
    Sigma (: , : , j) = diag (Params(2*j, :) ) . ^2;
    Create diagonal covariance matrix
    xlim = Mu (j, 1) + 4*[1 -1]*sqrt (Sigma (1, 1,
j) ) ;
    ylim = Mu (j, 2) + 4*[1 -1]*sqrt (Sigma (2, 2, j) ) ;
    ezcontour (@(x1, x2) mvnpdf ([x1, x2] ,Mu (j, :) ,
Sigma (:, :, j) ) , [xlimylim] )
    % Draw contours for the multivariate normal
distributions
```

```

end
h.XLim = cxlim;
h.YLim = cylim;
title ('Naïve Bayes Classifier --Fisher's' 's Iris Data')
xlabel ('Petal Length (cm)')
ylabel ('Petal Width (cm)')
hold off

```

Output



Discriminant Analysis Classifiers

- To prepare the classifier, check the capacity of appraises in the Gaussian parameter fitting circulation in the class.
- To anticipate new information classes in this created classifier identify minimal cost of misclassification.

Create Discriminant Analysis Classifiers

This case demonstrates to prepare a fundamental discriminant investigation classifier to order irises in Fisher's iris information.

Data loading as function

```
load fisheriris
```

Create a default (linear) discriminant analysis classifier

```
MdlLinear = fitcdiscr(meas,species);
```

To envision the grouping limits of a 2-D order of the information, see [Create and Visualize Discriminant Analysis Classifier](#).

Characterize an iris with normal estimations.

```
meanmeas = mean(meas);
meanclass = predict(MdlLinear, meanmeas)
```

```
>> discriminant_analysis
meanclass =
    'versicolor'
```

Create a quadratic classifier

```
MdlQuadratic = fitcdiscr(meas, species, 'DiscrimType', 'quadratic');

meanclass2 =
    'versicolor'
```

Train Discriminant Analysis Model

Dataset for loading Fisher's iris.

```
load fisheriris
```

Discriminant analysis model for entire dataset training.

```
Mdl = fitcdiscr(meas, species)

Mdl =

    ClassificationDiscriminant
        ResponseName: 'Y'
    CategoricalPredictors: []
        ClassNames: {1x3 cell}
        ScoreTransform: 'none'
    NumObservations: 150
        DiscrimType: 'linear'
                Mu: [3x4 double]
                Coeffs: [3x3 struct]
```

[Properties](#), [Methods](#)

To get to its properties, utilize dab documentation. For instance, show the gathering implied for every indicator.

`Mdl.Mu`

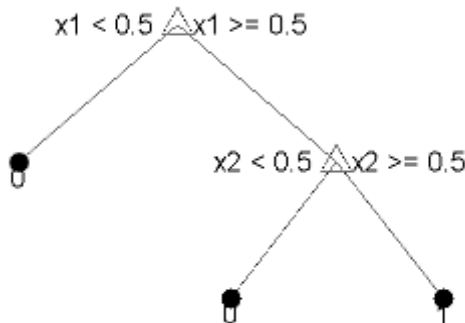
`ans =`

5.0060	3.4280	1.4620	0.2460
5.9360	2.7700	4.2600	1.3260
6.5880	2.9740	5.5520	2.0260

Decision Tree

Selection of trees and arrangement relapse trees help anticipate reactions information. Foresee reaction, take after the choices in the tree from the root (beginning) center point down to a leaf center point. The leaf center point contains the response. Course of action trees give responses that are apparent, such as 'true' or 'false'. Backslide trees give numeric responses.

Each progression in a forecast includes checking the estimation of one indicator (variable). For instance, here is a basic grouping tree:



This tree predicts orders in light of two predictors, x_1 and x_2 . To anticipate, begin at the best hub, spoken to by a triangle (Δ). The primary choice is whether x_1 is lesser than 0.5. Assuming this is the case, take after the left branch and see that the tree characterizes the information as type 0. On the

off chance that x_1 exceeds 0.5, at that point take after the correct branch to the lower-right triangle hub. Here the tree asks if x_2 is littler than 0.5. Assuming this is the case, at that point take after the left branch to see that the tree groups the information as type 0. If not, at that point take after the correct branch to see that the tree groups the information as type 1.

Classification Tree Training

Classification tree creation with ionosphere dataset.

```
load ionosphere % Contains X and Y variables
Mdl = fitctree(X,Y)
```

Mdl =

```
ClassificationTree
      ResponseName: 'Y'
      CategoricalPredictors: []
      ClassNames: {1x2 cell}
      ScoreTransform: '...'
      NumObservations: 351
```

Properties, Methods

Train Regression Tree

Make a relapse tree utilizing all perceptions in the carsmall data set. Think about the Horsepower and Weight vectors as indicator factors and the MPG vector as the reaction.

Superclasses: CompactClassificationTree

Binary decision tree for classification

Grow Classification Tree

Grow a classification tree using the ionosphere dataset.

```

load ionosphere
tc = fitctree(X,Y)

tc =

ClassificationTree
    ResponseName: 'Y'
    CategoricalPredictors: []
    ClassNames: {1x2 cell}
    ScoreTransform: 'none'
    NumObservations: 351

```

Properties, Methods

Estimations profundity for developing tree characterization are defined below:

- n - 1 for Maximum Number Splits. n is the preparation test measure.
- 1 for Minimum Leaf Size.
- 10 for Minimum Parent Size.

Esteems have a tendency to develop profound trees for extensive preparing test sizes. Prepare an order tree utilizing the default esteems for tree profundity control. Cross approve the model utilizing 10-overlap cross approval.

```

rng(1); % For reproducibility
MdlDefault = fitctree(X,Y,'CrossVal','on');

```

Program

```

numBranches = @ (x) sum (x. IsBranch) ;
mdlDefaultNumSplits = cellfun (numBranches, MdlDefault
. Trained) ;

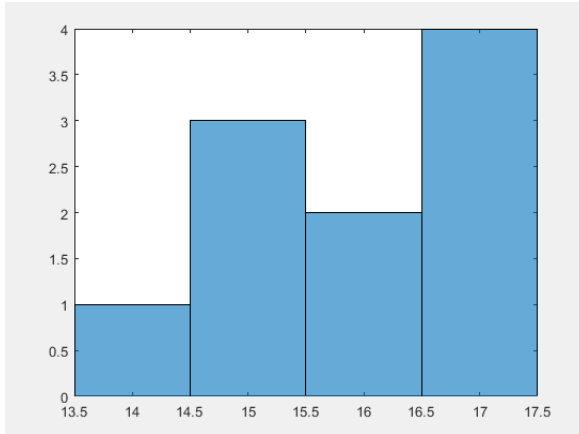
```

```

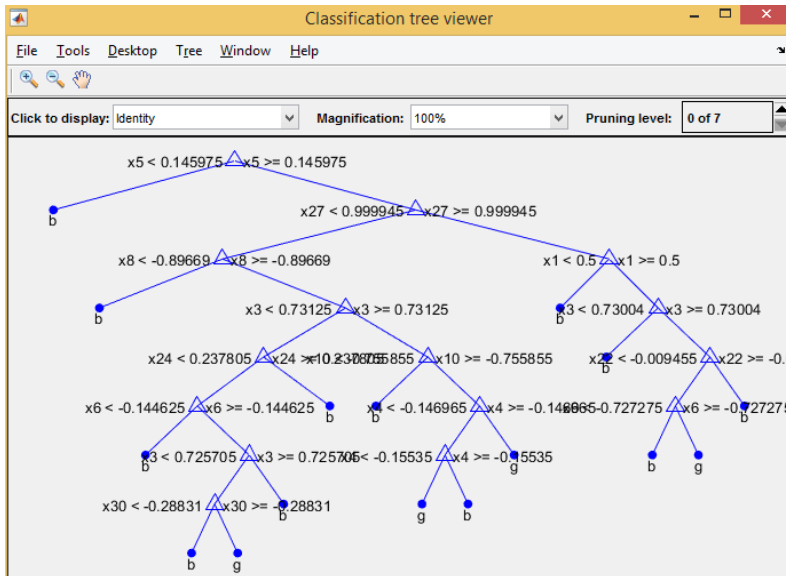
figure;
histogram(mdlDefaultNumsplits)

```


Output



view (MdlDefault.Trained{1}, 'Mode', 'graph')

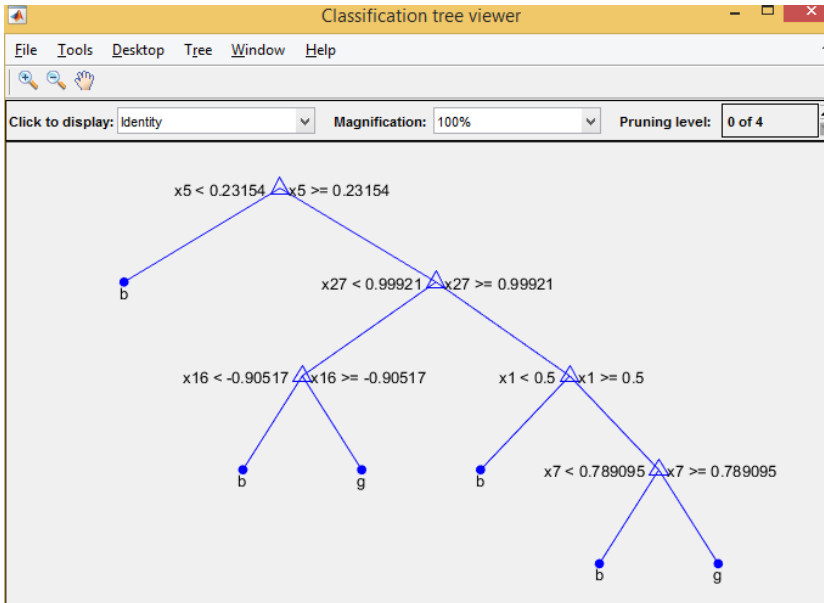


For each split the average count is 15.

Assume you need the characterization tree mind boggling (profound) prepared, utilizing parts number default. Prepare the characterization tree, however most extreme of 7 parts, which is about a large portion of the parts mean with arrangement tree in default. Cross this approve utilizing

10-overlap cross approval.

```
Mdl7 = fitctree(X,Y, 'MaxNumSplits',7, 'CrossVal', 'on');
view(Mdl7.Trained{1}, 'Mode', 'graph')
```



Error comparison of cross validation models.

```
classErrorDefault = kfoldLoss (MdlDefault)
classErrorDefault = 0.1140
classError7 = kfoldLoss (Mdl7)
classError7 = 0.1254
```

3.9.3 Regression

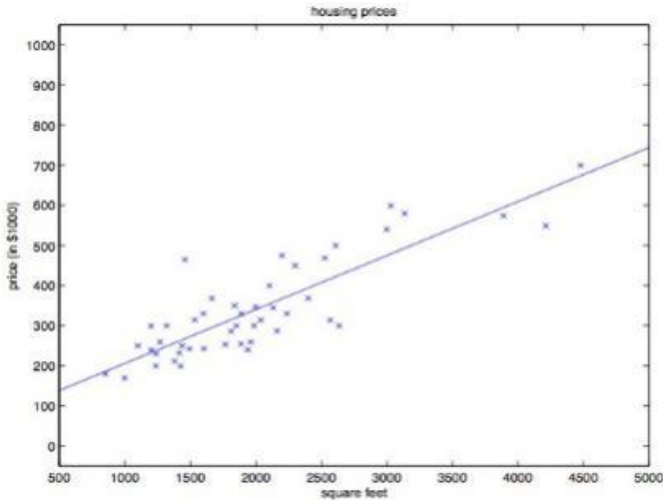
Linear, generalized linear, nonlinear and nonparametric techniques are used for supervised learning.

A simple example on how to perform a univariate logistic regression in MATLAB

Linear Regression

When given some variables X and corresponding results Y, linear regression is the approach we use to find or guess the relationship between X and Y.

If X contains only uni-dimensional variables then linear regression can be represented by the following graph, which we all very familiar with.



Every time someone talks about linear regression, they are most likely to say that “If your friend wants to sell a house...”. Given all the points above, we can guess the relationship between the area of house and the price of house using linear regression.

Furthermore, when our X is multi-dimensional, especially more than 2-dimensions, it will be less likely to represent the problem through a graph, but the problem is still similar with the 1-d problem, nothing but guess the θ s in $\theta_0 + X_1\theta_1 + X_2\theta_2 + \dots + X_j\theta_j = Y$, where j is the dimension of X .

SVM

Support Vector Machines for Binary Classification

Understanding Support Vector Machines

Separable Data

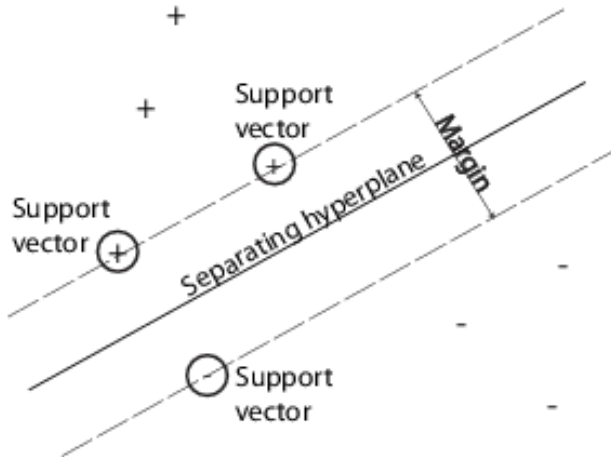
No separable Data

Nonlinear Transformation with Kernel

Separable Data

You can utilize a help vector machine (SVM) when your information has precisely two classes. A SVM groups information by finding the best hyperplane that isolates all information purposes of one class from those of an alternate class. The best hyperplane for an SVM implies the one with the largest margin between the two classes. Edge implies the maximal width of the section parallel to the hyperplane that has no inside information focuses.

The support vectors in the information indicates the nearest isolating hyperplane; these focus on the limit of the piece. The accompanying figure shows these definitions, with + demonstrating information purposes of sort 1 and - demonstrating information purposes of sort - 1.



Using Support Vector Machines

Likewise, with any administered learning model, you first prepare a help vector machine and afterward cross-approve the classifier. Utilize the prepared machine to characterize (anticipate) new information. Likewise, to acquire acceptable prescient exactness, you can utilize different SVM part capacities and tune the parameters of the portion capacities.

Preparing an SVM Classifier

Prepare and alternatively cross approve, a SVM classifier utilizing fitsvm.

```
SVMModel = fitsvm(X,Y,'KernelFunction','rbf',...
    'Standardize',true,'ClassNames',{'negClass','posClass'});
```

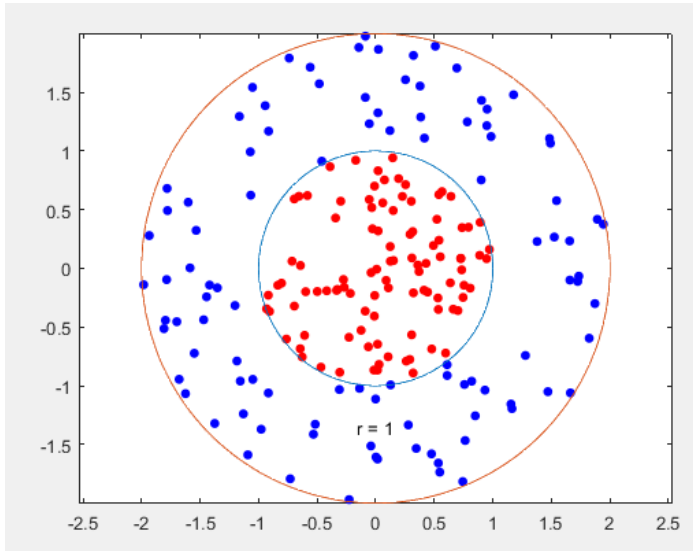
Program

```
%Generate 100 points uniformly distributed in the unit
disk
% To do so, generate a radius r as the square root of
a uniform random variable,
%generate an angle t uniformly in (0, ),
%and put the point at (r cos( t ), r sin( t ))

rng(1) ; % For reproducibility
r = sqrt(rand(100,1)) ; % Radius
t = 2*pi*rand(100,1) ; % Angle
data1 = [r.*cos(t), r.*sin(t)] ; % Points
%Generate 100 points uniformly distributed in the
annulus.
%The radius is again proportional to a square root,
%this time a square root of the uniform distribution
from 1 through 4.
r2 = sqrt(3*rand(100, 1)=1); % points

%Plot the points, and plot circles of radii 1 and 2
for comparison.
figure;
plot (data1(:,1),data1(:,2),'r.','MarkerSize',15)
hold on
plot (data2(:,1),data2(:,2),'r.','MarkerSize',15)
ezpolar(@(x)1) ;ezpolar(@(x)2);
axis equal
hold off
```

Ouput



Program

```
%Put the data in one matrix, and make a vector of
classifications.
data3 = [data1;data2] ;
theclass = ones(200,1) ;
theclass(1;100) = -1;
% Train an SVM classifier with KernelFunction set to
'rbf' and BoxConstraint
% set to Inf. Plot the decision boundary and flag the
support vectors.
c1 = fitsvm(data3,theclass, 'KernelFunction', 'rbf' ,
. . .
'BoxConstraint',Inf,'ClassNames', [-1,1] ) ;

% Predict scores over the grid
d = 0.02;
[x1Grid,x2Grid] =
meshgrid(min(data3(:,1)):d:max(data3(:,1)) , . . .
min(data3(:,2)) :d:max(data3(:,2)) ) ;
xGrid = [x1Grid( : ) ,x2Grid( : ) ] ;|
```

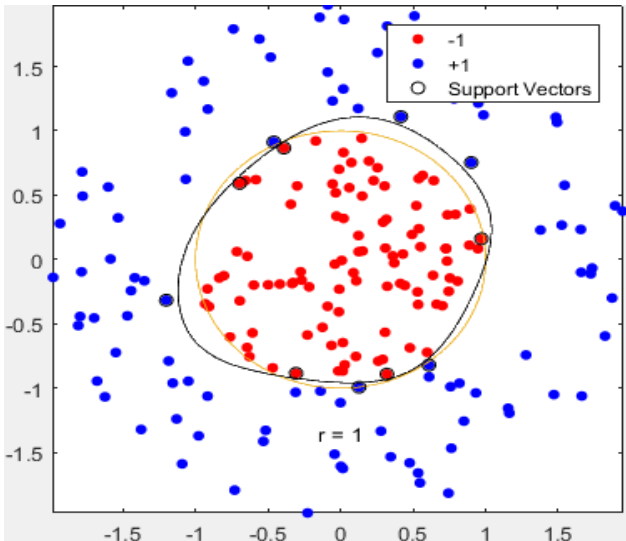
```

[~,scores] = predict (c1,xGrid) ;

% Plot the data and the decision boundary
figure;
h(1:2) = gscatter(data3(:,1),
data3(:,2),theclass,'rb','.') ;
hold on
ezpolar (2(x)1) ;
h(3) = plot(data3(c1.IsSupportVector,1),data3(c1.
IsSupportVector,2),'ko');
contour
(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid) ),[0
0], 'k') ;
legend (h,{'-1','+1','Support Vectors'}) ;
axis equal
hold off

```

Output



Program

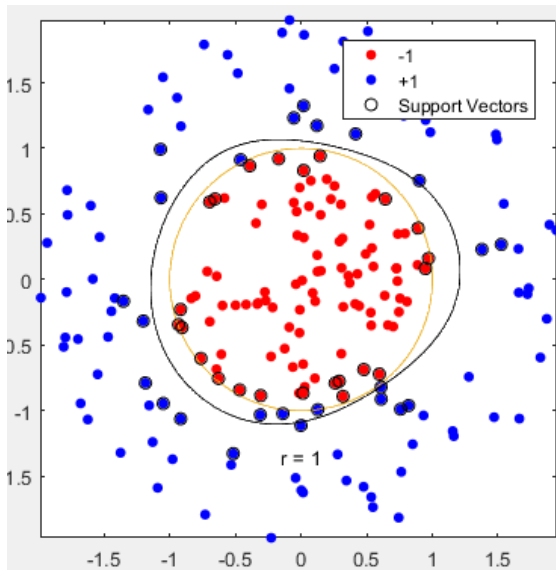
```

c12 = fitcsvm(data3,theclass,'KernelFunction','rbf') ;
[~,scorers2] = predict (c12, xGrid) ;

```

```
figure;
h(1:2) = gscatter (data3(:,1), data3(:,2), theclass,
'rb' , ' . ' ) ;
hold on
ezpolar (@ (x) 1 ) ;
h (3) = plot (data3(c12.IsSupportVector,1),data3(c12.
IsSupportVector,2),'ko') ;
contour (x1Grid,x2Grid,reshape (scores2( : , 2) , size
(x1Grid) ) , [0 0] , 'k') ;
legend (h, {'-1' , '+1' , 'Support Vectors'} ) ;
axis equal
hold off
```

Output



Optimize an SVM Classifier Fit Using Bayesian Optimization

1. Choose a base point m of the proper shading consistently at arbitrary.
2. Generate an autonomous arbitrary point with 2-D ordinary dispersion with mean m and change $I/5$, where I is the 2-by-2 character lattice. In this case, utilize a fluctuation $I/50$ to demonstrate the benefit of advancement all the more unmistakably.

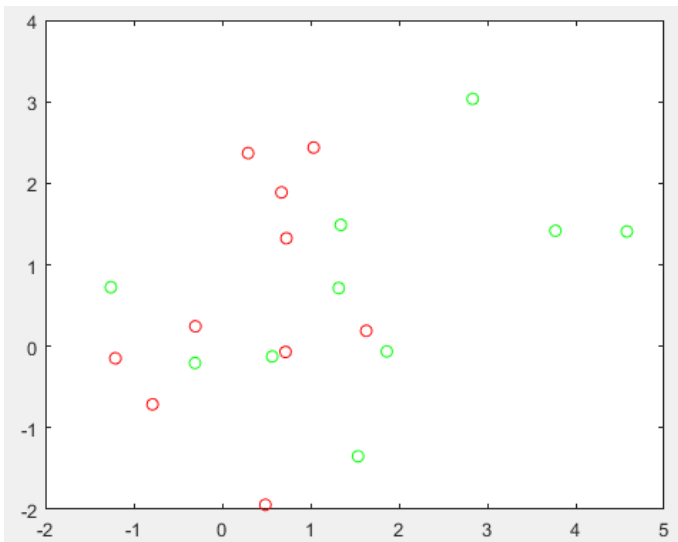
Generate the Points and Classifier

Create base point of 10 in each class.

Program

```
%Optimize an SVM Classifier Fit Using Bayesian
Optimization
rng default % For reproducibility
grnpop = mvrnd ( [1 , 0], eye (2) , 10) ;
redpop = mvrnd ( [0 , 1] , eye (2) , 10) ;
plot (grnpop (: , 1) , grnpop (: , 2) , 'go' )
hold on
plot (redpop (: ,1) , redpop (: , 2) , 'ro' )
hold off
```

Output



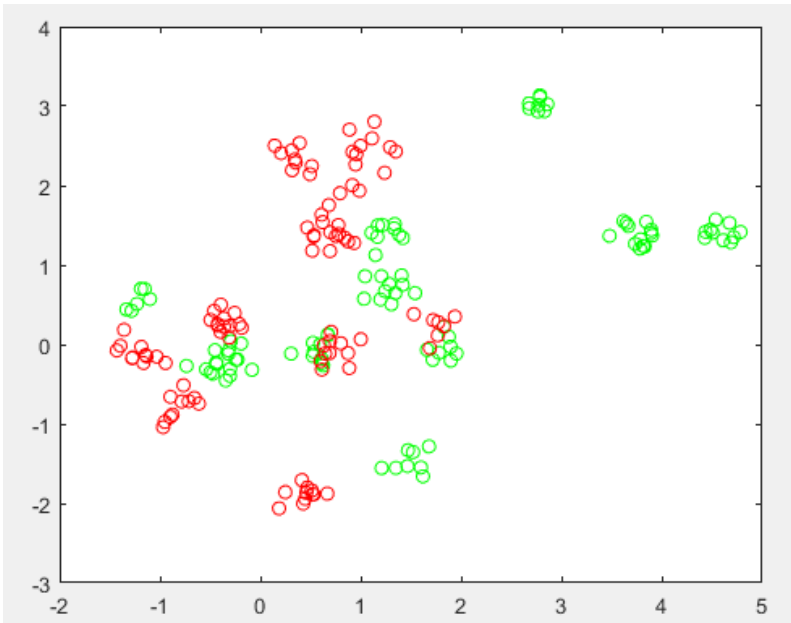
It is difficult to classify data points with respect to the location of red and green points. In each class create 100 points of data variable.

Program

