414458: COMPUTER LABORATORY – VII

PART B : Machine Learning Applcation

B.E. (Information Technology) 2015 Course
Semester I

Teaching Scheme		Examination Scheme	
Practical	**: 4 Hrs/Week**	**Term Work**	**: 50 Marks**
Theory	**: 3 Hr/Week**	**Practical**	**: 50 Marks**

DEPARTMENT OF INFORMATION TECHNOLOGY
2020-2021

INDEX

SCHEDULE

Sr. No.	Title	No. of	Week

		Hrs.	
1	**Study of platform for Implementation of Assignments** Download the open source software of your interest. Document the distinct features and functionality of the software platform. You may choose WEKA or R or Rjava.	2	1
2	**Supervised Learning - Regression** Generate a proper 2-D data set of N points. Split the data set into Training Data set and Test Data set. i) Perform linear regression analysis with Least Squares Method. ii) Plot the graphs for Training MSE and Test MSE and comment on Curve Fitting and Generalization Error. iii) Verify the Effect of Data Set Size and Bias-Variance Tradeoff. iv) Apply Cross Validation and plot the graphs for errors. v) Apply Subset Selection Method and plot the graphs for errors. vi) Describe your findings in each case.	4	2,3
3	**Supervised Learning - Classification** Implement Naïve Bayes Classifier and K-Nearest