```
redpts = zeros (100,2) ; grnpts = redpts;
for I = 1:100
    grnpts (i, :) = mvrnd (grnpop(randi (10) , :)
,eye(2)*0.02) ;
```

```
        redpts (i, :) = mvrnd (redpop(randi (10) , :)  
,eye(2)*0.02) ;  
end  
figure  
plot (grnpts(:, 1) ,grnpts (:, 2) , 'go' )  
hold on  
plot (redpts(:, 1) ,redpts (:, 2) , 'go' )  
hold off
```

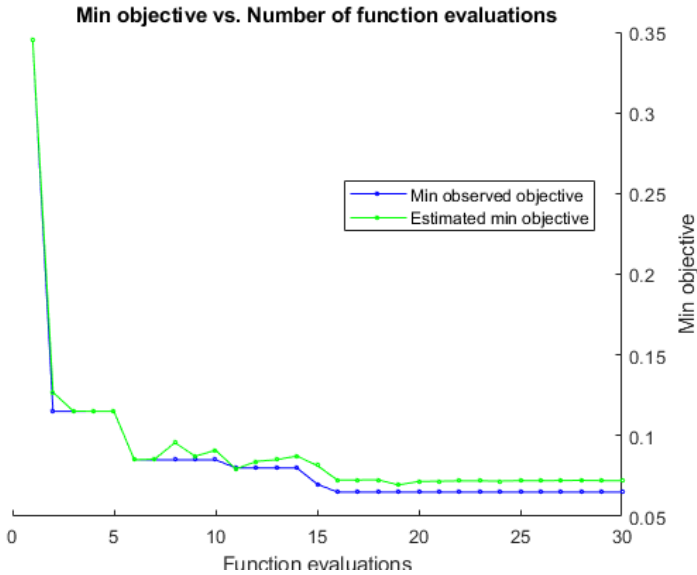
Output



Prepare Data for Classification

Optimize the Fit

Find the loss in the optimized model.



Find the loss in the optimized model.

Program

```
%Prepare Data For Classification,
%Put the data into one matrix,
%and make a vector grp that labels the class of each
point.
cdata = [grnpts;redpts] ;
grp = ones(200,1) ;
% Green label 1, red label -1
grp(101:200) = -1;
%prepare Cross-Validation
c = cvpartition(200,'KFold',10) ;
%Optimize the Fit
opts = struct('Optimizer','bayesopt','Show-
Plots',true,'CVPartition',c, . . .
'AcquisitionFunctionName','expected-improve-
ment-plus') ;
svmmod = fitcsvm(cdata,grp,'KernelFunction','rbf', . . .
.
```

```

    'OptimizeHyperparameters', 'auto', 'HyperparameterOptimizationOptions', opts)
%Find the loss of the optimized model.
lossnew = kfoldLoss(fitcsvm(cdata, grp, 'CVPartition', c, 'KernelFunction', 'rbf', . . .
    'BoxConstraint', svmmod.HyperparameterOptimizationResult.XAtMinObjective.
BoxConstraint, . . .
    'KernelScale', svmmod.HyperparameterOptimizationResults.XAtMinObjective.
    KernelScale) )
%This loss is the same as the loss reported in the Optimization output under
%"Observed objective function value". Visualize the optimized classifier.
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(cdata(:,1)):d:max(cdata(:,1)), . . .
    min(cdata(:,2)):d:max(cdata(:,2))) ;
xGrid = [x1Grid(:), x2Grid(:)] ;
[~, scores] = predict (svmmod, xGrid) ;
figure;
h = nan(3,1) ; % Preallocation
h(1:2) = gscatter(cdata(:,1), cdata(:,2), grp, 'rg', '+*')
;
hold on

h(3) = plot (cdata(svmmod.IsSupportVector,1), . . .
    cdata(svmmod.IsSupportVector,2), 'ko') ;
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k') ;
legend(h, {'-1', '+1', 'Support Vectors'}, 'Location', 'Southeast') ;
axis equal
hold off

```

3.9.4 Regression Algorithms

Linear Regression

Multiple, stepwise, multivariate regression models,
Preparing your Data

For relapse fitting, the information frame fitting capacities work to anticipate. Relapse systems start with X as the input array and reaction information different y vector, or information and reaction information function tbl. Each column information speaks to one perception. Every segment speaks to one indicator (variable).

The data format in the table or dataset array tbl, demonstrate a reaction variable as 'ResponseVar' name-esteem combined:

```
mdl = fitlm(tbl, 'ResponseVar', BloodPressure');
```

Dataset Array Data for Input and Response

Creation of an array for a dataset in the spreadsheet of excel format.

```
ds = dataset('XLSFile','hospital.xls', ...
            'ReadObsNames',true);
```

To create a dataset array from workspace variables

```
load carsmall
ds = dataset(MPG,Weight);
ds.Year = categorical(Model_Year);
```

Input data is included in a numeric matrix and the response is denoted in a Numeric Vector.

For example, for creating workspace variables we first create an array in a numeric manner.

```
load carsmall
X = [Weight Horsepower Cylinders Model_Year];
y = MPG;
```

To create numeric arrays from an Excel spreadsheet

```
[X, Xnames] = xlsread('hospital.xls');
y = X(:,4); % response y is systolic pressure
X(:,4) = []; % remove y from the X matrix
```

Selection of Fitting Method

In order to fit data into the model three methods are adopted:

- Least-Squares Fit
- Robust Fit
- Stepwise Fit

Data Fitting

The most extensively seen discretionary clashes for fitting:

For a powerful fall away from the faith in fitlm, set the 'RobustOpts' name-respect combine.

Exhibit a suitable model in the upper bound function stepwiselm, for example, 'Upper' to 'linear'.

It shows the factor straight out utilizing 'CategoricalVars' name-respect facilitate. Outfit your vector portion numbers as [1 6] to affirm predictors 1 and 6 unmitigated. On the other hand, give a canny vector an obscure length from the information partitions, with a 1 demonstrating that the variable is rigid. In the event with seven markers and predictions 1 and 6 straight out, specify logical ([1,0,0,0,0,1,0]).

The dataset in tabular form provides a choose reaction variable for utilizing the 'ResponseVar' name-respect join.

For instance

```
mdl = fitlm(X,y,'linear', ...
           'RobustOpts','on','CategoricalVars',3);
mdl2 = stepwiselm(tbl,'constant', ...
                 'ResponseVar','MPG','Upper','quadratic');
```

Inspect the Quality and Adjust the Fitted Model.

For fitting a model, look at the outcome and changes.

- Model Display
- ANOVA
- Diagnostic Plots
- Residuals — Model Quality for Training Data
- Plots to Understand Predictor Effects
- Plots to Understand Terms Effects
- Change Models

Demonstrate Display

- A straight drop into wrongdoing shows a few diagnostics when you enter its name or enter `disp mdl`. This show gives a dash of the basic information to check whether the fitted model watches out for the data pleasingly.
- For portrayal purpose, fit a provoke model to the data worked with two out of five pointers not present and with no piece term

Program

```
>> X = randn(100,50) ;
y = X*[1;0;3;0;-1] + randn(100,1) ;
mdl = fit1m(X,y)
```

```
mdl =
```

Linear regression model:

$$y \sim 1 + x1 + x2 + x3 + x4 + x5$$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	-0.11782	0.10124	-1.1637	0.24748
x1	1.053	0.10622	9.9135	2.8065e-16
x2	0.035404	0.10119	0.34988	0.72721
x3	2.9243	0.10854	26.942	5.3755e-46
x4	-0.028114	0.11533	-0.24376	0.80795
x5	-1.0608	0.10971	-9.6686	9.3096e-16

Number of observation: 100, Error degree of freedom: 94

Root Mean Squared Error: 0.99

R-squared: 0.904, Adjusted R-Squared 0.899

F-statistic vs. constant model: 177,p-value = 2.98e-46

ANOVA

Inspect the nature of the fitted model, counsel as ANOVA table. The instance, ANOVA provides direct model of 5 indicators

Program

```
>> x = randn(100,5) ;
y = X*[1;0;3;0;-1] + randn(100,1) ;
mdl = fitlm(X,y) ;
tbl = anova(mdl)
```

tbl =

	SumSq	DF	MeanSq	Fp	Value
x1	131.82	1	131.82	149.5	3.8772e-21
x2	0.12773	1	0.12773	0.14487	0.70435
x3	949.29	1	949.29	1076.7	2.8371e-53
x4	0.033959	1	0.033959	0.038515	0.84484
x5	107.84	1	107.84	122.31	1.0545e-18
Error	82.881	94	0.88171		

Diagnostic Plots

Symptomatic plots enable you to recognize exceptions and various issues in your model or fit. For instance, stack the carsmall data and make a model of MPG as a capacity of Cylinders (categorical) and Weight

Make a leverage plot of the data and model.

```
>> plotDiagnostics(mdl)
```

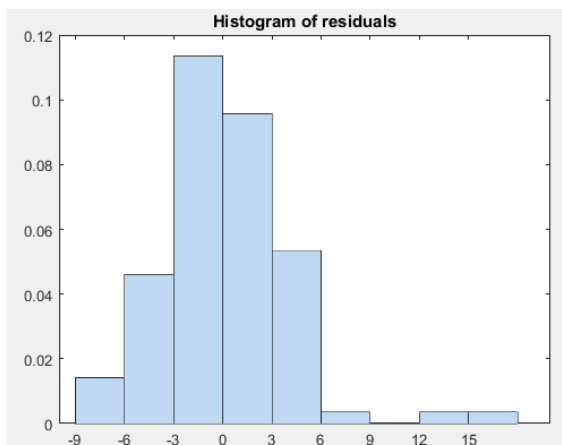



Load the carsmall data and make the model as an MPG function of Cylinders (categorical) and Weight

Program

```
>>plotDiagonstics (md1)
>> load carsmall
tb1 = table(weight,MPG,Cylinders) ;
tb1.Cylinders = categorical (tb1.Cylinders) ;
md1 = fitlm(tb1,'MPG ~ Cylinders*Weight + Weight^2') ;
>>
>>plotResiduals(md1)
```

Output



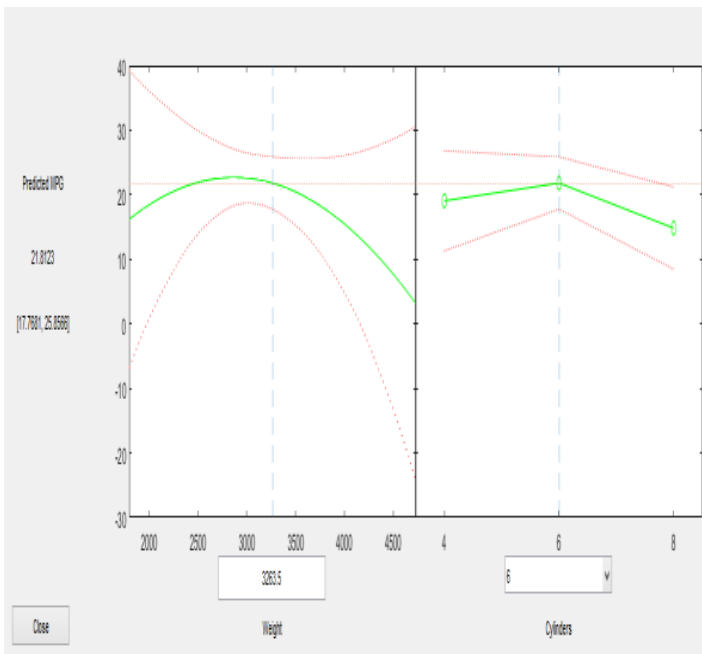
Plots to Understand Predictor Effects

Illustration demonstrates a comprehensive impact of every indicator relapse display utilizing an assortment of accessible plots.

Program

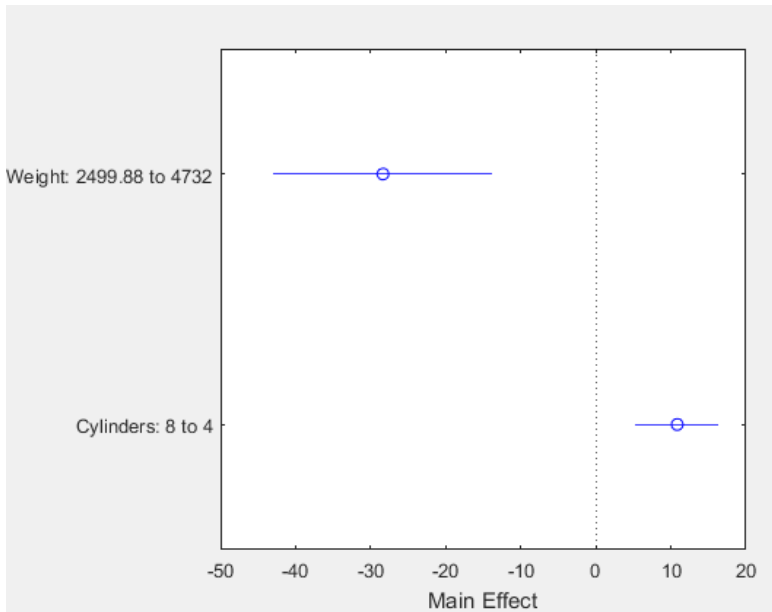
```
>> load carsmall
tb1 = table(Weight,MPG,Cylinders) ;
tb1.Cylinders = categorical (tb1.Cylinders) ;
mdl = fitlm(tb1,'MPG ~ Cylinders*Weight + Weight^2') ;
>>plotSlice(mdl)
```

Output



Utilize an impact plot to demonstrate another perspective of the impact of indicators on the reaction

```
>> plotEffects (mdl)
```



New Data Prediction Simulation

The following 3 approaches utilize straight foresee mimic reaction of new information:

- predict
- feval
- random

Anticipate

This case exhibits to foresee and get the desirable assurance break prediction method. In the model, created data for carbig are loaded into the default model MPG response for the Acceleration, Displacement, Horsepower and Weight predictors.

```
>> load carbig
X = [Acceleration,Displacement,Horsepower,Weight];
mdl = fitlm(X,MPG);
```

With the fewer NaN values, the functions ignore NaN.

```
>> Xnew = [nanmin(X);nanmean(X);nanmax(X)]; % new data
```

To evaluate the prediction response of the model, confidence values are predicted.

```
>> [NewMPG, NewMPGCI] = predict mdl, Xnew)
```

```
NewMPG =
```

```
34.1345
23.4078
4.7751
```

```
NewMPGCI =
```

```
31.6115    36.6575
22.9859    23.8298
0.6134     8.9367
```

Feval

When you develop a model from a table or dataset array, feval is once in a while more conclusive for envisioning mean reactions than predict. However, feval does not give confirmation bounds. The following case demonstrates the usage of mean reactions utilizing the feval method. Load the carbig data and make a default arrange model of the response MPG to the Acceleration, Displacement, Horsepower and Weight predictors.

```
>> load carbig
tbl = table(Acceleration, Displacement, Horsepower, Weight, MPG);
mdl = fitlm(tbl, 'linear', 'ResponseVar', 'MPG');
```

The Xnew array provides an accurate count of the prediction column hence feval is also used for predictions.

Find the predicted model responses.

```
>> NewMPG = feval(mdl,Xnew)
```

```
NewMPG =

    34.1345
    23.4078
     4.7751
```

Nonlinear Regression

Nonlinear Regression Workflow

Step 1. Preparing the data

Load the reaction data.

Load reaction

Examine the data in the workspace. Reactants are a matrix with 13 rows and 3 columns. Each row corresponds to one observation and each column corresponds to one variable.

The variable names are in xn

```
xn =

    3x10 char array
    'Hydrogen '
    'n-Pentane '
    'Isopentane'
```

Similarly, rate is a vector of 13 responses, with the variable name in yn

rate is a vector of 13 responses, with the variable name in yn

```
yn
```

```
yn =
```

```
    'Reaction Rate'
```

The `hougen.m` file contains a nonlinear model of the reaction rate as a function of the three predictor variables. For a 5-D vector b and 3-D vector x ,

$$hougen(b,x) = \frac{b(1)x(2) - x(3) / b(5)}{1 + b(2)x(1) + b(3)x(2) + b(4)x(3)}$$

As a start point for the solution, take b as a vector of ones.

`beta0 = ones(5,1);`

Step 2. Fit a nonlinear model to the data.

```
>> load reaction
>>
>> beta0 = ones(5,1);
>> mdl = fitnlm(reactants,...
    rate,@hougen,beta0)

mdl =

Nonlinear regression model:
    y ~ hougen(b,X)

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
b1	1.2526	0.86702	1.4447	0.18654
b2	0.062776	0.043562	1.4411	0.18753
b3	0.040048	0.030885	1.2967	0.23089
b4	0.11242	0.075158	1.4957	0.17309
b5	1.1914	0.83671	1.4239	0.1923

```

Number of observations: 13, Error degrees of freedom: 8
Root Mean Squared Error: 0.193
R-Squared: 0.999, Adjusted R-Squared 0.998
F-statistic vs. zero model: 3.91e+03, p-value = 2.54e-13

```

Step 3 - Examine the quality of the model.

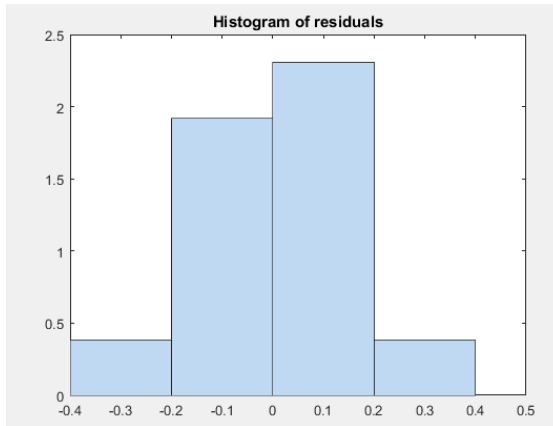
The root mean squared error is fairly low compared to the range of observed values.

```
>> [mdl.RMSE min(rate) max(rate)]
```

```
ans =
```

```
0.1933    0.0200    14.3900
```

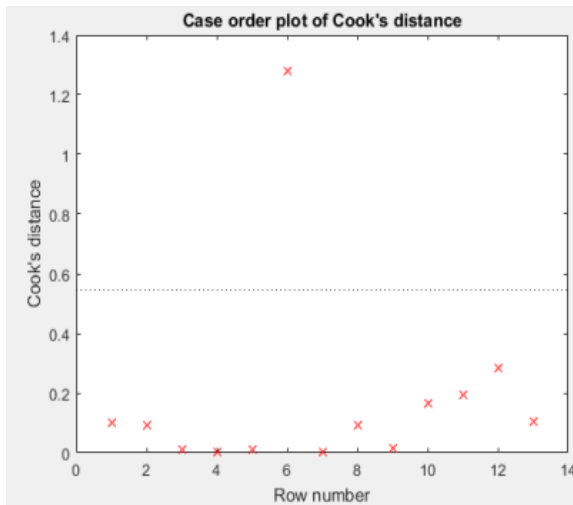
```
>> plotResiduals(mdl)
```



The model seems adequate for the data.

Examine a diagnostic plot to look for outliers.

```
>> plotDiagnostics (mdl, 'cookd')
```



Observation 6 seems out of line.

Step 4. Remove the outlier

Remove the outlier from the fit using the Exclude name-value pair.

```
>> mdl1 = fitnlm(reactants,...
    rate,@hougen,ones(5,1),'Exclude',6)
```

```
mdl1 =
```

```
Nonlinear regression model:
    y ~ hougen(b,X)
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
b1	0.619	0.4552	1.3598	0.21605
b2	0.030377	0.023061	1.3172	0.22924
b3	0.018927	0.01574	1.2024	0.26828
b4	0.053411	0.041084	1.3	0.23476
b5	2.4125	1.7903	1.3475	0.2198

```
Number of observations: 12, Error degrees of freedom: 7
```

```
Root Mean Squared Error: 0.198
```

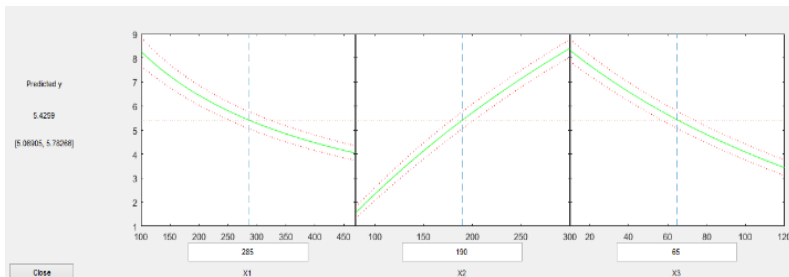
```
R-Squared: 0.999, Adjusted R-Squared 0.998
```

```
F-statistic vs. zero model: 2.67e+03, p-value = 2.54e-11
```

Step 5. Examine slice plots of both models.

To see the effect of each predictor on the response, make a slice plot using plotSlice(mdl)

```
>> plotSlice(mdl)
```



Step 6. Predict for new data.

Create new data and try predicting the response using both models.

```
>> Xnew = [200,200,200;100,200,100;500,50,5];
[ypred yci] = predict(mdl,Xnew)
```

```
ypred =
```

```
1.8762
6.2793
1.6718
```

```
ygi =
```

```
1.6283    2.1242
5.9789    6.5797
1.5589    1.7846
```

```
>> [ypred1 ygi1] = predict(mdl1,Xnew)
```

```
ypred1 =
```

```
1.8984
6.2555
1.6594
```

```
ygi1 =
```

```
1.6260    2.1708
5.9323    6.5787
1.5345    1.7843
```

Even though the model coefficients are dissimilar, the predictions are nearly identical.

Gaussian Process Regression Model

Gaussian process regression (GPR) models are nonparametric kernel-based probabilistic models. You can train a GPR model using the `fitrgp` function.

```
tbl = readtable('abalone.data', 'Filetype', 'text', ...
    'ReadVariableNames', false);
tbl.Properties.VariableNames = {'Sex', 'Length', 'Diameter', 'Height', ...
    'WWeight', 'SWeight', 'VWeight', 'ShWeight', 'NoShellRings'};
tbl(1:7, :)
ans =
```

Sex	Length	Diameter	Height	WWeight	SWeight	VWeight	ShWeight	NoShellRings
M	0.455	0.365	0.095	0.514	0.2245	0.101	0.15	15
M	0.35	0.265	0.09	0.2255	0.0995	0.0485	0.07	7
F	0.53	0.42	0.135	0.677	0.2565	0.1415	0.21	9
M	0.44	0.365	0.125	0.516	0.2155	0.114	0.155	10
T	0.33	0.255	0.08	0.205	0.0895	0.0395	0.055	7
T	0.425	0.3	0.095	0.3515	0.141	0.0775	0.12	8
F	0.53	0.415	0.15	0.7775	0.237	0.1415	0.33	20

Fit a GPR model using the subset of regressors method for parameter estimation and the fully independent conditional method for prediction. Standardize the predictors.

```
gprMdl = fitrgp(tbl, 'NoShellRings', 'KernelFunction', 'ardsquaredexponential', ...
    'FitMethod', 'sr', 'PredictMethod', 'fic', 'Standardize', 1)
```

Predict the responses using the trained model

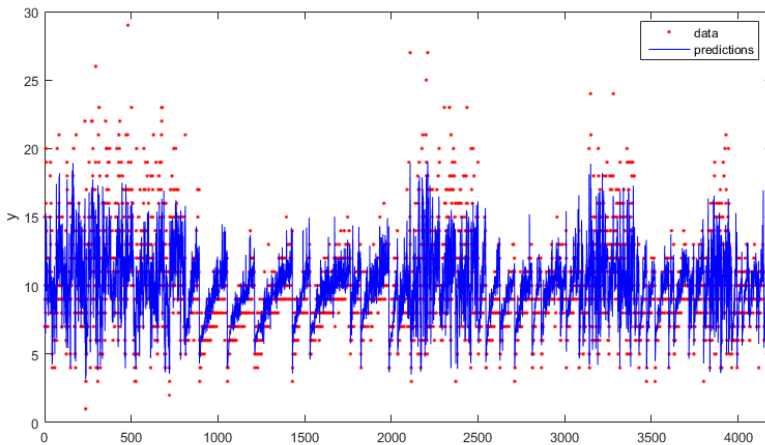
```

PredictorLocation: [10x1 double]
PredictorScale: [10x1 double]
Alpha: [1000x1 double]
ActiveSetVectors: [1000x10 double]
PredictMethod: 'FIC'
ActiveSetSize: 1000
FitMethod: 'SR'
ActiveSetMethod: 'Random'
IsActiveSetVector: [4177x1 logical]
LogLikelihood: -9.0013e+03
ActiveSetHistory: [1x1 struct]
BCDInformation: []
ypred = resubPredict(gprMdl);
Plot the true response and the predicted responses.
figure();
plot(tbl.NoShellRings,'r.');
```

```

hold on
plot(ypred,'b');
xlabel('x');
ylabel('y');
legend({'data','predictions'},'Location','Best');
axis([0 4300 0 30]);
hold off;
```

Output



```
L = resubLoss(gprMdl)
L =
    4.0064
```

Train GPR Model and Plot Predictions

Sample data generation

```
>> rng(0, 'twister'); % For reproducibility
n = 1000;
x = linspace(-10,10,n)';
y = 1 + x*5e-2 + sin(x)./x + 0.2*randn(n,1);
```

Fit a GPR model using a linear basis function and the exact fitting method to estimate the parameters. Also use the exact prediction method.

```
>> gprMdl = fitrgp(x,y,'Basis','linear',...
    'FitMethod','exact','PredictMethod','exact');
```

Predict the response corresponding to the rows of x (re-substitution predictions) using the trained model.

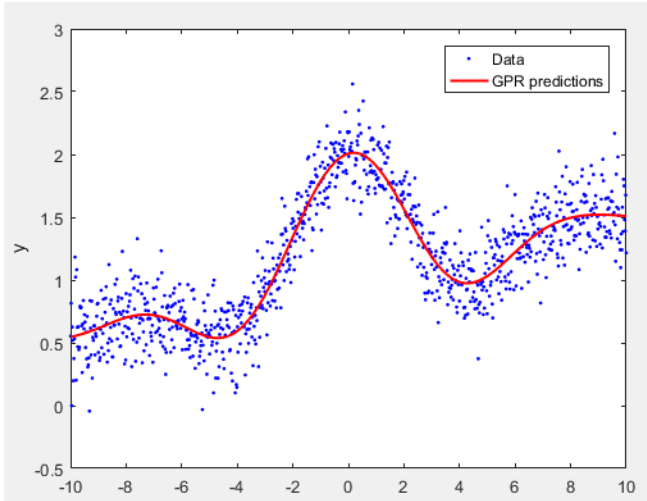
```
>>ypred = resubPredict(gprMdl);
```

Plot the true response with the predicted values.

Program

```
>>plot (x,y, 'b.') ;
hold on;
plot (x,ypred, 'r', 'LineWidth',1.5) ;
xlabel ('x') ;
ylabel ('y') ;
legend ('Data', 'GPR predictions') ;
hold off
```

Output



Impact of specifying Initial Kernel Parameter Values

Load the sample data.

Fit a GPR model using the squared exponential kernel function with default kernel parameters.

```
>> gprMdl1 = fitrgp(x,y,'KernelFunction','squaredexponential');
```

Now, fit a second model, where you specify the initial values for the kernel parameters

```
>> sigma0 = 0.2;
kparams0 = [3.5, 6.2];
gprMdl2 = fitrgp(x,y,'KernelFunction','squaredexponential',...
    'KernelParameters',kparams0,'Sigma',sigma0);
```

Compute the re-substitution predictions from both models.

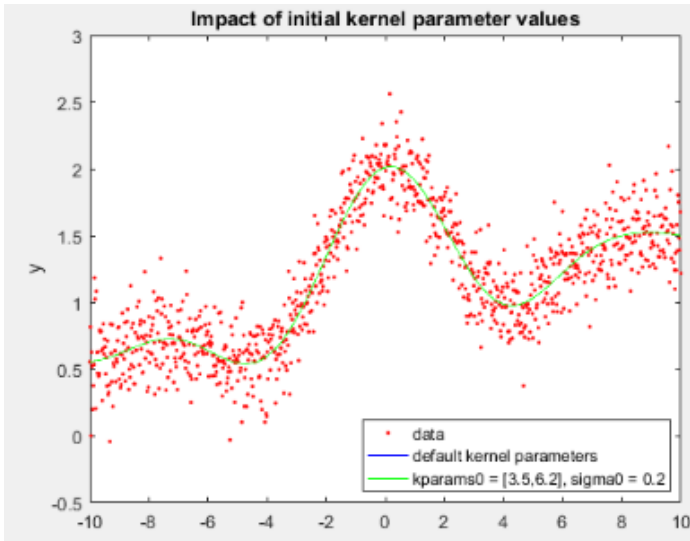
```
>> ypred1 = resubPredict(gprMdl1);
ypred2 = resubPredict(gprMdl2);
```

Plot the response prediction from both models and the responses in the training data.

Program

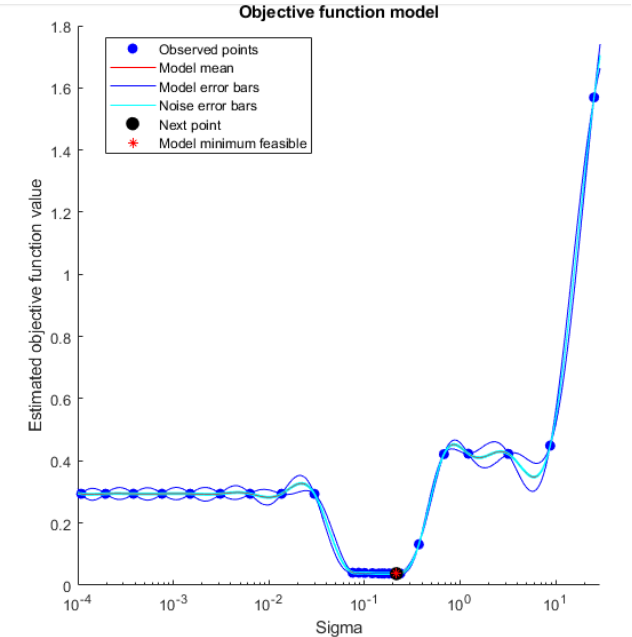
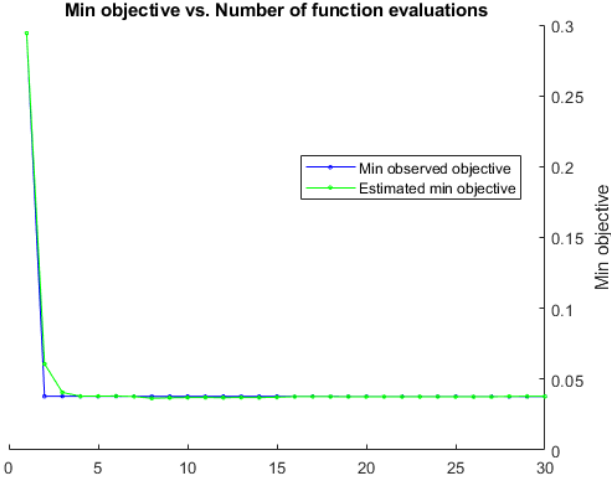
```
>>figure ( ) ;
plot (x,y, 'r.') ;
hold on
plot(x,ypred1, 'b') ;
plot(x,ypred2, 'g') ;
xlabel ( 'x' ) ;
ylabel ( 'y' ) ;
legend ( {'data' , 'default kernel parameters' , . . .
'kparams0 = [3.5,6.2], sigma0 = 0.2'} , . . .
'Location' , 'Best' ) ;
title ('Impact of initial kernel parameter values') ;
hold off
```

Output



The peripheral log probability that `fitrgp` maximizes to gauge GPR parameters has numerous nearby arrangements; the arrangement that it focalizes to relies upon the underlying point. Every neighborhood arrangement relates to a specific elucidation of the information. In this case, the arrangement with the default initial part parameters relates to a

low recurrence motion with high clamor though the second arrangement with custom beginning piece parameters compares to a high recurrence motion with low commotion.



Optimize GPR Regression

This illustration demonstrates the enhancement in hyperparameters consequently using `fitrgp`. The case utilizes the `gprdata2` data that boats with your product.

```
load(fullfile(matlabroot, 'examples', 'stats', 'gprdata2.mat'))
```

The data has one predictor variable and a continuous response. This is the simulated data.

```
gprMd11 = fitrgp(x,y,'KernelFunction','squareexponential');
```

Fit a GPR display utilizing the squared exponential piece work with default part parameters.

Discover hyper-parameters that limit five-crease cross-approval misfortune by utilizing programmed hyper-parameter enhancement.

For reproducibility, set the arbitrary seed and utilize the ‘expected-change plus’ acquisition work.

```
rng default
gprMd12 = fitrgp(x,y,'KernelFunction','squareexponential',...
'OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',...
struct('AcquisitionFunctionName','expected-improvement-plus'));
```


Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	Sigma
1	Best	0.29417	2.1518	0.29417	0.29417	0.0015045
2	Best	0.037898	1.3153	0.037898	0.060792	0.14147
3	Accept	1.5693	0.76112	0.037898	0.040633	25.279
4	Accept	0.29417	1.7549	0.037898	0.037984	0.0001091
5	Accept	0.29393	1.8182	0.037898	0.038029	0.029932
6	Accept	0.13152	1.2091	0.037898	0.038127	0.37127
7	Best	0.037785	1.4368	0.037785	0.037728	0.18116
8	Accept	0.03783	1.3877	0.037785	0.036524	0.16251
9	Accept	0.037833	1.5439	0.037785	0.036854	0.16159
10	Accept	0.037835	1.8267	0.037785	0.037052	0.16072
11	Accept	0.29417	1.8249	0.037785	0.03705	0.00038214
12	Accept	0.42256	1.0248	0.037785	0.03696	3.2067
13	Accept	0.03786	1.3548	0.037785	0.037087	0.15245
14	Accept	0.29417	1.8118	0.037785	0.037043	0.0063584
15	Accept	0.42302	1.027	0.037785	0.03725	1.2221
16	Accept	0.039486	1.2477	0.037785	0.037672	0.10069
17	Accept	0.038591	1.3022	0.037785	0.037687	0.12077
18	Accept	0.038513	1.334	0.037785	0.037696	0.1227
19	Best	0.037757	1.3904	0.037757	0.037572	0.19621
20	Accept	0.037787	1.452	0.037757	0.037601	0.18068
Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	Sigma
21	Accept	0.44917	0.90673	0.037757	0.03766	8.7818
22	Accept	0.040201	1.2108	0.037757	0.037601	0.075414
23	Accept	0.040142	1.1481	0.037757	0.037607	0.087198
24	Accept	0.29417	1.8435	0.037757	0.03758	0.0031018

```
Optimization completed.  
MaxObjectiveEvaluations of 30 reached.  
Total function evaluations: 30  
Total elapsed time: 59.3447 seconds.  
Total objective function evaluation time: 43.2254
```

```
Best observed feasible point:
```

```
  Sigma
```

```
  _____
```

```
  0.2367
```

```
Observed objective function value = 0.037704  
Estimated objective function value = 0.037881  
Function evaluation time = 1.3398
```

```
Best estimated feasible point (according to models):
```

```
  Sigma
```

```
  _____
```

```
  0.16159
```

```
Estimated objective function value = 0.037881  
Estimated function evaluation time = 1.3583
```

Support Vector Machine Regression

Support vector machines for relapse models

For more prominent exactness on low-through medium-dimensional informationa indexes, prepare a help vector machine (SVM) demonstration using fitrsvm.

For a decreased calculation time on high-dimensional informational indexes that fit in the MATLAB Workspace, productively prepare a direct relapse show, for example, a straight SVM display, using fitrlinear.

Prepare Linear SVM Regression Model.

This illustration demonstrates the preparation of an SVM relapse display utilizing test information put away in lattices. Load the carsmall data set.

```
>> load carsmall
rng default % for reproducibility
```

Specify Horsepower and Weight as the predictor variables (X) and MPG as the response variable (Y).

```
>> X = [Horsepower, Weight];
Y = MPG;
```

The Command Window shows that Mdl is a trained RegressionSVM model and a list of its properties.

```
>> Mdl = fitrsvm(X,Y)
```

```
Mdl =
```

RegressionSVM

```
    ResponseName: 'Y'
  CategoricalPredictors: []
    ResponseTransform: 'none'
                Alpha: [75x1 double]
                Bias: 43.2943
  KernelParameters: [1x1 struct]
  NumObservations: 93
    BoxConstraints: [93x1 double]
  ConvergenceInfo: [1x1 struct]
  IsSupportVector: [93x1 logical]
                Solver: 'SMO'
```

Properties, Methods

Check the model for convergence.

```
>> Mdl.ConvergenceInfo.Converged
and =
    0
```

```

>> MdlStd = fitrsvm(X,Y,'Standardize',true)

MdlStd =

RegressionSVM
    ResponseName: 'Y'
  CategoricalPredictors: []
    ResponseTransform: 'none'
                Alpha: [77x1 double]
                Bias: 22.9131
  KernelParameters: [1x1 struct]
                Mu: [109.3441 2.9625e+03]
                Sigma: [45.3545 805.9668]
  NumObservations: 93
    BoxConstraints: [93x1 double]
  ConvergenceInfo: [1x1 struct]
  IsSupportVector: [93x1 logical]
                Solver: 'SMO'

Properties, Methods

>> MdlStd.ConvergenceInfo.Converged

ans =

    1

>> lStd = resubLoss(MdlStd)

lStd =

    17.0256

```

Train Support Vector Machine Regression Model

Train a support vector machine regression model using the abalone data from the UCI Machine Learning Repository.

Download the data and save it in your current folder with the name 'abalone.csv'.

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data';
websave('abalone.csv',url);
```

Read the data into a table and specify the variable names.

```
varnames = {'Sex'; 'Length'; 'Diameter'; 'Height'; 'Whole_weight';...
            'Shucked_weight'; 'Viscera_weight'; 'Shell_weight'; 'Rings'};
Tbl = readtable('abalone.csv','Filetype','text','ReadVariableNames',false);
Tbl.Properties.VariableNames = varnames;
```

The example information contains 4177 perceptions. All the indicator factors are persistent aside from for Sex, which is a clear-cut variable with conceivable values 'M' (for males), 'F' (for females) and 'I' (for babies). The objective is to anticipate the quantity of rings (put away in Rings) on the abalone and decide its age utilizing physical estimations.

Prepare an SVM relapse demonstrate, utilizing a Gaussian portion work with a programmed piece scale. Institutionalize the information.

```
rng default % For reproducibility
Mdl = fitrsvm(Tbl,'Rings','KernelFunction','gaussian','KernelScale','auto',...
             'Standardize',true)
conv = Mdl.ConvergenceInfo.Converged
iter = Mdl.NumIterations
```

Mdl =

```
RegressionSVM
    PredictorNames: {1x8 cell}
    ResponseName: '...'
    CategoricalPredictors: 1
    ResponseTransform: '...'
        Alpha: [3635x1 double]
        Bias: 10.8144
    KernelParameters: [1x1 struct]
        Mu: [1x10 double]
        Sigma: [1x10 double]
    NumObservations: 4177
    BoxConstraints: [4177x1 double]
    ConvergenceInfo: [1x1 struct]
    IsSupportVector: [4177x1 logical]
    Solver: '...'
```

[Properties, Methods](#)

```

Mdl =

  RegressionSVM
      PredictorNames: {1x8 cell}
      ResponseName: '...'
      CategoricalPredictors: 1
      ResponseTransform: '...'
      Alpha: [3635x1 double]
      Bias: 10.8144
      KernelParameters: [1x1 struct]
          Mu: [1x10 double]
          Sigma: [1x10 double]
      NumObservations: 4177
      BoxConstraints: [4177x1 double]
      ConvergenceInfo: [1x1 struct]
      | IsSupportVector: [4177x1 logical]
      Solver: '...'

  Properties, Methods

conv =

    1

iter =

    2759

```

Generalized Linear Models

What Are Generalized Linear Models?

Straight relapse models depict a direct connection between a reaction and at least one prescient term. Ordinarily, be that as it may, it provides a relationship in non-linear manner. An exceptional class of direct or straight model is the non-linear model.

- Review that straight models have these qualities: At each arrangement of qualities for the indicators, the reaction has a typical dispersion with mean μ . A coefficient vector b defines a linear combination Xb of the predictors X .
- The model is $\mu = Xb$.

In summed up direct models, these attributes are summed up as after the following:

- At each arrangement of qualities for the indicators, the reaction has a conveyance that can be normal, binomial, Poisson, gamma, or inverse Gaussian, with parameters including a mean μ .
- A coefficient vector b defines a direct combination Xb of the predictors X .
- A link function f defines the model as $f(\mu) = Xb$.

Preparing Data

To start fitting a relapse, put your information into a shape which fitting capacities anticipate. All relapse procedures start with input information in an array X and reaction information in a different vector y , or info information in a table or dataset array tbl and reaction information as a segment in tbl . Each column of the information speaks to one perception. Every segment speaks to one indicator.

Through a dataset or table the function tbl provides a variable reaction in combination with 'ResponseVar':

```
mdl = fitlm(tbl, 'ResponseVar', 'BloodPressure');
% or
mdl = fitglm(tbl, 'ResponseVar', 'BloodPressure');
```

For a table or dataset array tbl , fitting capacities expect that this information composed is straight out:

- Logical
- Character cluster

In the event that you need to show that a numeric indicator is all out, utilize the ‘Categorical’ name-esteem combine.

Speak to the missing numeric information as NaN. To speak to missing information for other information writes, see the section on Missing Group Values.

For a ‘binomial’ model with an information matrix X, the response y can be:

Binary segment vector — Each passage speaks to progress (1) or disappointment (0).

Two-segment framework of whole numbers — The principal segment is the quantity of accomplishments in every perception, the second segment is the quantity of trials in that perception.

For a ‘binomial’ model with table or dataset tbl:

Use the ResponseVar name-esteem match to indicate the section of tbl that gives the quantity of accomplishments in every perception.

Use the BinomialSize name-esteem combine to determine the segment of tbl that gives the quantity of trials in every perception.

Dataset Array for Input and Response Data

For example, to create a dataset array from an Excel® spreadsheet

```
ds = dataset('XLSFile', 'hospital.xls', ...
            'ReadObsNames', true);
```

To create a dataset array from workspace variables

Table for Input and Response Data

```
load carsmall
ds = dataset(MPG, Weight);
ds.Year = ordinal(Model_Year);
```

For Input Data in Numeric Matrix and response in Numeric Vector

```
load carsmall
tbl = table(MPG, Weight);
tbl.Year = ordinal(Model_Year);
```

If you want to create an array with numeric workspace, its variables are evaluated as follows:

```
load carsmall
```



```
X = [Weight Horsepower Cylinders Model_Year];
y = MPG;
```

For creating an array in a numeric format with Spreadsheet in Excel:

```
[X, Xnames] = xlsread('hospital.xls');
y = X(:,4); % response y is systolic pressure
X(:,4) = []; % remove y from the x matrix
```

Regression Tree Class

Description

Choices tree with twofold parts for relapse. A question Regression Tree can foresee reactions of information using the prediction model. For the process of re-substitution, forecasting for preparing information is utilized.

Construct Regression Trees

Sample Data creation

```
load carsmall;
```

Through sample data, a regression tree is constructed using the following command:

```
tree = fitrtree([Weight, Cylinders],MPG,...
               'categoricalpredictors',2, 'MinParentSize',20,...
               'PredictorNames',{'W','C'})
```

RegressionTree

```
PredictorNames: {1x2 cell}
ResponseName: 'Y'
CategoricalPredictors: 2
ResponseTransform: '...'
NumObservations: 94
```

Properties, Methods

Predict the mileage of 4,000-pound cars with 4, 6 and 8 cylinders

```
mileage4k =
19.2778
```

19.2778

14.3889

Nonlinear Regression

In the case of the Nonlinear regression model, it provides a sequential relation between variables of response and continuous for prediction of its values. This uses the following value for prediction

$$y = f(X, \beta) + \varepsilon,$$

Where,

y is represented as *n*-by-1 vector reaction variable perceptions.

f denotes *X* and β capacity for assessment of *X* and β vector with *y* as the comparing line.

For perception and indication consider a matrix $n \times p$ with the value of *X*.

The vector of β is denoted by *p*-by-1 for vector evaluation.

Autonomous vectors are denoted by μ is an *n*-by-1, indistinguishably appropriated irregular aggravations.

Non-parametric models don't endeavor to describe the connection amongst indicators and the reaction with demonstrated parameters. Depictions are frequently in a graphical format for the Decision tree situation.

The function `fitlm` is used to estimate a parameter β to evaluate the mean square limit with respect to *Y* for model $f(X, \beta)$. It needs a beginning value as zero for the beta iteration changed to β vector with mean squared insignificance measures.

Data Preparation

We start fitting the relapse set information frame for fitting capacities anticipation. In all relapse methods we the start input information to array *X* and reaction information to different vector *y* or information table or dataset array `tbl` and reaction information section in `tbl`. Every column information speaks to one perception. Every section speaks to one indicator (variable).

In the dataset array `tbl` or array presents a reaction variable 'ResponseVar' through esteem name.

```
mdl = fitlm(tbl, 'ResponseVar', 'BloodPressure');
```

Input and Response Data Set Array

```
>> ds = dataset('XLSFile', 'hospital.xls', .
    'ReadObsNames', true);
```

Creation of a dataset array from workspace variables:

```
>> load carsmall
ds = dataset(Weight, Model_Year, MPG);
```

Input and Response Data Table

To create a table from an Excel spreadsheet

```
>> tbl = readtable('hospital.xls', ...
    'ReadRowNames', true);
```

Input Data Numeric Matrix and Response of Numeric vector

Consider the example, numeric array creation from variable of workspace:

Nonlinear Model Representation

Nonlinear models are represented in several ways and we can use several appropriate methods among them.

A Nonlinear model requires fitnlm input as its modelfun input.

```
>> load carsmall
X = [Weight Horsepower Cylinders Model_Year];
y = MPG;
```

Anonymous Function or Function File handling

The limit handle `@modelfun(b,x)` accepts a vector `b` and system, table, or dataset array `x`. The limit handle ought to reestablish a vector `f` with a comparative number of segments as `x`. For example, the limit file `hougen.m` computes as following:

$$hougen(b, x) = \frac{b(1)x(2) - x(3) / b(5)}{1 + b(2)x(1) + b(3)x(2) + b(4)x(3)}$$

Text Representation of Formula

For data in a matrix `X` and response in a vector `y`:

- Represent the formula using `'x1'` as the first predictor (column) in `x`, `'x2'` as the second predictor, etc.

- Represent the vector of parameters to optimize as 'b1', 'b2', etc.
- Write the formula as 'y ~ (mathematical expressions)'.

For example, to represent the response of the reaction data:

```
modelfun = 'y~(b1*x2 - x3/b5) / (1+b2*x1 + b3*x2 + b4*x3)';
```

Nonlinear Model Fitted Adjustment and Examination of Quality

There are expressive plots to empower you to take a gander at the idea of a model. `plotDiagnostics mdl` gives a combination of plots, including impact and Cook's partition plots. `plotResiduals mdl` gives the complexity between the fitted model and the data.

There are similarly properties of `mdl` that relate to the model quality. `mdl.RMSE` gives the root mean square bumble between the data and the fitted model. `mdl.Residuals.Raw` gives the unrefined residuals. `mdl.Diagnostics` contains a couple of fields, for instance, `Leverage` and `CooksDistance`, that can empower you to recognize particularly interesting observations.

This representation shows how to examine a fitted nonlinear model using symptomatic, for waiting and cut to `plotsLoad` the sample data.

```
load reaction
```

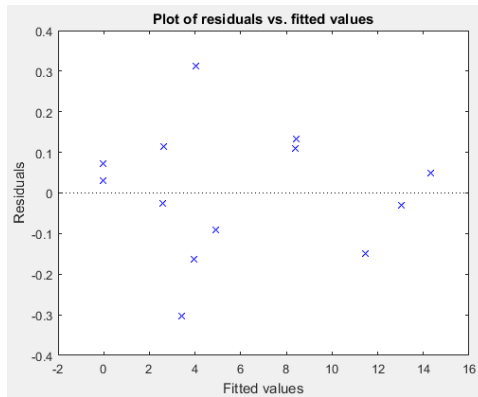
Make a nonlinear model of rate as the capacity of reactants using the `hougen.m` function.

```
beta0 = ones(5,1);
mdl = fitnlm(reactants,...)
      rate,@hougen,beta0;
```

Data and model plot.

```
plotDiagnostics(mdl)
```

```
plotDiagnostics (mdl)
```

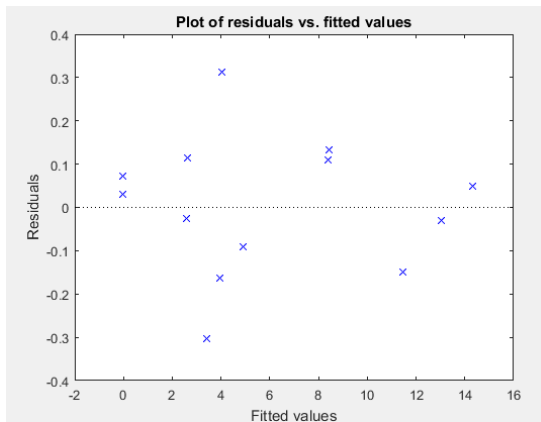


Locate the point with higher leverage.

```
[~,max] = max(mdl.Diagnostics.Leverage)
max1 =
6
```

Residuals plot examination.

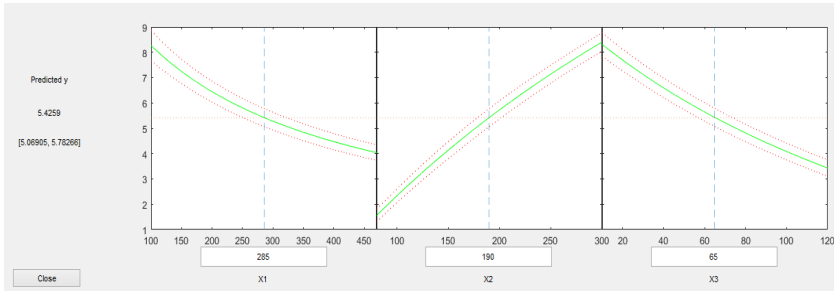
```
plotResiduals(mdl, 'fitted')
```



Outlier standing values.

```
plotSlice mdl)
```

Utilize a cut plot to demonstrate the impact of every indicator on the model.



Nonlinear Model Simulation Prediction or examination

This case demonstrates how to utilize the methods predict, feval and random to anticipate and reproduce reactions to new information.

Haphazardly create an example from a Cauchy conveyance

```
rng('default')
X = rand(100,1);
X = tan(pi*X - pi/2);
```

Model generation based on $y = b_1 * (\pi/2 + \text{atan}((x - b_2) / b_3))$ and response, add noise.

```
modelfun = @(b,x) b(1) * ...
    (pi/2 + atan((x - b(2))/b(3)));
y = modelfun([12 5 10],X) + randn(100,1);
```

Arbitrary parameters model initialization $b = [1, 1, 1]$.

```
beta0 = [1 1 1]; % An arbitrary guess
mdl = fitnlm(X,y,modelfun,beta0)
```

The fitted values are within a few percent of the parameters [12,5,10].

Examine the fit.

Nonlinear regression model:

$$y \sim b1*(\pi/2 + \text{atan}((x - b2)/b3))$$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
b1	12.082	0.80028	15.097	3.3151e-27
b2	5.0603	1.0825	4.6747	9.5063e-06
b3	9.64	0.46499	20.732	2.0382e-37

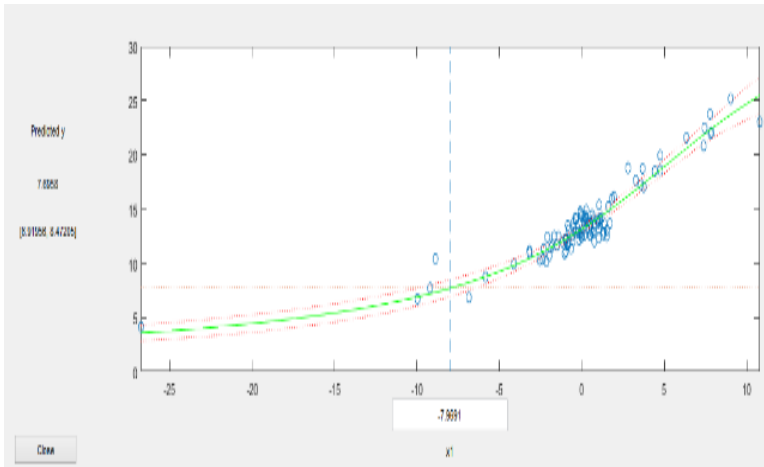
Number of observations: 100, Error degrees of freedom: 97

Root Mean Squared Error: 1.02

R-Squared: 0.92, Adjusted R-Squared 0.918

F-statistic vs. zero model: 6.45e+03, p-value = 1.72e-111

plotSlice mdl



Predict

The predict method predicts the mean reactions and, if asked for, gives certainty limits. Discover the anticipated reaction esteems and anticipate certainty interims about the reaction at X values [-15;5;12].

```

Xnew = [-15;5;12];
[ynew,ynewci] = predict mdl,Xnew

ynew =

    5.4122
   18.9022
   26.5161

ynewci =

    4.8233    6.0010
   18.4555   19.3490
   25.0170   28.0151

```

The confidence intervals are reflected in the slice plot.

Feval

The feval method predicts the mean responses. Feval is often more convenient to use than predict when you construct a model from a dataset array.

Create the nonlinear model from a dataset array.

```

ds = dataset ({X, 'X'}, {Y, 'Y'});
mdl2 = fitnlm(ds,modelfun,beta0);

```

Find the predicted model responses (CDF) at X values [12;5;-15].

```

Xnew = [-15;5;12];
[ynew,ynewci] = predict mdl,Xnew

ynew =

    5.4122
   18.9022
   26.5161

```


Random

The random method simulates new random response values, equal to the mean prediction plus a random disturbance with the same variance as the training data.

```
Xnew = [-15;5;12];
ysim = random mdl, Xnew

ysim =

    6.0505
   19.0893
   25.4647
```

Rerun the random method. The results change.

```
ysim = random mdl, Xnew
ysim =

    6.3813
   19.2157
   26.6541
```

3.10 Techniques for Model Improvement

3.10.1 Selecting Features for Classifying

High-dimensional Data

This case demonstrates how to choose highlights for arranging high-dimensional information. All the more particularly, it demonstrates how to perform consecutive element determination, which is a standout amongst the most famous component choice calculations. Likewise, it demonstrates the utilization of holdout and cross-approval evaluation for executing chosen highlights.

Diminishing quantity highlights (dimensionality) is imperative for factual learning. For some, countless informational collections and a predetermined number of perceptions, for example, bioinformatics information, generally

numerous highlights helpful in delivering a coveted learning result and the restricted perceptions may lead the learning calculation with an overfit commotion. Lessening highlights likewise spares the stockpiling calculation time and increment conceivability.

There are two fundamental ways to deal with decreasing highlights: include determination and highlight change. Highlight determination calculations by selecting a subset of highlights from the first list of capabilities; include change strategies change information from the first high-dimensional element space with diminished dimensionality.

3.10.2 Loading the Data

Outline diagnostics can be used to isolate observations from patients with and without sickness. Profile outlines are made using a surface-updated laser desorption and ionization (SELDI) protein mass spectrometry. These features are molecule control levels at specific mass/charge regards.

```
>> load ovariancancer;
whos
  Name          Size          Bytes  Class  Attributes
  grp          216x1          26784  cell
  obs          216x4000       3456000 single
```

Dividing Data into a Training Set and a Test Set

A bit limits piece of this case called MATLAB worked in a subjective number of age limits

```
rng(8000, 'twister');
```

For planning data, the re-substitution execution is certainly better than the average check for a model's execution on a free test set. Re-substitution execution will usually be over-hopeful. For execution prediction, a certain picked show evaluates the execution on an instructive accumulation used for model development.

Here, to segment information into a plan, set a size of 160 and a test set of size of 56. Both the availability set and the test set have all things considered in a similar get-together degree as in grp. We select highlights utilizing the arranging information and judge the execution of the picked

joins on the test information. This is reliably called holdout underwriting. Another major and broadly utilized system for assessing and picking a model is cross-underwriting, which will be addressed later in this outline.

```
>> holdoutCVP = cvpartition(grp, 'holdout' , 56)
holdoutCVP =
Hold-out cross validation partition
    NumObservations: 216
        NumTestSets: 1
            TrainSize: 160
            TestSize: 56

>> dataTrain = obs(holdoutCVP.training, : ) ;
grpTrain = grp(holdoutCVP.training) ;
```

The Problem of Classifying Data Using All the Features

The quantity highlights is significantly bigger than perceptions quality. In this case, the Quadratic Discriminant Analysis (QDA) is used for order calculation. In the event that we apply QDA on the information utilizing every one of the highlights, as appeared in the accompanying, we will get a blunder in light of the fact that there are insufficient examples in each gathering to evaluate a covariance framework.

```
>> try
    yhat = classify(obs(test(holdoutCVP),:), dataTrain, grpTrain, 'quadratic
catch ME
    display(ME.message);
end
The covariance matrix of each group in TRAINING must be positive definite.
```

Selecting Features Using a Simple Filter Approach

We will likely diminish the measurement of the information by finding a little arrangement of critical highlights which can give great characterization execution.

Highlight determination calculations can be generally assembled into two classes: channel techniques and wrapper strategies. Channel techniques depend on the general attributes information assess choose component subsets without including picked learning calculation (QDA in this case).

Wrapper strategies utilizes the execution of the picked learning calculation to assess every applicant include subset. Wrapper techniques look for highlights that better fit the picked learning calculation, however they can be fundamentally slower than channel strategies if the learning calculation sets aside a long opportunity to run.

Channels are for the most part used as a pre-getting ready progress since they are direct and snappy. A comprehensively used channel procedure for bioinformatics data is to apply a univariate administer autonomously on every part, tolerating that there is no correspondence between features.

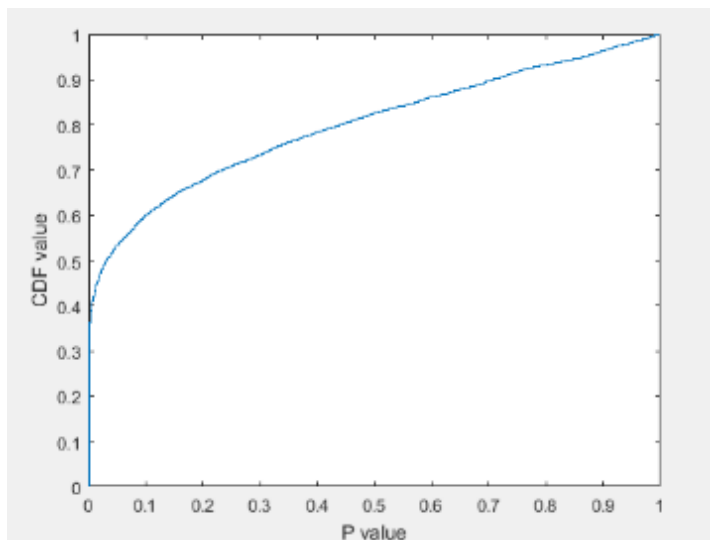
We may apply the t-test on each component and compare p-esteem (or the supreme qualities of t-insights) for each element as a measure of how viable it is at isolating gatherings.

```
>> dataTrainG1 = dataTrain(grp2idx(grpTrain)==1, :);
dataTrainG2 = dataTrain(grp2idx(grpTrain)==2, :);
[h,p,ci,stat] = ttest2(dataTrainG1,dataTrainG2,'Vartype','unequal');
```

For p -values plot empirical cumulative distribution function (CDF).

Program

```
>>ecdf (p) ;
xlabel ( 'P value' ) ;
ylabel ( 'CDF value' ) ;
```



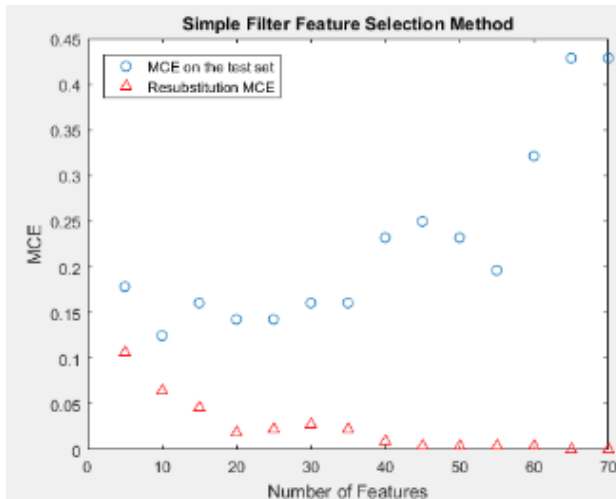
The calculation of p-values provides 35% of value nearer to 0 where it is slightly higher to 0.05. For effective separation it has an excess of 2500 to 5000 separation values. In any case, it is generally hard to choose what number of highlights are required unless one has some space information or the greatest number of highlights that can be considered has been managed ahead of time in light of outside imperatives.

One snappy approach to choose the quantity of require MCE (misclassification mistake, i.e., misclassification perceptions partitioned quantity perceptions) component quantity highlights. There are 160 perceptions for preparing the biggest highlights for restricted QDA.

```
>> [~,featureIdxSortbyP] = sort(p,2); % sort the features
testMCE = zeros(1,14);
resubMCE = zeros(1,14);
nfs = 5:5:70;
classf = @(xtrain,ytrain,xtest,ytest) ...
    sum(~strcmp(ytest,classify(xtest,xtrain,ytrain,'quadratic')));
resubCVP = cvpartition(length(grp),'resubstitution')
for i = 1:14
    fs = featureIdxSortbyP(1:nfs(i));
    testMCE(i) = crossval(classf,obs(:,fs),grp,'partition',holdoutCVP) ...
        /holdoutCVP.TestSize;
    resubMCE(i) = crossval(classf,obs(:,fs),grp,'partition',resubCVP)/...
        resubCVP.TestSize;
end
plot(nfs, testMCE,'o',nfs,resubMCE,'r^');
xlabel('Number of Features');
ylabel('MCE');
legend({'MCE on the test set' 'Resubstitution MCE'},'location','NW');
title('Simple Filter Feature Selection Method');

resubCVP =

Resubstitution (no partition of data)
  NumObservations: 216
    NumTestSets: 1
      TrainSize: 216
        TestSize: 216
,
```

Output

Class is characterized as a mysterious capacity. In the event that you were building up your own grouping calculation, you need to place it in a different document.

The re-substitution shows MCE over-hopeful. Reliability diminishes highlights utilized nearer to zero in the utilization exceeding 60 highlights. Be that as it may, if the test mistake increments while the re-substitution blunder still performs reductions, at that point overfitting may have happened. This straightforward channel includes the determination strategy that gets the littlest MCE on the test set when 15 highlights are utilized. The above plot demonstrates how overfitting starts to happen when at least 20 highlights are utilized. The littlest MCE on the test set is 12.5%

```
>> testMCE(3)
and =
0.1607
```

To achieve minimal MCE 15 features are first evaluated.

```
>> featureIdxSortbyP(1:15)

ans =

Columns 1 through 7
    2814    2813    2650    2338    2452    2645    2897

Columns 8 through 14
    2337    2642    2644    2643    2398    2399    2637

Column 15
    2646
```

3.10.3 Sequential Feature Selection Application

In the above element determination calculation think about the connection of highlights; in addition, highlights chosen from the rundown in light of their individual positioning may likewise contain repetitive data, with the goal that not every one of the highlights are required. For instance, the straight relationship coefficient between the primarily chosen highlight (section 2814) and the second choice included (segment 2813) is very nearly 0.95.

```
corr(dataTrain(:,featureIdxSortbyP(1)),dataTrain(:,featureIdxSortbyP(2)))

ans =

single

    0.9447
```


CHAPTER 4

Data Pre – Processing in Python

4.1 Data Preparation

Machine learning is the process of learning from available data. In this process, the critical thing is feeding of proper data for solving the aimed problems' solution. Even if good quality data is available we need to ensure that useful format, scale, and features with appropriate meaning are included.

4.1.1 Data Preparation Process

More appropriate data needs to be applied for effective data handling in order to achieve better and consistent results. Machine learning data ready process is accomplished under three steps as follows:

1. **Step 1:** Data Selection
2. **Step 2:** Data Pre - Processing
3. **Step 3:** Data Transformation

Now let's understand each step in detail:

Step 1: Select Data

Data selection is the process of selecting the data subset we need to work with. Here, it deals with desirable data availability with maxim for hold, "more is better". Here it comes it may or may not be true.

You clearly need to evaluate the actual question for solving a particular data or problem you are focusing on. Also, some assumptions need to be considered for required data to carefully record the data with those assumptions and test them for a later scenario.

Step 2: Pre - process Data

Once the data selection is completed, consider how the selected data is used for a particular process. Generally, pre-processing is a process of converting a particular data in to another form which can be work.

In pre-processing, data is formatted, sampled, and cleaned.

- **Formatting:** The data you have been assigned would not be in an extremely organized way that is fitting for you to figure with. the data could likewise be in an exceptionally electronic data benefit and you'd am enamored with it in an extremely document, or the data could likewise be in an extremely exclusive record organize and you'd am partial to it in an exceptionally electronic data benefit or a PC document.
- **Cleaning:** Learning is evacuation or settling of the missing information. Likewise, there could be learning cases which are deficient and don't convey the data you speculate, you might want to deal with the issue. These examples may must be evacuated. what's more, there could likewise be delicate information in some of the traits and these ascribes may anonymize or go a long way from the data totally.
- **Sampling:** Similarly, there could be much more assigned learning realistic than you might want to figure with. A considerable measure of information may bring about any long running circumstances for calculations, greater processes, and memory needs. For fitting example, circulation at littler space information is procured with finished datasets for investigation and prototyping.

Step 3: Data Transformation

- The last advance is to rebuild the strategy learning. The exact equation you're working with and furthermore the information of space can impact this progression and you'll horrendously, without a doubt, get the chance to return very surprising changes of your pre-processed learning as you're utilized on your concern.

Three normal information changes are scaling, characteristic deteriorates, and property collections. In the field of highlight designing, extra factors are specified as follows:

- **Scaling:** The pre-handled learning could contain traits with a blends of scales for various amounts like bucks, kilograms, and deals volume. A few machine learning methodologies like information ascribes to have indistinguishable scale like in the vicinity of zero and one at the smallest and biggest cost for a given element. Consider any component scaling you should be constrained to perform.
- **Decomposition:** There could be alternatives that speak to a favor develop that will be a ton of accommodating into a machine learning procedure once split into the constituent components. Relate degree case might be a date that will have day and time parts that progressively may well be part out any. Possibly, exclusively the hour of the day has importance to the issue being illuminated, think about what decay you'll perform.
- **Aggregation:** There could be alternatives that might be mass into one component and a considerable measure of substantial to the issue you're endeavoring to determine. For instance, there could be an information occasion each time a customer signed into a framework that may be mass into a mean the amount of login allowed in the additional examples to be disposed of. Consider what style of highlight accumulations may perform.

4.2 Feature Selection for Machine Learning

In feature selection, particular datasets are used for processing, consider the example of Pima Indians datasets about diabetes. Based on numerical attributes, the problem with classification arises.

Univariate Selection

For certain statistical analysis, certain features are developed for output variable relationship estimation between variables.

For few features, SelectkBest classes are used with specific points in scikit-learn library database.

In case of statistical analysis, process Pima Indian onset with four feature datasets are examined with chi-square (χ^2) statistical analysis and an example is presented as follows:

```

# Feature Extraction with Univariate Statistical Tests (Chi-squared for classifi
import pandas
import numpy
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
test = SelectKBest(score_func=chi2, k=4)
fit = test.fit(X, Y)
# summarize scores
numpy.set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)

# summarize selected features
print(features[0:5,:])

[ 111.52   141.887   17.605   53.108  2175.565  127.669   5.393
 181.304]
[[ 148.   0.   33.6  50. ]
 [  85.   0.   26.6  31. ]
 [ 183.   0.   23.3  32. ]
 [  89.  94.  28.1  21. ]
 [ 137. 168.  43.1  33. ]]

```

4.3 Recursive Feature Elimination

The Recursive Feature Elimination (or RFE) works by recursively evacuating characteristics and building a model on the properties that remain.

It utilizes the model precision to recognize which characteristics (and blend of qualities) contribute the most to foreseeing the objective attribute. You can take in more about the RFE class. The following case utilizes RFE with the strategic relapse calculation to choose the main three highlights:

```

# Feature Extraction with RFE
from pandas import read_csv
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# load data
url = "https://raw.githubusercontent.com/jbrownlee/
Datasets/master/pima-indians-
names = ['preg', 'plas', 'pres', 'skin', 'test',
'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values

```

```

X = array[:, 0:8]
Y = array[:, 8]
# feature extraction
model = LogisticRegression( )
rfe = RFE(model, 3)
fit = rfe.fit(X, Y)
print( ("Num Features: %d") % (fit.n_features_ ) )
print( ("Selected Features: %s") % (fit.support_ ) )
print( ("Feature Ranking: %s") % ( fit.ranking_ ) )

```

These are marked “True” in *support_array* and marked with a choice “1” in the *ranking_array*:

```

Num Features: 3
Selected Features: [ True False False False False  True  True False]
Feature Ranking: [1 2 3 5 6 1 1 4]

```

4.4 Principal Component Analysis

For conversion of datasets into compressed data format, linear algebra is used in Principal Component Analysis (PCA).

Further, this is also known as general class of data reduction technique. The primary property of PCA is the number of dimensions can be chosen on our own or in the transformed results principal component are observed.

In the following example, three principal components are used for PCA:

```

# Feature Extraction with PCA
import numpy
from pandas import read_csv
from sklearn.decomposition import PCA
# load data
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test',
         'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:, 0:8]
Y = array[:, 8]
# feature extraction
pca = PCA(n_components=3)

```

```

fit = pca.fit(X)
# summarize components
print( ("Explained Variance: %s") % (fit.explained_
variance_ratio_ ) )
print(fit.components_)

Explained Variance: [ 0.88854663  0.06159078  0.02579012]
[[-2.02176587e-03  9.78115765e-02  1.60930503e-02  6.07566861e-02
  9.93110844e-01  1.40108085e-02  5.37167919e-04 -3.56474430e-03]
 [ -2.26488861e-02 -9.72210040e-01 -1.41909330e-01  5.78614699e-02
  9.46266913e-02 -4.69729766e-02 -8.16804621e-04 -1.40168181e-01]
 [ -2.24649003e-02  1.43428710e-01 -9.22467192e-01 -3.07013055e-01
  2.09773019e-02 -1.32444542e-01 -6.39983017e-04 -1.25454310e-01]]

```

4.5 Feature Importance

In scikit-learn, API for estimating the feature of Extra Trees Classifier Bagged decision trees are used which is similar to Random Forest and Extra Trees.

```

# Feature Importance with Extra Trees Classifier
from pandas import read_csv
from sklearn.ensemble import ExtraTreesClassifier
# load data
filename = "pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
model = ExtraTreesClassifier()
model.fit(X, Y)
print(model.feature_importances_)

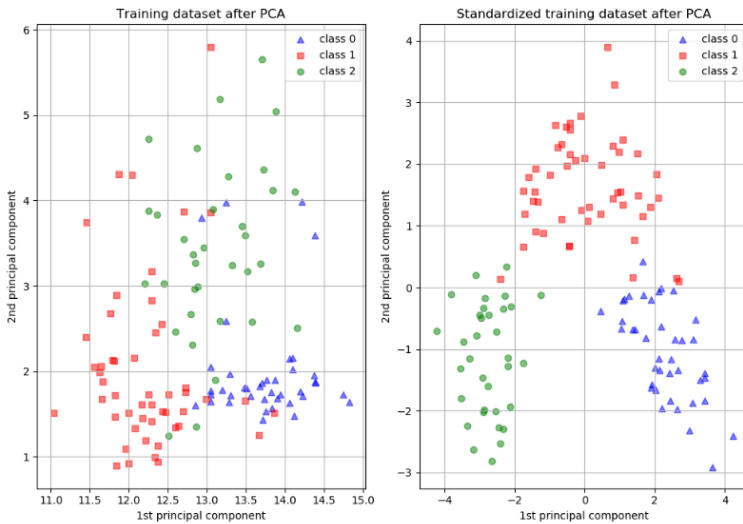
[ 0.1061128  0.26666157  0.09512675  0.07437106  0.0748202  0.13575556
  0.11585816  0.13129392]

```

4.6 Feature Scaling

Feature scaling is standardization (or Z-score normalization). It is a very important pre-processing step for several machine learning algorithms. Standardization involves rescaling the options specified, they need the properties of a customary distribution with a mean of zero and a customary deviation of 1.

While several algorithms (such as SVM, K-nearest neighbors, and supplying regression) need options to be normalized, intuitively, we will consider Principle Component Analysis (PCA) as being a chief example of once standardization is vital. In PCA, we tend to have an interest within the parts that maximize the variance. If one element (for example, human height) varies from the other (for example, weight), thanks to their individual scales (meters versus kilos), PCA would possibly confirm that the direction of largest variance additional closely corresponds with the ‘weight’ axis, if those options don’t seem to be scaled. As a modification tall of 1 meter is thought of far more vital than the modification in weight of 1 kg, this is often clearly incorrect.



To illustrate this, PCA scrutinizes the utilization of knowledge with Standard Scaler applied, to un-scaled information. The resultant square measure can be visualized and a transparent distinction noted. The first principal element within the upscale set is seen. It is seen that feature #13 dominates the direction, being an entire 2 orders of magnitude higher than the opposite options. This is often contrasted once observant the principal element for the scaled version of the info. Within the scaled version, the orders of magnitude square measure roughly constant across all the options.

The dataset used is the Wine dataset out there at UCI. This dataset has continuous options that square measure heterogeneous in scale thanks to

differing properties that they measure (that is, alcohol content, and malic acid).

The reworked information is then accustomed trained a naive mathematician classifier, and a transparent distinction in prediction accuracy is ascertained whereby the dataset that is scaled before PCA immensely outperforms the unscaled version.

```

from __future__ import print_function
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.pipeline import make_pipeline
print(__doc__)

# Code source: Tyler Lanigan <tylerlanigan@gmail.com>
#              Sebastian Raschka <mail@sebastianraschka.com>

# License: BSD 3 clause

RANDOM_STATE = 42
FIG_SIZE = (10, 7)

features, target = load_wine(return_X_y=True)

# Make a train/test split using 30% test size
X_train, X_test, y_train, y_test = train_test_split(features, target,
                                                    test_size=0.30,
                                                    random_state=RANDOM_STATE)

# Fit to data and predict using pipelined GNB and PCA.
unscaled_clf = make_pipeline(PCA(n_components=2), GaussianNB())
unscaled_clf.fit(X_train, y_train)
pred_test = unscaled_clf.predict(X_test)

# Fit to data and predict using pipelined scaling, GNB and PCA.
std_clf = make_pipeline(StandardScaler(), PCA(n_components=2), GaussianNB())
std_clf.fit(X_train, y_train)
pred_test_std = std_clf.predict(X_test)

# Show prediction accuracies in scaled and unscaled data.
print('\nPrediction accuracy for the normal test dataset with PCA')
print('{:.2%}\n'.format(metrics.accuracy_score(y_test, pred_test)))

# Extract PCA from pipeline
pca = unscaled_clf.named_steps['pca']
pca_std = std_clf.named_steps['pca']

# Show first principal components
print('\nPC 1 without scaling:\n', pca.components_[0])
print('\nPC 1 with scaling:\n', pca_std.components_[0])

# Scale and use PCA on X_train data for visualization.
scaler = std_clf.named_steps['standardscaler']
X_train_std = pca_std.transform(scaler.transform(X_train))

# visualize standardized vs. untouched dataset with PCA performed
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=FIG_SIZE)

```



```

for l, c, m in zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):
    ax1.scatter(X_train[y_train == l, 0], X_train[y_train == l, 1],
               color=c,
               label='class %s' % l,
               alpha=0.5,
               marker=m
              )

for l, c, m in zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):
    ax2.scatter(X_train_std[y_train == l, 0], X_train_std[y_train == l, 1],
               color=c,
               label='class %s' % l,
               alpha=0.5,
               marker=m
              )

ax1.set_title('Training dataset after PCA')
ax2.set_title('Standardized training dataset after PCA')

for ax in (ax1, ax2):
    ax.set_xlabel('1st principal component')
    ax.set_ylabel('2nd principal component')
    ax.legend(loc='upper right')
    ax.grid()

plt.tight_layout()

plt.show()

```

```

Prediction accuracy for the normal test dataset with PCA
81.48%

```

```

Prediction accuracy for the standardized test dataset with PCA
98.15%

```

```

PC 1 without scaling:

```

```

[ 1.76342917e-03 -8.35544737e-04 1.54623496e-04 -5.31136096e-03
 2.01663336e-02 1.02440667e-03 1.53155502e-03 -1.11663562e-04
 6.31071580e-04 2.32645551e-03 1.53606718e-04 7.43176482e-04
 9.99775716e-01]

```

```

PC 1 with scaling:

```

```

[ 0.13443023 -0.25680248 -0.0113463 -0.23405337 0.15840049 0.39194918
 0.41607649 -0.27871336 0.33129255 -0.11383282 0.29726413 0.38054255
 0.27507157]

```

4.7 Seven Ways to Handle Large Data Files for Machine Learning

1. Allocate More Memory

There are seven ways in which you can handle giant information files for machine learning:

1. Memory Allocation Portion

Machine learning or libraries could be confined by a default memory design.

In the event, check that you'll have the capacity to re-arrange your instrument or library to partition a considerable measure of memory.

A decent case, wherever to build memory parameter for starting and applying.

2. Work with a Smaller Sample

Is it safe to say you are sure that you wish to figure with the greater part of the information?

Take an unpredictable case of your information, like the fundamental 1,000 or 1,00,000 lines. Use this tinier case to figure through your drawback before fitting a keep going model on the greater part of your information (using dynamic information stacking systems).

I think this is routinely a respectable take after conventionally for machine making sense of how to give you speedy spot-checks of counts and turnaround of results.

Moreover, you may consider acting an affectability examination of the measure of information accustom organized one algorithmic program appeared differently in relation to the model capacity. Maybe there's a trademark explanation behind consistent losses that you basically will use as a heuristic size of your more diminutive illustration.

3. Utilization of computer for more memory allocation

Did you get the opportunity to work away at your PC?

Possibly, you'll have the ability to pick up induction to a way greater PC with Associate in Nursing solicitation of degree a significant measure of memory.

For example, not all that terrible choices are to rent figure time on a cloud advantage like Amazon net Services that outfits machines with numerous gigabytes of RAM for not as much as a North American nation dollar consistently.

I have found this approach horrendously strong inside the past.

Have a look at the post:

The most effective method to Develop and valuate goliath Deep Learning Models with Keras on Amazon net Services

4. Amendment of Data Format

Is your data kept in crude American Standard Code for Information Interchange content, kind of a CSV document?

Maybe you'll have the capacity to accelerate data stacking and utilize less memory by abusing another information. A fair illustration might be a paired arrangement like GRIB, NetCDF, or HDF.

The square measure of a few summon instruments that you essentially will use to revise one information into another that needn't bother with the total datasets to be stacked into memory.

Utilizing another typeat could allow you to store the data in an exceptionally conservative shape that spares memory, similar to 2-byte numbers, or 4-byte drifts.

5. Stream Data or Progressive Loading

Does all the data have a chance of comparable time to get?

For effective utilization of library stream or code progressive stack data will be required for training data in memory.

This may require estimations that may learn iterative mishandle change procedures like discretionary slant fall, instead of computations that need all information in memory to perform and organize exercises like a couple of use of direct and plan backslide.

For example, the Keras significant learning library offers this component for logically stacking picture archives and is called `flow_from_directory`.

Another case is that the Pandas library that can stack mammoth CSV records in knots.

6. Relational Database Usage

Social databases are often normal range for putting away and obtaining appallingly mammoth datasets.

Inside, the data is kept in circle might be progressively stacked in clusters and might be questioned utilizing an ordinary inquiry dialect (SQL).

Free open supply information devices such as Postgres or MySQL are utilized for programming dialects and in machine learning apparatuses for interface of relative databases. Furthermore, you have the capacity to utilize a light-weight approach, for example, SQLite.

I have observed this way to deal with be horrendously compelling inside the past for awfully goliath forbidden datasets.

Once more, you'll get the opportunity to utilize calculations that may deal with unvaried learning.

7. Big Data Utilization Platform

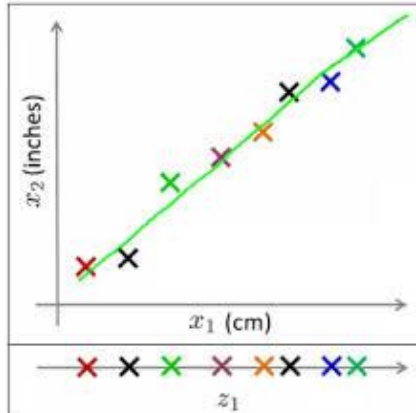
In fewer cases, you'll get abundant information data platform.

In this platform, big data are handled with datasets which permits information transforming and prime machine learning algorithms.

Hadoop and machine learning algorithm measures square measures for SarkMLLib library for processing.

At top of the choice, in final resort, the values are exhausted with extra code and hardware for complex machine learning algorithms.

However, the square measure issues wherever the information is incredibly giant and also the previous choices won't cut it.



4.8 Dimensionality Reduction

Dimension Reduction alludes to the technique for changing a gathered data, having tremendous measurements into information, with lesser measurements guaranteeing that it passes on comparable data in short. These systems square measure for the most part which is utilized while taking care of the machine learning issues to acquire better choices for an order or relapse errand.

We should investigate the picture demonstrated as follows. It indicates two measurements x_1 and x_2 , the square measure enables us to state estimations of numerous protest in cm (x_1) and inches (x_2). Presently,

if you somehow managed to utilize each of these measurements in machine learning, they will pass on comparable data and present loads of commotion in the framework. In this way, you're higher of basically abuse one measurement. Here, we have renewed the measurement of data from second (from x_1 and x_2) to 1D (z_1), that has made the data relatively less demanding to clarify.

In comparable manners by which, we can downsize n measurements of dataset to k measurements ($k < n$). These k measurements might be specifically known (sifted) or might be a blend of measurements (weighted midpoints of measurements) or new dimension(s) that speak to existing different measurements well.

One of the first basic utilization of this strategy is image process. You would conceivably have returned over this Facebook application—"Which Celebrity Do You Look Like?". Here's the appropriate response: to set up the coordinated superstar picture, we tend to utilize segment data and each segment is the same as one measurement. In each picture, there are high number of pixels that is, high assortment of measurements. Also, each measurement is indispensable here. You can't discard measurements arbitrarily to understand your general dataset. In such cases, measurement diminish systems help you to look out the various dimension(s) abuse various method(s). We'll talk about these procedures in the blink of an eye.

4.9 Cross Validation

Model Performance for Assess and Predictive Improvisation

Cross-validation is an appraisal model strategy for assessing machine learning algorithms execution to forecast newly defined dataset created. To achieve this, dividing datasets are utilized as a subset for calculating the rest of the information for testing. Since cross-approval does not utilize the information greater part model separation, usually utilized technique forestall amid preparation.

Every cross-approval includes arbitrarily apportioning the first classified as testing and training dataset. In supervised learning, for execution purpose, testing data is used and training data is used for creation of data. Method rehashed a few counts and normal cross-approval blunder utilized an execution pointer.

Normal cross-validation strategies contain:

K-fold: Information is divided into k haphazardly picked subsets (generally approach estimate). In this a single subset is utilized approve prepared utilizing rest of the subsets. The steps are rehashed k counts with the end goal that every subset is utilized precisely for approval.

Holdout: Classification information precisely has two subsets: indicated proportion of preparation and approval.

Leaveout: Information classification utilizing k -means clustering where k equivalent to the aggregate perceptions information number. Otherwise called forget one cross-approval.

Rehashed arbitrary sub-inspecting: Performs Monte Carlo repetition of arbitrarily parceling information and totaling comes about finished everyone of the runs.

Stratify: Partitions information with the end goal preparing and testing data generally a similar extent reaction or outcome.

Re-substitution: No information are parcel; utilizes preparation information approval. Regularly creates excessively idealistic evaluations for execution and must maintain a strategic distance from adequate information.

Cross-validation computation is a serious task preparing and approving the completed few counts. Since the parcel set are autonomous, examinations are performed parallel with procedure accelerate.

Feature Selection

Recognize persuasive highlights to enhance demonstrate execution.

Highlight determination is a dimensionality lessening method that chooses just a subset of estimated highlights (indicator factors) that give the best prescient power in demonstrating the information. It is especially helpful when managing high-dimensional information or when displaying with all highlights is unwanted.

Preamble of Feature Selection

Feature selection minimizes information dimensionality choosing just dataset estimated highlights (indicator factors) to make a model. Choice criteria normally includes a particular measure reduction prescient blunder to fit models of different subsets. Calculation subset indicator ideally shows the estimated reactions, subject to limitations, for example, required or rejected highlights and the span of the subset.

Highlight choice is desirable over component change when the first units and significance of highlights are essential and the demonstrating objective is to recognize a powerful subset. At the point when absolute highlights are available and numerical changes are improper, include determination turns into the essential methods for measurement diminishment.

Highlight choice are utilized to:

Enhance precision count through machine learning algorithms

Lift execution with information of high-dimensionality

Interpretability display enhancement

Overfitting of Forestall

A few basic ways to deal with include choice:

Stepwise regression sequential include expel highlights until there is a change as expectation; linear regression utilization with summed up straight relapse calculations.

Sequential feature selection performs step-wise relapse to be utilized with supervised learning algorithm and execution time with customized manner.

Boosted and bagged decision trees troupe strategies to process different significance from out-of-sack gauges.

Regularization stated as estimator shrinkage for expel repetitive highlights lessening weights to zero.

Dimensionality diminish approach utilization includes highlight change systems, which existing highlights indicator factors in less unmistakable highlights.

Ways deal with highlight change contains:

Principal component analysis (PCA) to abridge information less measurements projection with special orthogonal premise

Factor analysis, used to fabricate informative models of information relationships

Nonnegative matrix factorization demonstrate speaking to non-negative amounts, for example, physical amounts

Sequential Feature Selection

A typical technique for include choice of Sequential Feature Selection. The strategy has two parts:

A goal work criterion technique tries to limit attainable element. Regular mean squared blunder (relapse models) and misclassification rate.

Successive hunt calculation expels highlights competitor subset for assessing the basis. Through correlation measure and incentive $2n$ subsets and n -include informational collection regularly in feasible (contingent upon n and the cost of target calls), successive hunts just a single bearing, continually developing or continually contracting the hopeful set.

Sequential forward selection (SFS), is defined as highlights consecutively unfilled hopeful point that the option highlight minimizes the measure.

Sequential in reverse selection (SBS) highlighted successively expelled for full applicant, the point that the expulsion of further highlights increment the foundation.

Step-wise relapse is a successive component determination procedure outlined particularly for slightest squares fitting. The step-wise fit utilization enhancements are just conceivable for minimum squares. Dissimilar to sum up consecutive component choice, step-wise relapse may evacuate highlights that have been included or include highlights that have been expelled.

Predictive Power for selection of subset Features

Consider an informational index with 100 perceptions of 10 indicators. Produce the irregular information of calculated model, the binomial circulation reactions arrangement qualities indicators. A few coefficients are zero with the goal which are greater part indicators influences reaction.

```
rng(456) % Editor: E. Data Analytics/Data Science/pauls/patterns
n = 100;
m = 10;
X = rand(n,m);
b = [1 0 0 2 .5 0 0 0.1 0 1];
Xb = X*b';
p = 1./(1+exp(-Xb));
N = 50;
y = binornd(N,p);
```

Fit a logistic model to the data using fitglm:


```

Y = [y N*ones(size(y))];
model0 = fitglm(X,Y,'Distribution','binomial')

model0 =

Generalized Linear regression model:
  y ~ [Linear formula with 11 terms in 10 predictors]
  Distribution = Binomial

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	0.22474	0.30043	0.74806	0.45443
x1	0.68782	0.17207	3.9973	6.408e-05
x2	0.2003	0.18087	1.1074	0.26811
x3	-0.055328	0.18871	-0.29319	0.76937
x4	2.2576	0.1813	12.452	1.3566e-35
x5	0.54603	0.16836	3.2432	0.0011821
x6	0.069701	0.17738	0.39294	0.69437
x7	-0.22562	0.16957	-1.3306	0.18334
x8	-0.19712	0.17317	-1.1383	0.25498
x9	-0.20373	0.16796	-1.213	0.22514
x10	0.99741	0.17247	5.7832	7.3296e-09

```

100 observations, 89 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 222, p-value = 4.92e-42

```

Display the deviance of the fit:

```

dev0 =

    101.5648

```

Perform feature selection. Sequentialfs calls the criterion function via a function handle:

```

maxdev = chi2inv(.95,1);
opt = statset('display','iter',...
            'TolFun',maxdev,...
            'TolTypeFun','abs');

inmodel = sequentialfs(@critfun,X,Y,...
                    'cv','none',...|
                    'nullmodel',true,...
                    'options',opt,...
                    'direction','forward');
model = fitglm(X(:,inmodel),Y,'Distribution','binomial')

```

Sequential Forward feature selection:

Initial columns value: none

Columns that cannot be included: none

Step 1, used initial columns, criterion value 323.173

Step 2, added column 4, criterion value 184.794

Step 3, added column 10, criterion value 139.176

Step 4, added column 1, criterion value 119.222

Step 5, added column 5, criterion value 107.281

Final columns included: 1 4 5 10

Iterative value shown demonstrates decline basis esteem new element addition to model. Final outcome diminished just 4 among 10 values: columns 1, 4, 5, and 10 for X, as demonstrated in the legitimate vector in model returned by sequentialfs.

4.10 Feature Transformation

Feature transformation methods decrease the dimensionality in the information by changing information into new highlights.

Transform

Transform predictors into extracted features.

Syntax

```
z = transform(Mdl,x)
```

Description

Example:

$z = \text{transform}(\text{Mdl}, x)$ transforms the data x into the features z via the model Mdl .

Transform Data to Learned features

Make an element change demonstrate with 100 highlights from the caltech101patches data.

CHAPTER 5

Building Regression Models

5.1 Parametric regression Methods

Learning Function

Machine learning is condensed by taking capacity (f) with input factors (X) for achieving yield factors (Y).

$$Y = f(x)$$

A calculation takes in this objective mapping capacity from preparing information.

The type of the capacity is obscure, which act as machine learning specialists for assessing distinctive machine learning calculations for better approximation of fundamental capacity.

Presumptions can extraordinarily rearrange the learning procedure, yet can restrain are realized. Calculations disentangle capacity of the frame provide machine learning calculation parameters.

The calculations include two stages:

1. For a frame capacity choose frame.
2. Coefficient of function are evaluated for information learning selection.

A straightforward useful shape for the mapping capacity is a line, as is utilized as a part of direct relapse:

$$b_0 + b_1 * x_1 + b_2 * x_2 = 0$$

In the preceding equation, b_0 , b_1 , and b_2 represent line capture and incline coefficients, x_1 and x_2 are two info factors.

Expecting practical type incredibly line streamlines provides learning procedure. Presently, in this appraise line coefficients condition and prescient model issue.

Frequently accepted useful shape direct mix information factors of all things considered machine learning parameter calculations which are regularly stated as “linear machine learning calculations”.

The problem associated with genuine obscure basic capacity direct capacity similar to line. This is very close to the line with minimal change in information of work. On the other hand, it could be not at all like a line in which case the suspicion isn't right and the approach will create poor outcomes.

Parametric model, we have a limited number of parameters, parametric models, we have a settled number of parameters.

The “non-parametric” expression might sound somewhat confounding at first: non-parametric does not imply that they have no parameters! Despite what might be expected, non-parametric models (can) turn out to be increasingly intricate with an expanding measure of information.

A few parameters of machine learning include:

- Logistic Regression
- Linear Discriminant Analysis
- Perceptron
- Naive Bayes
- Simple Neural Networks

Advantages of machine learning algorithms:

- Simpler: Techniques less demanding for comprehend to translate comes about.
- Speed: For gaining information model with parametric factor
- Less Data: It does not require information preparation and works effectively, regardless of whether the information is great or not

Constraints in machine learning algorithms:

- Constraint: The practical frame strategies profoundly compel predetermined shape.
- Limited Complexity: Techniques are easier issues for appropriate approaches.
- Poor Fit: These techniques are far-fetched coordinate with fundamental mapping capacity.

5.2 Nonparametric Machine Learning Algorithms

Calculations which don't influence the solid presumptions type mapping to work are called non-parametric machine learning. Through suspicions, allowed take in practical shape information creation.

Nonparametric strategies look to best fit the preparation information in developing the mapping capacity, while keeping up some capacity to sum up to inconspicuous information. In that capacity, they can fit a substantial number of utilitarian structures.

A straightforward non-parametric model provides k-means calculation to forecasts in light of the k most comparative, preparing designs for another information occurrence. The expect mapping capacity type without designs closer to comparative yield variable.

Nonparametric models, the quantity of parameters is (conceivably) boundless. On the other hand, in nonparametric models, the intricacy of the model develops with the quantity of preparing information.

Fewer classifications of machine learning algorithms are:

- K-Nearest Neighbors
- Decision Trees like CART and C4.5
- Support Vector Machines

Advantages of machine learning algorithms:

- Flexibility: Countless structure fitting capability
- Power: No suppositions (or frail presumptions) about the hidden capacity
- Performance: Can bring about higher execution models for expectation

Restrictions of machine learning algorithms:

- Higher information: To significant all the preparing information assess capacity mapping.
- Slower: Great deal minimizes frequently significantly for preparation of parameters.
- Over-fitting: More of a hazard overfit preparation information, it is difficult to clarify particular forecasts selection.

Choose the regression functions

Continuous response, Continuous, or categorical predictors and linear model	Coefficients of fitted model	fitlm. Refer <u>Linear Regression</u> .
Predictors in Continuous or categorical, continuous response, Unknown level of complexity in linear model	Fitted coefficients and fitted model	stepwiselm. Refer Stepwise Regression
Continuous or categorical predictors, Non - negative or integer-valued response possible with restrictions, generalized linear model	Fitted generalized linear model coefficients	fitglm or stepwiseglm. Refer Generalized Linear Models
Continuous nonlinear response of Continuous predictors, Parameterized nonlinear model	Nonlinear model coefficients fitting	fitnlm. Refer Nonlinear Regression
Continuous predictors, continuous response, linear model	Models for ridge, lasso or elastic net regression	lasso or ridge Refer Lasso and Elastic Net or Ridge Regression
Correlated continuous predictors, continuous response, linear model	Fitted coefficients and Fitted model	plsregress Refer Partial Least Squares

Continuous or categorical predictors, continuous response, unknown model	Non - parametric model	fitrtree or fitrensemble See
Categorical predictors only	ANOVA	Anova,anova1,anovan,anova2
Continuous predictors, multivariable response, linear model	Fitted multivariate regression model coefficients	mvregress
Continuous predictors, continuous response, mixed-effects model	Fitted mixed-effects model coefficients	nlmefit or nlmefitsa Refer Mixed-Effects Models

5.3 Evaluation of Regression Models

Superclasses: CompactGeneralizedLinearModel

Generalized linear regression model class

A question containing preparing information, display portrayal, indicative data, and fitted coefficients for a summed up direct relapse. Foresee reactions with the prediction or feval approach.

Construction

`mdl = fitglm(tbl)` or `mdl = fitglm(X,y)` makes summed up straight design in the format of a table or a dataset array `tbl`, information matrix is represented as `X` and response matrix is denoted as `y` to identify the points of interest, refer `fitglm`.

`mdl = stepwiseglm(tbl)` or `mdl = stepwiseglm(X,y)` makes summed up direct format in a table or a dataset array `tbln` with information matrix `X` and response matrix `Y`, with immaterial indicators prohibited to identify points of interest, refer `stepwiseglm`.

Fit a Generalized Linear Model

In linear relapse, model of likelihood factors associated with smokers are evaluated based on the age, weight, and sex, which utilizes demonstrate association in two-way.

Dataset cluster based on hospital are loaded:

```
load hospital
ds = hospital; % just to use the ds name
```

Specify the model using a formula that allows upto two-way interactions:

```
modelspec = 'Smoker ~ Age*Weight*Sex - Age:Weight:Sex';
```

Create the generalized linear model:

```
mdl = fitglm(ds,modelspec,'Distribution','binomial')
```

```
mdl =

Generalized Linear regression model:
  Smoker ~ [Linear formula with 7 terms in 3 predictors]
  Distribution = Binomial

Estimated Coefficients:

              Estimate          SE          tStat          pValue
-----
(Intercept)      -6.0492         19.749         -0.3063         0.75938
Sex_Male         -2.2859         12.424         -0.18399        0.85402
Age              0.11691         0.50977         0.22934         0.81861
Weight           0.031109        0.15208         0.20455         0.83792
Sex_Male:Age     0.020734        0.20681         0.10025         0.92014
Sex_Male:Weight  0.01216         0.053168        0.22871         0.8191
Age:Weight       -0.00071959     0.0038964       -0.18468        0.85348

100 observations, 93 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 5.07, p-value = 0.535
>>
```

Linear Model Stepwise Creation in Generalized Manner

Make reaction information utilizing only three of the 20 indicators and make a summed up straight model step-wise to check if it utilizes only the right indicators. Make information with 20 indicators and Poisson reaction utilizing only three of the indicators, in addition to a steady.

```

rng default % for reproducibility
X = randn(100,20);
mu = exp(X(:, [5 10 15]) * [.4; .2; .3] + 1);
y = poissrnd(mu);

```

With the use of Poisson distribution generalized linear model will be evaluated.

```

mdl = stepwiseglm(X,y,...
    'constant','upper','linear','Distribution','poisson')

```

1. Adding x5, Deviance = 134.439, Chi2Stat = 52.24814, PValue = 4.891229e-13
2. Adding x15, Deviance = 106.285, Chi2Stat = 28.15393, PValue = 1.1204e-07
3. Adding x10, Deviance = 95.0207, Chi2Stat = 11.2644, PValue = 0.000790094

```
mdl =
```

Generalized Linear regression model:

```
log(y) ~ 1 + x5 + x10 + x15
Distribution = Poisson
```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	1.0115	0.064275	15.737	8.4217e-56
x5	0.39508	0.066665	5.9263	3.0977e-09
x10	0.18863	0.05534	3.4085	0.0006532
x15	0.29295	0.053269	5.4995	3.8089e-08

100 observations, 96 error degrees of freedom

Dispersion: 1

Chi^2-statistic vs. constant model: 91.7, p-value = 9.61e-20

CHAPTER 6

Creating Neural Networks

6.1 Self-organizing the Maps and their use in Obtaining K-Clusters

In spite of the fact that the expression “Self-Organizing Map” could be connected to various distinctive methodologies, we should utilize it as an equivalent word of Kohonen’s Self Organizing Map (SOM), otherwise called Kohonen Neural Networks. The fundamental thought of a SOM is to delineate information designs onto an n-dimensional matrix of neurons or units. That network shapes what is known as the yield space, instead of the information space where the information designs are. This mapping tries to protect topological relations, that is, designs that are shut in the info space will be mapped to units that are shut in the yield space and the other way round. In order to permit a simple representation, the yield space is normally one or two dimensional.

The essential SOM preparing calculation can be portrayed as taken after the area work h is generally a capacity with the separation (in the yield space) to the triumphant unit with charge of the communications between various units. Amid preparing, the sweep of this capacity will normally diminish, so every unit will turn out to be more detached from the impacts of its neighbors. It is imperative to take note of the numerous executions of SOM diminish this sweep to 1, implying that even in the last phases of preparing every unit will affect its closest neighbors, while different usage will enable this parameter to be lessened to zero.

SOMs can be utilized as a part of a wide range of courses even inside bunching assignments. In this paper, we will accept that every SOM unit is a group focus and along these lines a k-unit SOM will play out an errand like k-implies. It must be noticed that SOM and k-implies calculations are thoroughly indistinguishable when the range of the area of work in the SOM measures up to zero.

Selforg map

Self-organizing map

Syntax

`selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn)`

Self-organizing our maps figure out how to bunch information in view of likeness and topology, with an inclination (yet no certification) of allocating similar occurrences in every classes.

Self-organizing our utilized bunch of information, diminishes the information dimensionality. In this roused with tactile and engine well evolved creature cerebrum, which appears to naturally sort out data network.

Selforgmap (dimensions, coverSteps, distanceFcn, initNeighbor, topologyFcn) consider the arguments

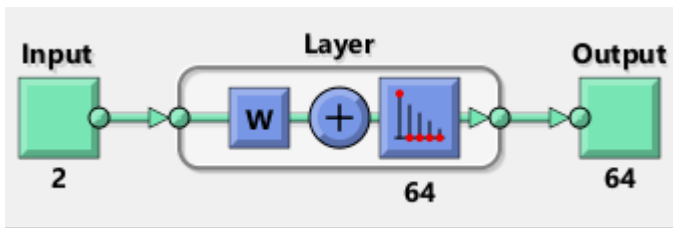
Utilize cluster data in self-organizing map

In self-sorting our guide utilized by a bunch of straightforward arrangement information.

Program:

```
>> x = simplecluster_dataset;  
net = selforgmap ( [8 8] ) ;  
net = train (net,x) ;  
view (net)  
y = net (x)  
classes = vec2ind (y);
```

Output:



6.2 Classification with Feed-Forward Networks

Cluster with Self-Organizing Map Neural Network

Self-organizing feature maps (SOFM) makes sense for portraying input vectors which showed accumulated data space. By differentiated centered layers in the self-dealing, performed with neighboring neurons to control makes sense to the data space of nearby sections. Consequently, mapping of self-organizing network provides a wider range of (as do forceful layers) topologies for learning input vectors.

SOFM layer neurons are sorted out at first, based on the physical positions of the network. Other MATLAB functions such as gridtop, randtop, or hextop coordinate neurons in hexagonal, grid, or interconnected network. Through the classification of work neuron, detachments are evaluated with different circumstances. Neuron detachment are evaluated with the boxdist, mandist, dist, and linkdist functions. For clear understanding of neurons, isolations are depicted in the topology with certain distance limits. In this, the distance between the neurons are calculated with the gridtop, hextop, and randtop functions; distance functions are given as dist, linkdist, mandist, and boxdist.

Here, the self-dealing component depict perceives a triumphant neuron i^* , using an unclear framework used by an engaged layer. In any case, as opposed to the neuron count, a particular neighborhood $N_{i^*}(d)$ neuron distance is calculated with the neuron which is evaluated using Kohonen run the show. Based on this, all the neuron values are adjusted by $i \in N_{i^*}(d)$:

$$w_i(q) = w_i(q-1) + \alpha(p(q) - w_i(q-1))$$

or

$$w_i(q) = (1 - \alpha)w_i(q-1) + \alpha p(q)$$

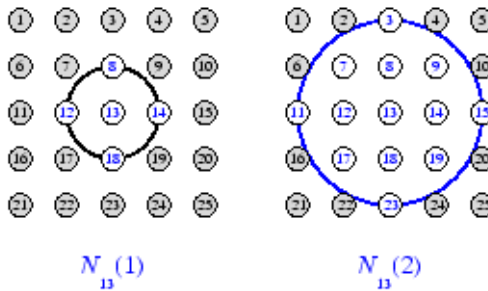
Here the neighborhood $N_{i^*}(d)$ contains the indices for all of the neurons that lie within a radius d of the winning neuron i^* .

$$N_{i^*}(d) = \{j, d_{ij} \leq d\}$$

In this way, the vector is denoted as p and weights are defined in the network. The neurons which lie close to the parent neuron are emerged to p . After a sequence of presentation, the nearby neurons learn through each other.

The SOFM algorithm creates another algorithm, stated as batch algorithm, for presenting a complete record of illumination with restored weights and values. The home neuron picks a data vector at each class where weights are moved to the center position with champ in vector data in the vector region.

To relay on likelihood condition of nearby neuron value, consider the following figure. In the following graph at left side neighborhood, the radius of $d = 1$ in 2-D with 13 neurons are presented whereas appropriate display lies at radius $d = 2$.



The preceding areas are composed as $N_{13}(1) = \{8, 12, 13, 14, 18\}$ and $N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$.

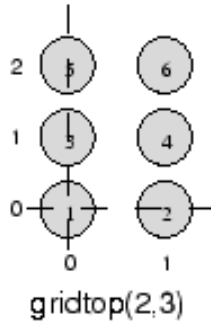
This network starts with rectangular neuron grids as shown in the preceding diagram. For example, expect require a 2×3 display of neuron count 6.

```
>> pos = gridtop([2, 3])
```

```
pos =
```

0	1	0	1	0	1
0	0	1	1	2	2

At this point, the position of neuron 1 lies in (0,0), (1,0) lies in neuron 2, and (0,1) in neuron position 3 and so on.



This requested measurement size turned around, in this we got a marginally extraordinary course of action:

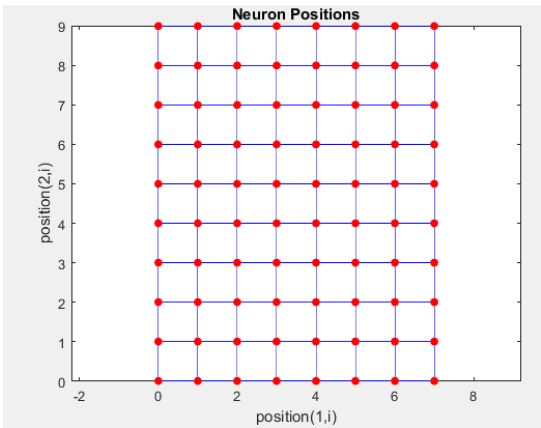
```
>> pos = gridtop([3, 2])

pos =

    0    1    2    0    1    2
    0    0    0    1    1    1
```

From the preceding approach, set the neuron count 8*10 topology with the following function in code:

```
>> pos = gridtop([8 10]);
plotsom(pos)
```



The hextopfunction makes a comparable neuron arrangement, however hexagonal example. A design of 2*3 hextop neurons created taken as follows:

```
>> pos = hextop([2, 3])

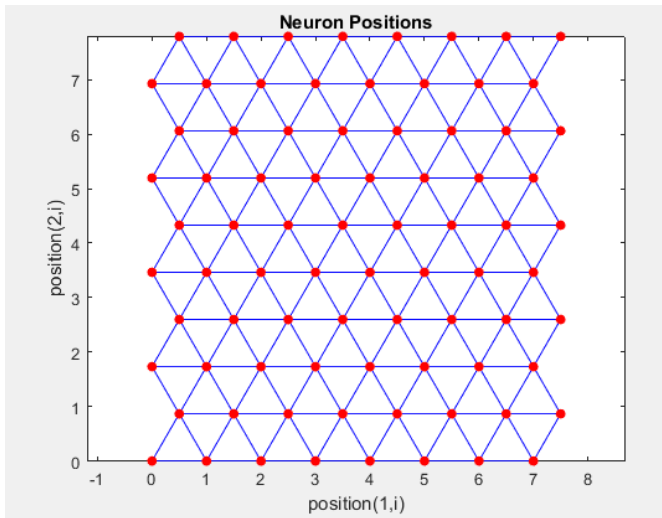
pos =

    0    1.0000    0.5000    1.5000         0    1.0000
    0         0    0.8660    0.8660    1.7321    1.7321
```

Through the hextop function, the neuron set with dimension 8*10 can be plotted in hexagonal topology with following code:

```
>> pos = hextop ([8 10]);
plotsom (pos)

>> pos = hextop([8 10]);
plotsom(pos)
```



Consider that neurons are arranged in a hexagonal shape structure.

At last, the randtop function makes neurons count in the n-dimensional irregular example. The accompanying code is created as an arbitrary example of neurons.

```
>> pos = randtop([2, 3])
```

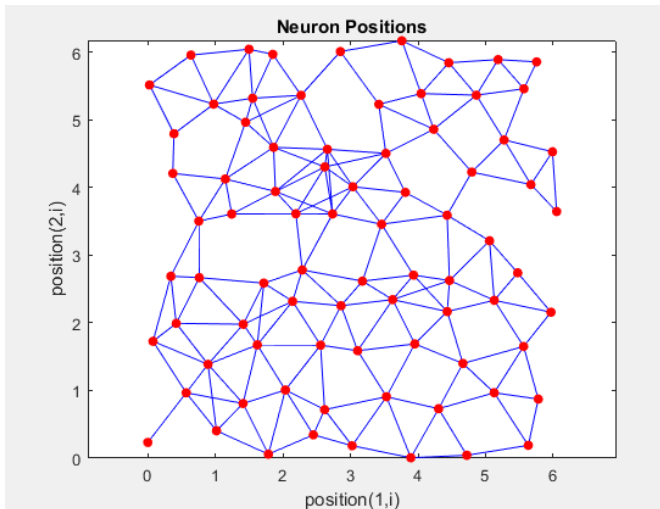
```
pos =
```

```

      0    0.6884    0.0672    1.1448    0.0434    0.7473
0.1338      0    0.9024    0.7198    1.2045    1.3489
```

Further, also with the use of `randtop` network configuration, neurons can be arranged in 8-by-10 structure.

```
>> pos = randtop([8 10]);
plotsom(pos)
```



Separation Functions ()

In order to evaluate the neurons distance from the neighbors, the four functions, **dist**, **linkdist**, **mandist**, and **boxdist** are utilized. Every computation strategy is actualized through unique capacity. The **dist** function calculates Euclidean separations from a home and other neurons.

Assume that there are three neurons:

```
>> pos2 = [0 1 2; 0 1 2]

pos2 =

    0    1    2
    0    1    2
```

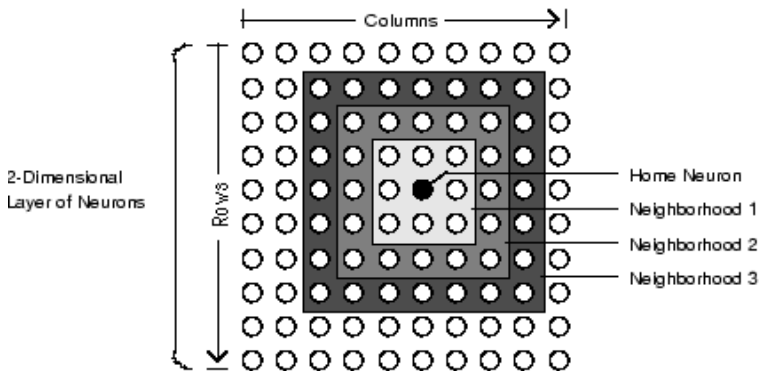
You can now discover the separation from every neuron to the next with

```
>> D2 = dist(pos2)

D2 =

    0    1.4142    2.8284
    1.4142    0    1.4142
    2.8284    1.4142    0
```

The partition of neuron 1 is defined as 0, in a similar manner, neuron 1 is stated as neuron 2 with a value of 1.4142. In underneath the diagram exhibits a neuron in two-dimensional (gridtop) neuron layer. Neighborhood neurons extending separation crosswise over including it. A region of width 1 joins the neuron with fast neighbors. The territory separation crosswise over 2 fuses estimation of 1 neuron and their snappy neighbors.



With respect to the dist function, everyone of areas S-neuron layer delineate spoke to S-by-S matrix separations. Specific separations appeared over (1 in the prompt neighborhood, 2 in neighborhood 2 and so on.) are produced by boxdist function.

```
>> pos = gridtop([2, 3])

pos =

     0     1     0     1     0     1
     0     0     1     1     2     2
```

At that point, the box separations are as follows:

```
>> d = boxdist(pos)

d =

     0     1     1     1     2     2
     1     0     1     1     2     2
     1     1     0     1     1     1
     1     1     1     0     1     1
     2     2     1     1     0     1
     2     2     1     1     1     0
```

Based on the preceding results obtained, it can be concluded that the neurons distance for 1 to 2 and 3 to 4 is 1, since they are nearby neighborhood. In this manner, the neuron distance for 1 to neuron 5 and 6 are 2. Separation between neurons 3 and 4 are 1.

```
>> d = boxdist(pos)

d =

     0     1     1     1     2     2
     1     0     1     1     2     2
     1     1     0     1     1     1
     1     1     1     0     1     1
     2     2     1     1     0     1
     2     2     1     1     1     0
```

Distance link from one neuron to another neuron is defined by connection quality steps which need to be connected with the thought of the neuron. In this manner, on the off chance that you ascertain the separations from a similar arrangement of neurons with linkdist, you get

The Manhattan remove is computed for x and y vector as follows:

```
>> D = sum(abs(x-y))

>> W1 = [1 2; 3 4; 5 6]

W1 =

     1     2
     3     4
     5     6

>> P1 = [1;1]

P1 =

     1
     1

>> Z1 = mandist(W1,P1)

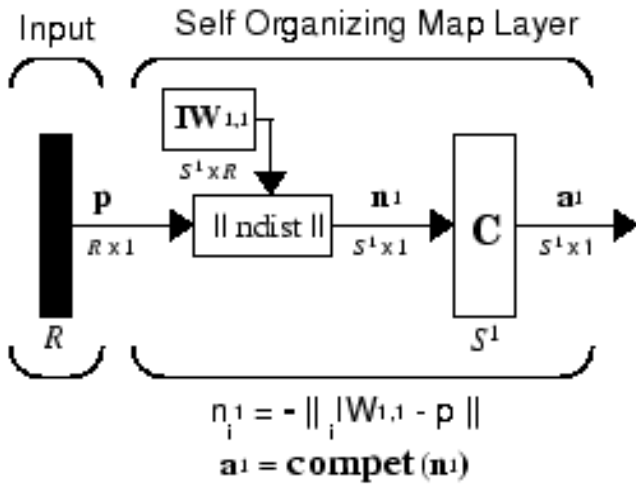
Z1 =

     1
     5
     9
```

The separations ascertained with mandist do to be sure take after the numerical articulation given above

Architecture

The engineering for this SOFM is demonstrated as follows:



This engineering resembles that of an aggressive system apart from no inclination. The focused exchange work creates a^1 to yield element a_i^1 related to i^* . All the remaining components are denoted as 0 by a^1 .

Presently, be that it may, as portrayed previously, neurons near the triumphant neuron are refreshed alongside the triumphant neuron. You can look over different topologies of neurons. Correspondingly, you can browse different separation articulations to figure neurons near the triumphant neuron.

Make a Self-Organizing Map Neural Network (selforgmap)

SOM coordinates with the selforgmap function. This capacity characterizes factors utilized as a part of learning in two periods:

- Learning rate at the stage of ordering
- Steps in ordering stage
- Learning rate of tuning stage
- Neighborhood separation in tuning stage

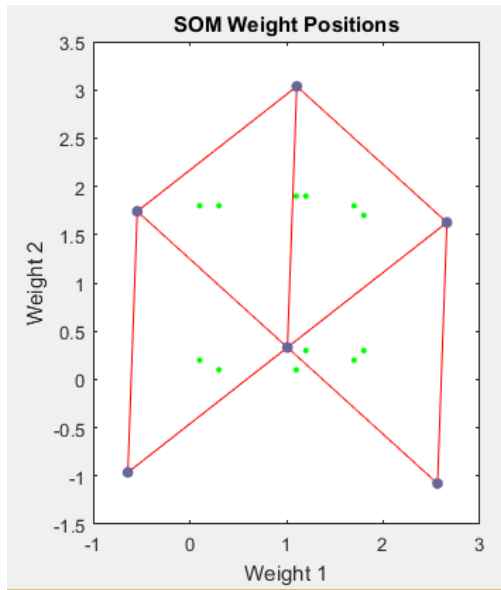
The preceding qualities are utilized in both preparation and adjustment.

Assume to make a system with input vectors and components with six neurons organized in hexagonal shape. Code for this is illustrated here:

Program:

```
>> net = selforgmap([2 3]);
>> P = [.1 .3 1.2 1.1 1.8 1.7 .1 .3 1.2 1.1 1.8 1.7;...
0.2 0.1 0.3 0.1 0.3 0.2 1.8 1.8 1.9 1.9 1.7 1.8];
>> net = configure(net,P);
plotsompos(net,P)
```

Output:



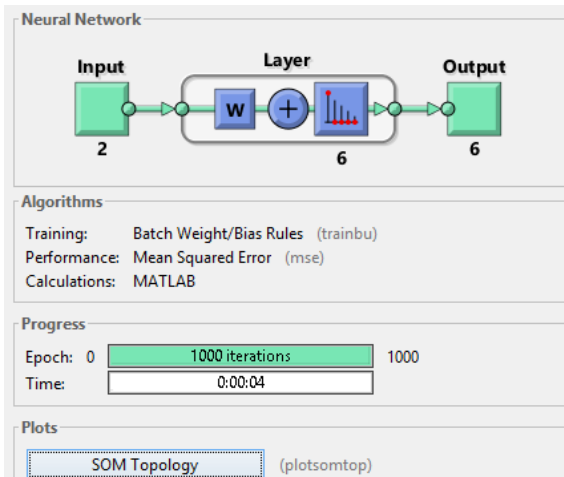
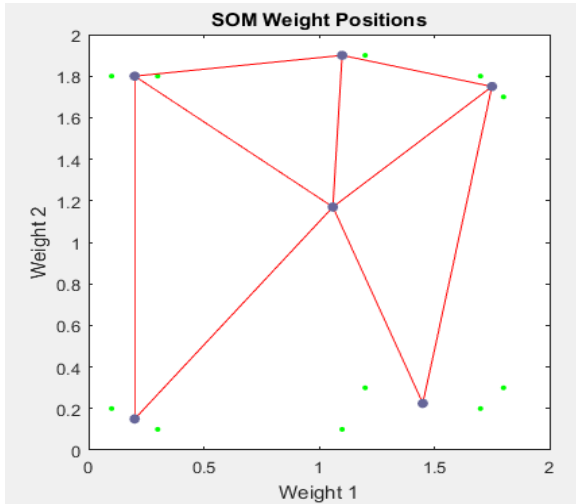
Highlight maps, while figuring out how to classify their information, additionally learn both the topology and the appropriation of their info.

You can prepare the system for 1000 ages with the following code:

Program

```
>>net.trainParam.epochs = 1000;
net = train(net,p) ;
plotsompos(net,p)
```


Output



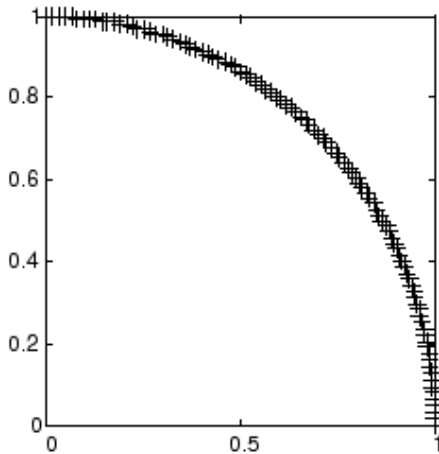
Self-Organizing map in one-dimensional

Consider an input unit vector count of 100 in two-component which spread equally in the vicinity of 0° and 90° .

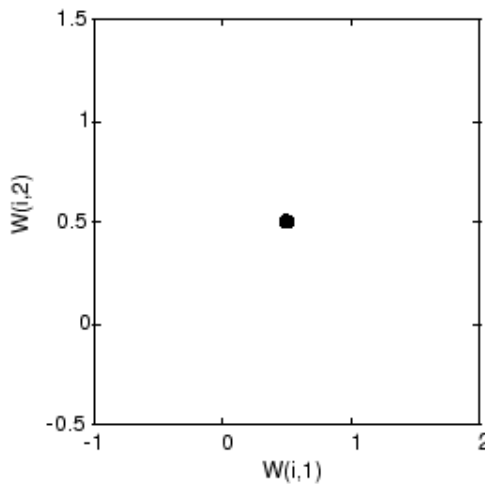
$$\text{angles} = 0:0.5*\pi/99:0.5*\pi;$$

Here is a plot of the information.

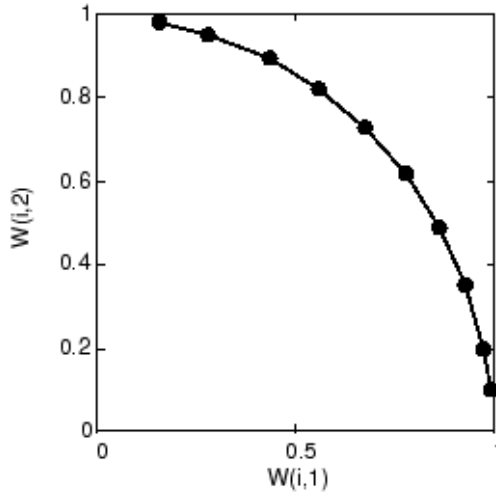
$$P = [\sin(\text{angles}); \cos(\text{angles})];$$



A self-sorting out guide is characterized as 10 neurons count in 1-dimensional layer. Initially, neurons have a target point as illustrated in the following figure:



As preparation begins, weight pushes towards the information vectors. Additionally, ends up requested as the area measure diminishes. At last, the layer changes its weight with the goal that every neuron reacts firmly district information space possessed by input vectors. Neighboring neuron weight vectors additionally mirrors of vector topology.



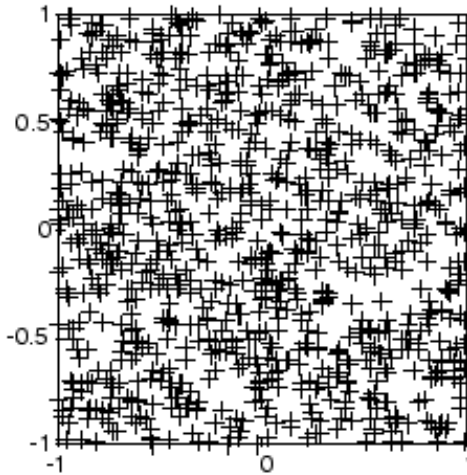
Self-Organizing map in 2D

The preceding illustration demonstrates self-sorting two-dimensional guide preparation.

To begin with, some irregular information is made through the code:

```
P = rands(2,1000);
```

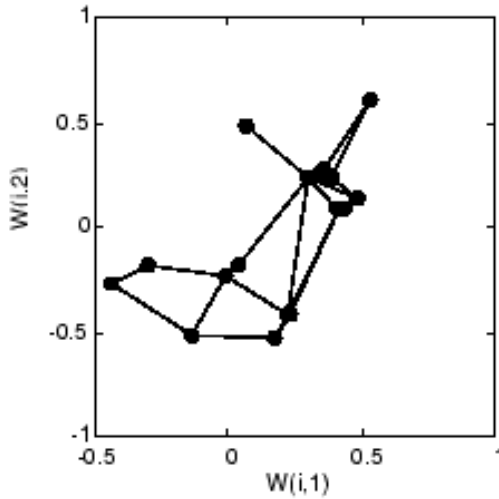
Plot for 1000 information vectors are presented here:



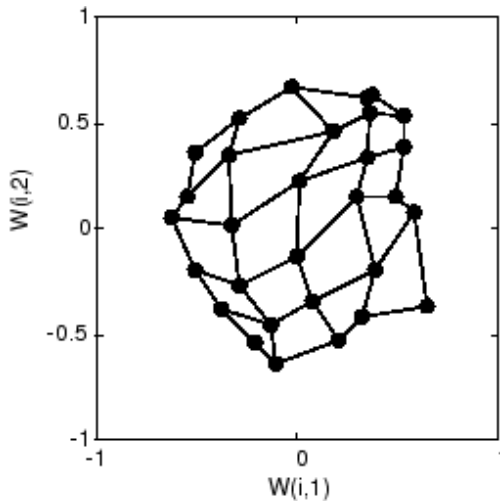
For effective processing of data, 30 neurons are processed with 5-by-6 vector in two-dimensions. In 2-dimension, vector has 5-by-6 neurons with mandist Manhattan neighborhood function.

Based on the 5000 cycles of presentation, the guide is arranged with 20 cycles count.

The simple thing is to look into 40 cycles of operation.

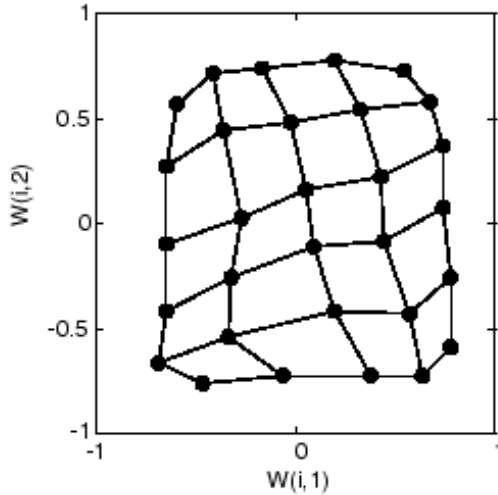


The following cycle presents as a structure after cycle count of 120.

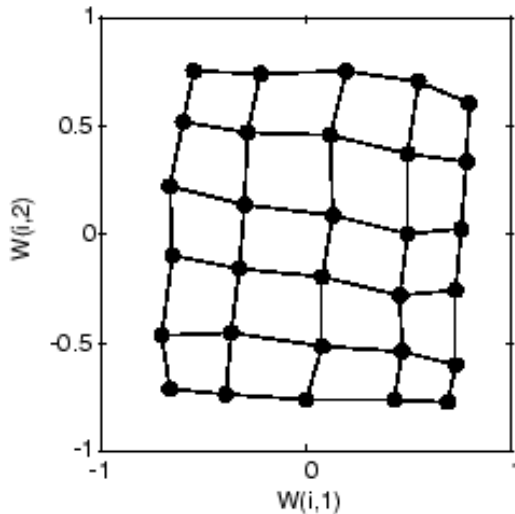


Based on the input vector of data space topology, the entire structure deals itself after 120 cycle count.

The cycle count plot of 500 demonstrates the guide of uniformly appropriated over information space.



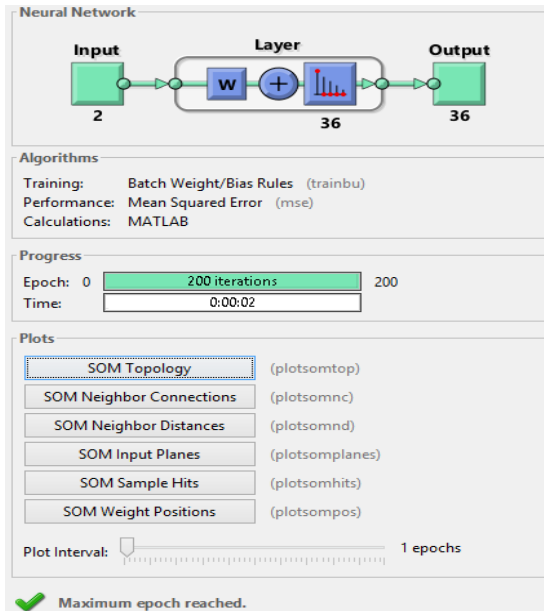
After completion of cycle count of 5000, somewhat uniformly displayed in information space. Likewise, the neurons are equally divided, mirroring the even dispersion of info vectors in this issue.



Preparing with the batch algorithm

The batch preparing calculation is for the most part significantly speedy for calculation which provides default SOMF preparing calculation. Different things of calculation are performed straightforward through informational index for accompanying summons:

```
>> x = simplecluster_dataset
net = selforgmap([6 6]);
net = train(net,x);
```



There are a few helpful perceptions that you can access from this window. In the event that you click Weight Positions of SOM as shown in the preceding figure. The figure also provides vector weight and information value. As presented in the figure, calculation of neuron group is evaluated after 200 count for information space.

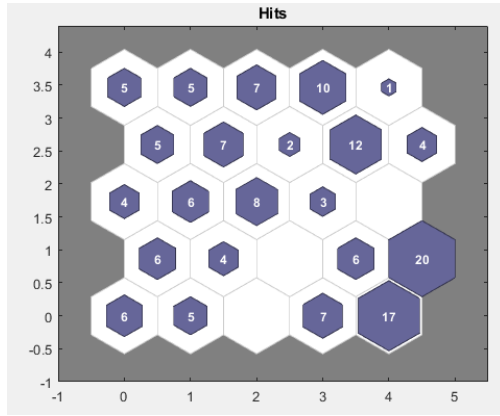
Plot SOM sample hits

Program

```
>> x = iris_dataset;
```

```
net = selforgmap ( [5 5] ) ;
net = train (net,x) ;
plotsomhits (net,x)
```

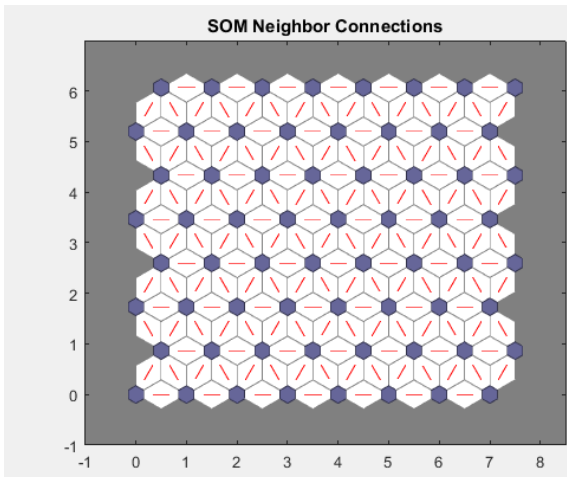
Output



Plot SOM neighbor connections

Program

```
>> x = iris_dataset;
net = selforgmap ( [8 8] ) ;
net = train (net,x) ;
plotsomnc (net)
```

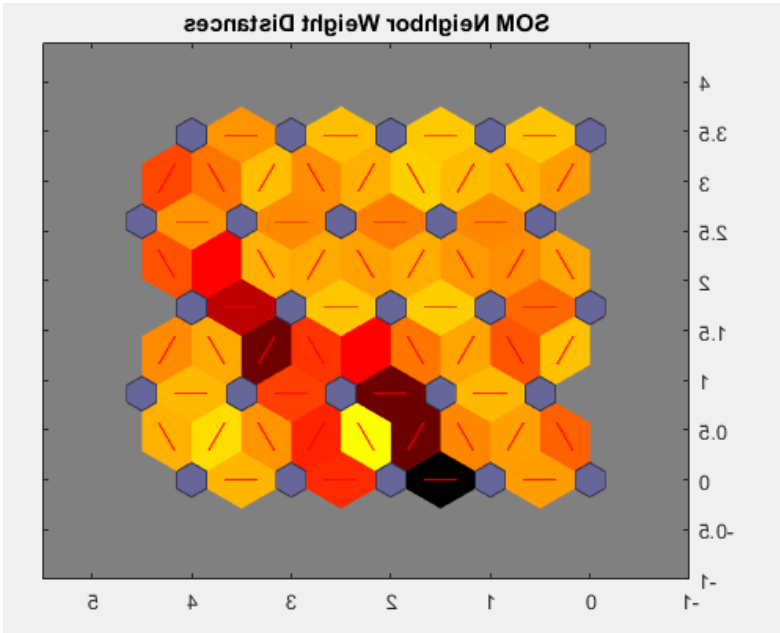


Plot SOM neighbor distances

Program

```
>> x = iris_dataset;  
net = selforgmap ( [5 5] ) ;  
net = train (net,x) ;  
plotsomnc (net)
```

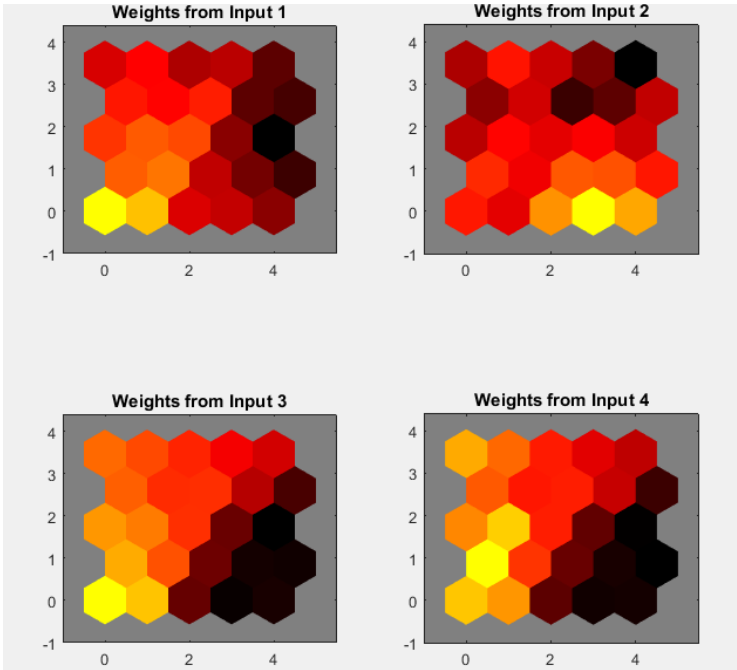
Output



Plot SOM weight planes

```
>> x = iris_dataset;  
net = selforgmap ( [5 5] ) ;  
net = train (net,x) ;  
plotsomplanes (net)
```


Output

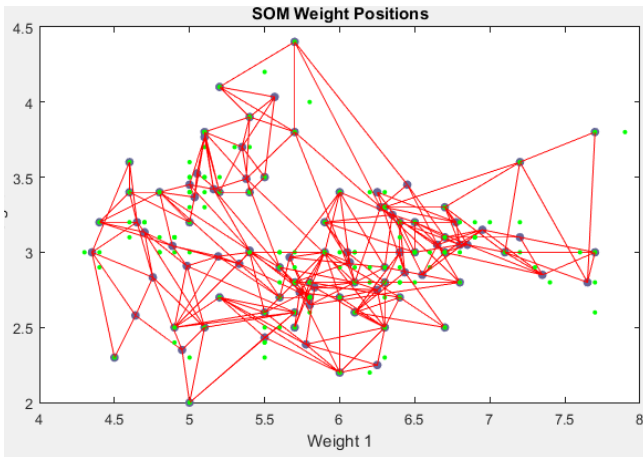


Plot SOM weight positions

Program

```
>> x = iris_dataset;
net = selforgmap ( [10 10] ) ;
net = train (net,x) ;
plotsompos (net,x)
```

Output

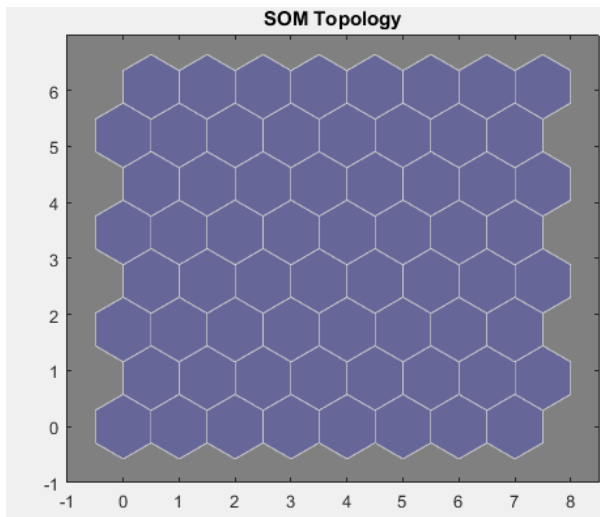


Plot SOM topology

Program

```
>> x = iris_dataset;  
net = selforgmap ( [8 8] ) ;  
plotsomtop (net)
```

Output



6.3 Regression with Feed-forward Networks

Feed forward neural network

Syntax: `feedforwardnet(hiddenSizes,trainFcn)`

Depiction

Feed-forward systems comprise of a progression of layers. The principal layer has an association from the system input. Each ensuing layer has an association from the past layer. The last layer delivers the system's yield.

Feed-forward systems can be utilized for any sort of contribution to yield mapping. A feed-forward coordinate with one shrouded layer and enough neurons in the concealed layers can fit any limited info yield mapping issues.

Specific forms of the feed-forward systems organize incorporate fitting (`fitnet`) and design acknowledgment (`patternnet`) systems. A minor departure from the feed-forward organize is the course forward system (`cascadeforwardnet`) which has extra associations from the contribution to each layer and from each layer to every single after layer.

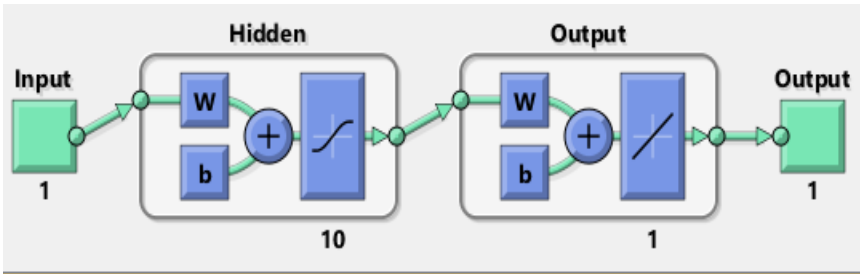
Feed forward neural network

Program

```
>> [x,t] = simplefit_dataset;
net = feedforwardnet (10) ;
net = train(net,x,t) ;
view(net)
y = net (x) ;
perf = perform(net,y,t)
```

```
perf =
    6.5740e-06
```

Output



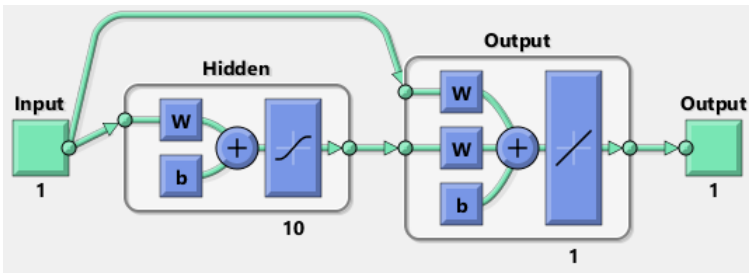
Make and train a cascade network

Program

```
>> [x,t] = simplefit_dataset;
net = cascadeforwardnet (10) ;
net = train(net,x,t) ;
view(net)
y = net (x) ;
perf = perform(net,y,t)
```

```
perf =
    4.3175e-06
```

Output



You can see that the sizes of the information and yield are 1.

Gauge the objectives utilizing the prepared system.

```
>> y = net (x) ;
```

Evaluate the execution of the prepared system. The default execution work is mean squared blunder.

```
perf = perform (net, y, t)
```

```
perf =
1.4639e-04
```

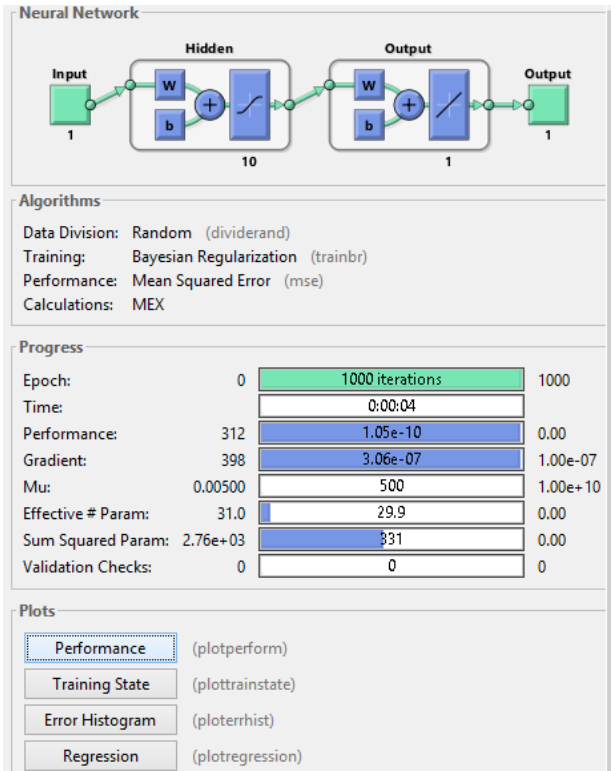
The default preparing calculation for a capacity fitting system is Levenberg-Marquardt ('trainlm'). Utilize the Bayesian regularization preparing calculation and analyze the execution outcomes.

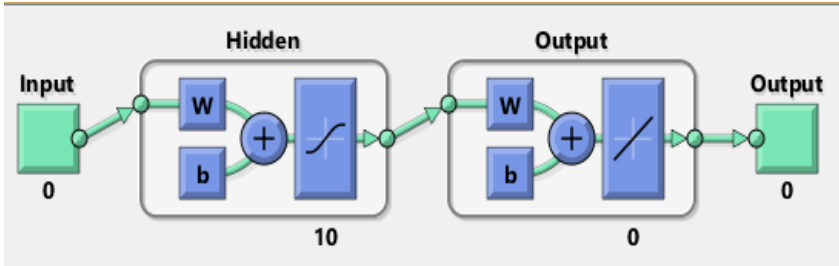
Program

```
>>net = fitnet(10,'trainbr') ;
net = train(net,x,t) ;
y = net (x) ;
perf = perform(net,y,t)
```

```
perf =
2.0386e-10
```

Output





The Bayesian regularization preparing calculation enhances the execution of the system as far as evaluating the objective esteems.

CHAPTER 7

Introduction to Deep Learning

7.1 Deep Learning Overview

Deep learning is a concept of AI which is similar to programming brain neural networks artificially which has the ability of coding itself eventually. Hinton and YannLeCun proposed Deep Learning in 1980's, they chattered about the multilayer neural nets. The unique indication behind AI was to make computers to behave and think smart as a human brain.

7.2 How Deep Learning Works

Deep learning classification mainly works in two stages or steps. The initial process is training and is followed by inferring. Deep learning uses a neural network to emulate animal intelligence. Large dataset is needed to train a neural network. In neural network, it underwent with three types of layers of neurons such as the input layer, the hidden layer and the output layer. Connections between neurons are related with a weight, dictating the importance of the input value. Neurons smear an activation function on the data to “standardize” the output impending out of the neuron. To train a neural network, a large dataset is needed. Iterating through the dataset and comparing the outputs will produce a cost function, indicating how much the AI is off from the real outputs. Gradient descent is used after every iteration for processing the dataset, it is set right to and adjust the weights between neurons to reduce the cost function. Deep learning is based on neural networks but differs from the neural networks and their set of algorithms designed according to the human neural networks and it's programmed to work similar to the human brain.

7.2.1 How is Deep Learning Different from Machine Learning?

Machine learning learns what the programmer teaches it and further makes its own decisions according to what it has learned till now, however, deep

learning structures the algorithms in layers to create a neural network that can make decisions on its own and eventually creates its own codes. It supports both supervised and unsupervised learning tasks, but machine learning at this stage is not as successful as deep learning while dealing with unsupervised learning.

7.2.2 Is Deep Learning Different from AI (artificial intelligence)?

Deep learning is a part of artificial intelligence which deals with total neural concepts and to complete more complex problems with ease, deep learning as I further discussed above it is simulating a brain artificially through programming so it can perform tasks very accurately and efficiently.

7.2.3 What is Deep Learning Framework?

The deep learning computing framework that supports deep learning to a great extent and is widely used by many companies like Facebook, Google, Twitter, and so on, is Lua.

7.2.4 What are the Dimensions of the Deep Learning?

Deep learning use two types of dimensions: vectors and matrices. These two dimensions are commonly called as Tensors in deep learning where vectors are one-dimensional and matrices are two-dimensional these are present in both machine learning—which is going to be the smart brain in future—and deep learning.

7.3 Deep Learning uses and Functioning

Deep learning is focused to imitate the human brain process of thinking, processing data, and creating patterns to use them in decision making. It is designed artificially just like the human brain.

7.4 Programming Languages used to Program (design) Deep Learning?

According to many surveys and researches, the most used AI developing programming languages are Python, R, Lisp, Prolog, Java, and so on. In these languages, Python is most widely used in many fields of development of deep learning. Based on many algorithms and techniques, deep learning

is programmed in the programming languages. In Python programming language, numpy, pybrain, and so on are used.

7.5 Meaning and importance of Deep Learning

Deep learning permits procedure models that are composed of multiple process layers to find out representations of information with multiple levels of abstraction. These ways have dramatically improved the progression in speech recognition, visual perception, object detection, and plenty of alternative domains like drug discovery and genetics. Deep learning discovers tortuous structure in giant knowledge sets by exploiting the back-propagation formula to point. However, a machine ought to modify its internal parameters that are accustomed cypher the illustration in every layer from the illustration within the previous layer. Deep neural network plays a conventional role in different fields especially audio, speech, and video. For text and speech process lightweight sequential knowledge is performed.

Big data and deep learning are two high-focal points of data science. Gigantic information has turned out to be fundamental as a few associations every open and individual are gathering a lot of area particular data, which may contain supportive data with respect to issues like national insight, digital security, extortion recognition, advertising, and restorative logical train. Partnerships like Google and Microsoft are examining mammoth volumes of data for business examination and choices, affecting existing and future innovation. Profound learning calculations extricate abnormal state, confused deliberations as information portrayals through a stratified learning technique. Convoluted deliberations are learnt at a given level bolstered similarly less muddled reflections created inside the former level inside the chain of command. A key preferred standpoint of deep learning is the examination and learning of a lot of unattended information, making it a significant instrument for mammoth learning analytic wherever information is basically unlabeled and unordered. Inside the blessing study, we have a tendency to investigate, however, deep learning will be used for some important issues in huge information analytics, and also removing confused examples from substantial volumes of data, semantics categorization, learning labels, speedy data recovery, and streamlining discriminate undertakings. We have a tendency to furthermore research a few parts of deep learning investigation that requires more investigation to incorporate particular difficulties presented by gigantic information

analytics, and additionally spilling information, high-dimensional information, quantifiable models, and conveyed figuring. We have a tendency to finish up by showing experiences into significant future works by motility a few inquiries, and also process information examining criteria, space adjustment displaying, process criteria for getting supportive learning reflections, rising etymology categorization, semi-regulated learning, and dynamic learning.

“Deep learning could be an explicitly reasonable machine learning technique that achieves land and suppleness by learning to represent the planet as a nested hierarchy of ideas, with every conception outlined in reference to less complicated ideas, and a lot of abstract representations computed in terms of less abstract ones.”

7.6 What Deep Learning can do in Future?

Deep learning is a giant which can make computer as intelligent as humans or even more. In the future, many applications are getting developed and been developed day-by-day and released in the market and are being to be loved by the human society due to its accuracy and efficiency towards it's work and it's dedication towards the work and it becomes perfect day by day as it performs the work and perform it even better. In present days, automatic game playing Image Capture Generation , Character Text Generation, Automatic Image Colourizing, and many more.

Some of the deep learning architectures are deep neural networks and recurrent natural networks have been applied to various fields including computer hallucination, speech and audio recognition, natural language processing which forms a bridge between communication of human and brain. Under the massive data collection, it takes responsibility for filtering social network, machine translation, bioinformatics, drug design, and medical image analysis, where it result in new research progression in medical and biological image analysis, material inspection, and board game programs which runs in all platforms. They have produced results similar to and in some cases it is even grander to human professionals by executing the same process multiple times again and again it become perfect in that task and it will not stop doing the process till it bypasses its own perfection in that field of work.

7.7 Applications of Deep Learning in Artificial Intelligence

1. Computer vision
 - a. Image classification
 - b. Object detection
2. Speech and audio
 - a. Voice recognition
 - b. Language translation
3. Natural language processing
 - a. Recommendation engines
 - b. Sentiment analysis

7.8 Fields were deep learning boom:

- Medical image exploration
- Material assessment
- Bio informatics
- Drug enterprise
- Speech and audio recognition
- Board game platforms
- Computer vision
- Machine translation
- Filtering of social networking

7.9 The future of deep learning

- Deep learning process will assemble on a fundamental set of de-facto tool structures. More open source will change the dealing out of computing world such as:
 - Tensor flow
 - Big DL
 - Open Deep
 - Theano

- Torch
- MXnet
- Spark is going to be an essential platform for scaling and accelerating deep learning algorithms built in various tools.
- Simplified programming code is claimed on APIs and other programming abstractions for fast coding of the core algorithmic capabilities with fewer lines of code. Much easier for graphic enterprise, modeling, and designing of different models from the prevailing pre-models.
- It will embed with every design surface.

7.10 Algorithms in Deep Learning

Few algorithms are going to play a vital role in deep learning progression.

1. Linear recession
2. Flood fill algorithm
3. Floyd's cycle detection algorithm
4. Greedy algorithm
5. Linear discriminant exploration
6. Regression and classification of trees
7. Navie Bayes
8. Learning route quantization
9. Support vector machines
10. Capturing and random forest
11. Boosting and Ada boost
12. Logistic recession
13. K-nearest neighbors

Machine learning challenges

Machine learning, which facilitates predictive analytics using large volumes of data by employing algorithms that iteratively learn from that data, is one of the fastest growing areas of computer science. It uses a range from the credit card fraud detection and self-driving cars to optical character recognition (OCR) and online shopping recommendations.

It makes us smarter by making computers smarter. Its usefulness will only increase as more and more data becomes available and the desire to perform predictive analytics from that data grows, too.

Bayesian deep learning

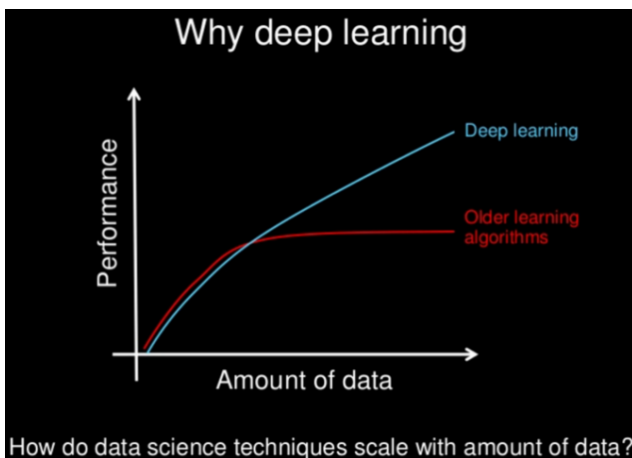
They tend to care about uncertainty, because of element-wise dropout. An input image is taken and predicted to the depth and perform model uncertainty. Possible solution is AutoML which form a new network with:

- Number of filters
- Filter height and width
- Stride height and width

7.11 Comparison of Machine Learning and Deep Learning

7.11.1 Data Dependencies

The most necessary distinction between deep learning and ancient machine learning is its performance because the scale of information will increase. If the information is tiny, deep learning algorithms don't perform that well. This can be because the deep learning algorithms might like an oversized quantity of information to grasp it dead. On the other hand, ancient machine learning algorithms with their handcrafted rules prevail during this situation. The following image summarizes this reality.



7.11.2 Hardware dependencies

Deep learning algorithms relies on higher order machine learning approaches which also works on low-end machine learning approaches. This can be a result of the necessities of deep learning formula embody GPUs that are an integral part of its operating. Deep learning algorithms inherently do an oversized quantity of matrix operations. These operations will be expeditiously optimized employing a GPU as a result, the GPU is constructed for this purpose.

7.11.3 Execution time

Usually, a deep learning formula takes an extended time to coach. This can be as a result of having a lot of parameters in an exceedingly deep learning formula such that coaching them takes longer than usual. State of the art deep learning formula ResNet takes a fortnight to coach fully from scratch. Whereas, machine learning relatively takes very less time to coach, starting from some seconds to some hours. This flip is totally reversed on testing time. At check time, deep learning formula takes very less time to run. This can be comparatively examined with k-nearest neighbors which is also a class of machine learning algorithm, check time will increase on increasing the scale of information. This can be not applicable on all machine learning algorithms, as a number of them have tiny testing times too.

7.11.4 Interpretability

Last, however, not the smallest amount, we've interpretability as an element for comparison of machine learning and deep learning. This issue is the main reason deep learning is thought of ten times before its use in a business.

Let's take an example, suppose we tend to use deep learning to grant machine-driven evaluation to essays. The performance it offers in evaluation is sort of wonderful and is close to human performance. However, there's is a problem. It doesn't reveal why it's provided that score. So mathematically, you'll establish that nodes of a deep neural network were activated, however, we tend to not grasp what there neurons were alleged to model and what these layers of neurons were doing jointly. Thus, we tend to fail to interpret the results.

On the other hand, machine learning algorithms like call trees provide America crisp rules on why it selected, what it selected, thus, it's significantly simple to interpret the reasoning behind it. Therefore, algorithms like call trees and linear/logistic regression are primarily employed in businesses for interpret ability.

7.12 TensorFlow

Introduction to the Python deep learning library TensorFlow

TensorFlow could be a Python library for fast numerical figuring made freely available by Google. The TensorFlow library is used for deep learning.

7.12.1 What is TensorFlow

TensorFlow is an open supply library for brisk numerical computing. It was made and is kept up by Google and free underneath the Apache a couple of open supply permit. The API is ostensibly for the Python simulated dialect, however, there's entrance to the fundamental C++ API. Unlike elective numerical libraries intended to be utilized as a part of deep learning like Theano, TensorFlow was intended to be utilized each in investigation, advancement, and underway frameworks. It will keep running on single processor frameworks, GPUs still as cell phones and tremendous scale conveyed frameworks of numerous machines.

7.12.2 Steps to install TensorFlow

Download TensorFlow from the link https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html, transfer and setup instructions are available on the TensorFlow website.

First examples in TensorFlow

Computation is delineate in terms of information flow and operations within the structure of a directed graph.

Nodes: Hubs perform calculations and have zero or a great deal of information sources and yields. Information that moves between hubs are alluded to as tensors, that are multi-dimensional varieties of genuine esteems.

Edges: The diagram characterizes the stream of data, expanding, processes, and updates to state. Uncommon edges will be acclimated synchronize

conduct inside the diagram, for example, expecting calculation on assortment of contributions to finish.

Operation: An activity could be a named unique calculation which may include characteristics and turn out yield qualities. For example, you'll diagram an include or duplicate task.

Computation with TensorFlow

```
import tensorflow as tf
sess = tf.Session()
a = tf.constant(10)
b = tf.constant(32)
print(sess.run(a+b))
```

42

7.12.3 Linear Regression with TensorFlow

TensorFlow could be a tool for machine learning, whereas, it contains a good vary of practicality. TensorFlow is especially designed for deep neural network models.

These cases appear anyway you'll diagram factors (e.g. W b) still as factors that are the consequences of calculation (y).

In TensorFlow, we have a separate declaration and definition for calculating computation and execution session to call run.

```
import tensorflow as tf
import numpy as np

# Create 100 phony x, y data points in NumPy,  $y = x * 0.1 + 0.3$ 
x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute  $y\_data = W * x\_data + b$ 
# (We know that W should be 0.1 and b 0.3, but TensorFlow will
# figure that out for us.)
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the variables. We will 'run' this first.
init = tf.initialize_all_variables()

# Launch the graph.
sess = tf.Session()
sess.run(init)
```



```

# Fit the line.
for step in range (201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

# Learns best fit is W: [0.1], b: [0.3]

0 [0.3019389] [0.26119673]
20 [0.14882548] [0.27409264]
40 [0.11372912] [0.2927152]
60 [0.10386045] [0.2979516]
80 [0.10108552] [0.29942402]
100 [0.10030525] [0.29983804]
120 [0.10008585] [0.29995447]
140 [0.10002415] [0.2999872]
160 [0.1000068] [0.2999964]
180 [0.10000192] [0.299999]
200 [0.10000055] [0.2999997]

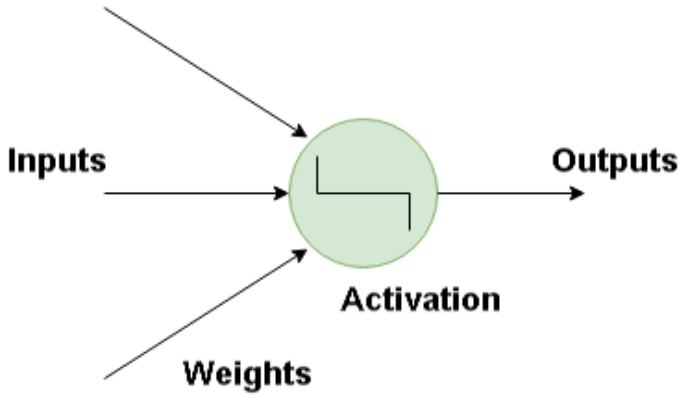
```

7.13 Artificial Neural Networks

The main reason behind deep learning is the concept that computing ought to draw inspiration from the brain. This angle gave rise to the “neural network” language. In human brain, tens and billions of neurons are interconnected among them. Deep learning algorithms gibe the brain in several conditions, as the brain and deep learning models involve a huge range of computation units (neurons) that aren’t very intelligent in isolation. However, they become intelligent after they move with one another.

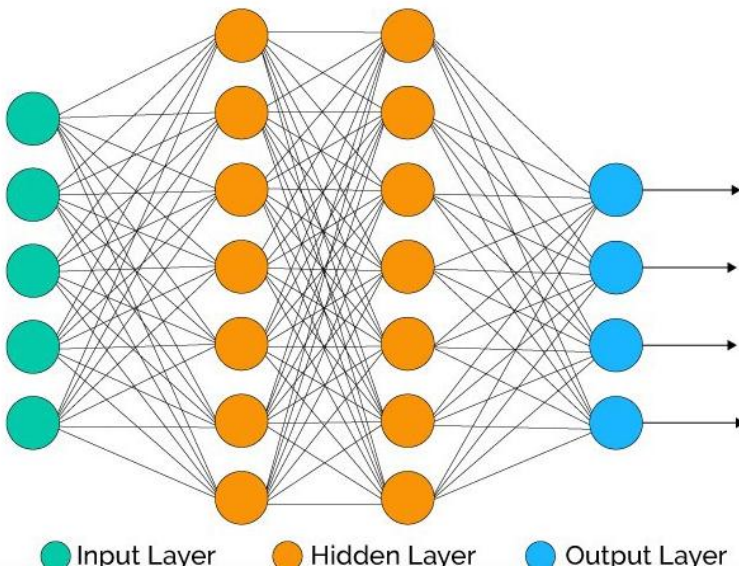
7.13.1 Neurons

The basic building block for neural networks are artificial neurons, that imitate human brain neurons. These are straightforward, powerful procedure units that have weighted input signals are turn out to signal exploitation an activation perform. These neurons are unfolded across the many layers within the neural network.



7.13.2 How will Artificial Neural Network Work?

Deep learning consists of artificial neural networks that are modeled on similar network gifts within the human brain. As knowledge travels through this artificial mesh, every layer processes a side of the information, filters outlines, spots acquainted entities, and produces the ultimate output.



Input layer: This layer consists of the neurons that do nothing than receiving the inputs and pass it on to the opposite layers. The quantity of layers within the input layer ought to be capable of the attributes or options within the datasets.

Output layer: The output layer is the expected feature, it essentially depends within the form of model you're building.

Hidden layer: In between input and output layer be hidden layers supported the sort of model. Hidden layers contain huge range of neurons. The neurons within the hidden layers apply transformations to the inputs before passing them because the network is trained, the weights get updated to be a lot of prophetical.

7.13.3 Neuron Weights

Weights check with the strength or amplitude of an association between two neurons, if you're acquainted with simple regression, you'll compare we tend on inputs like coefficients we use in an exceedingly equation. Weights are typically initialized to tiny random values, like values within the vary zero to one.

7.13.4 Feed-forward Deep Networks

Feed-forward supervised neural networks were among the primary and most triple-crown learning algorithms. They're additionally referred to as deep networks, multi-layer perceptron (MLP), or just neural networks and therefore the vanilla design with one hidden layer is illustrated. Every vegetative cell is related to alternative vegetative cell with some weight

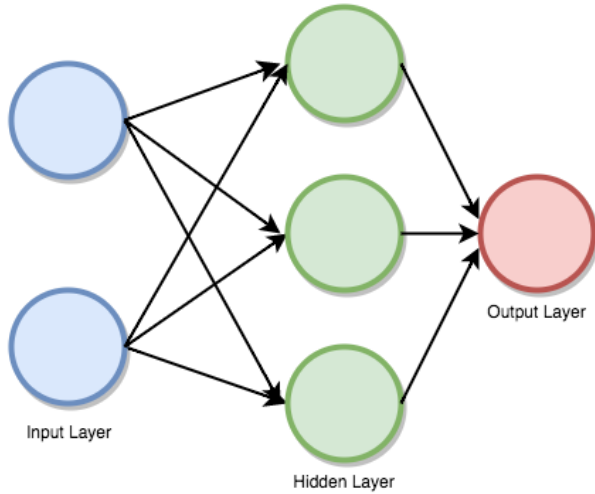
Neuron weights

Weights check with the strength or amplitude of an association between two neurons, if you're acquainted with simple regression you'll compare that we tend on inputs like the coefficients we use in an exceedingly equation. Weights are typically initialized to tiny random values, like values within the vary zero to one.

7.13.5 Feed-forward Deep Networks

Feed-forward supervised neural networks were among the primary and most triple-crown learning algorithms. They're additionally referred to

as deep networks, multi-layer Perceptron (MLP), or just neural networks and therefore the vanilla design with one hidden layer is illustrated. every vegetative cell is related to alternative vegetative cell with some weight,



7.14 Activation function








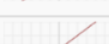
An activation perform could be a mapping of summed weighted input to the output of the vegetative cell. It's referred to as an activation/transfer perform as a result of it governs the beginning at that the vegetative cell is activated and therefore the strength of the signal.

Mathematically, it is defined as:

$$Y = \Sigma(\text{weight} * \text{input}) + \text{bias}$$

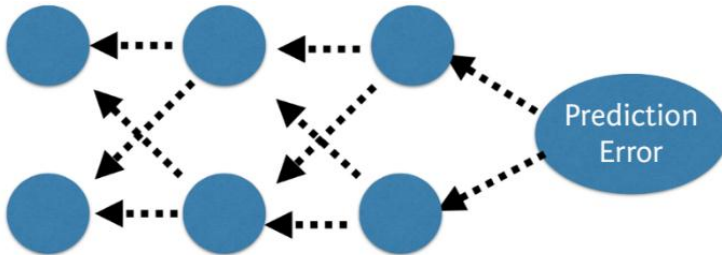
Among several activation functions, we mostly utilize relu, tanh, and solfPlus.

The cheat sheet for activation functions in given here:

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

7.14.1 Back propagation

The predicted worth of the network is compared to the expected output, and miscalculation is calculated employing a perform. This error is then propagated back inside the total network, one layer at a time, and therefore, the weights are updated in line with the worth that they contributed to the error. This clever little bit of mathematics is termed as the back-propagation formula. The method is perennial for all of the examples in your coaching knowledge. One spherical of change the network for the whole coaching data sets is termed an epoch. A network could also be trained for tens, tons of or several thousands of epochs.



7.14.2 Cost Perform and Gradient Descent

The cost perform is that the live of “how good” a neural network did for it’s given coaching input and therefore the expected output. It additionally could rely upon attributes like weights and biases.

A cost perform is single-valued, not a vector as a result of it rates however well the neural network performed as a full. Exploiting the gradient descent improvement formula, the weights are updated incrementally once every epoch.

Compatible value function:

Mathematically,

Sum of square errors (SSE)

$$J(w) = \frac{1}{2} \sum_i ((\text{target})^i - (\text{output})^i)^2$$

The magnitude and direction of the burden update is computed by taking a step within the wrong way of the price gradient.

$$\Delta w_j = -\eta \frac{\delta J}{\delta w_j}$$

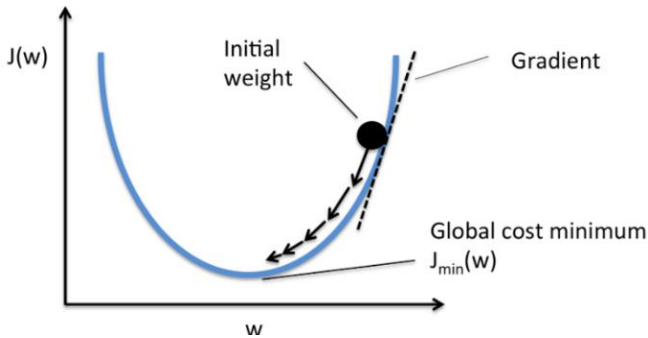
where η is the learning rate.

Where, Δw is a vector that contains the burden updates of every weight constant w , that is computed as follows:

$$\Delta w_j = \frac{1}{2} \sum_i \left((\text{target})^i - (\text{output})^i \right) x_j^{(i)}$$

Graphically, considering value perform with single constant:

We calculate the gradient descent till the by-product reaches the minimum error, and every step is decided by the preciousness of the slope (gradient).



7.15 Multi-layer perceptron (forward propagation)

This category of networks consists of multiple layers of neurons, typically interconnected in an exceedingly feed-forward means (moving in an exceedingly forward direction). every vegetative cell in one layer has direct connections to the neurons of the next layer. In several applications, the units of those networks apply a sigmoid or ReLA (Rectified Linear Activation) perform as an activation perform.

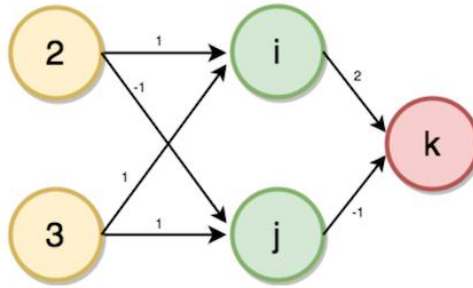
Now, contemplate a drag to search out the quantity of transactions, given accounts, and members of the family as input. To solve this first, we'd like to begin with making a forward propagation neural network. Our input layers are the quantity of members of the family and accounts, the quantity hidden layers are one, and therefore the output layers are the quantity of transactions.

Given weights as shown within the figure from the input layer to hidden layer with number of members of the family a pair of and number of accounts three as inputs.

Now the values of hidden layer (i, j) and output layer (k) are calculated through exploitation of forward propagation by the subsequent steps.

Process

- Multiply—add method.
- Dot product (inputs * weights).
- Forward propagation for one information at a time.
- Output is that the prediction for that information.



Value of i will be able to be calculated from the input worth and therefore, the weights are comparable to the vegetative cell connected.

$$i = (2 * 1) + (3 * 1)$$

$$\rightarrow i = 5$$

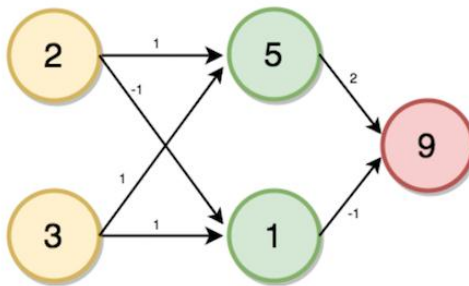
Similarly,

$$j = (2 * -1) + (3 * 1)$$

$$\rightarrow j = 1$$

$$K = (5 * 2) + (1 * -1)$$

$$\rightarrow k = 9$$



Solving the multi-layer perceptron downside in Python:

```
import numpy as np
print("Enter the two values for input layers")
print('a = ')
a = int (input( ) )
# 2
print('b = ')
b = int(input ( ) )
# 3
input_data = np.array( [a, b] )
```



```

weights = {
    'node_0': np.array( [1, 1] ),
    'node_1': np.array( [-1, 1] ),
    'output_node': np.array( [2,-1] )
}
node_0_value = (input_data * weights['node_0']
).sum( )
# 2 * 1 +3 * 1 = 5
print('node 0_hidden: ( )'.format(node_1_value) )
hidden_layer_values = np.array( [node_0_value, node_1_
value] )
output_layer = (hidden_layer_values * weights['output_
node'] ).sum( )
print("output layer : { }".format (output_layer) )

Enter the two values for input layers
a =
3
b =
4
node_0_hidden: 7
node_1_hidden: 1
output_layer : 13

```

7.16 Using Activation Perform

For neural network to attain their most prophetic power we'd like to use AN activation perform for the hidden layers. It is accustomed to capture the non-linearities.

For corrected linear activation function (ReLU):

$$RELU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

```

import numpy as np
print ("Enter the two values for input layers")
print ('a = ')
a = int (input( ) )
# 2
print('b = ')
b = int (input ( ) )

```

```

weights = {
    'node_0': np.array( [2,4] ) ,
    'node_1': np.array( [ [4, -5] ] ) ,
    'output_node': np.array( [2, 7] )
}
input_data = np.array ( [a, b] )
def relu(input) :
    # Rectified Linear Activation
    output = max(input, 0)
    return(output)

node_0_input = (input_data * weights['node_0'] ) .sum(
)
node_0_output = relu (node_0_input)
node_1_input = (input_data * weights['node_1'] ) .sum(
)
node_1_output = relu (node_1_input)
hidden_layer_outputs = np.array( [node_0_output,
node_1_output] )
model_output = (hidden_layer_outputs *
weights['output_node'] ) .sum ( )
print (model output)

```

Enter the two values for input layers

a =

3

b =

4

44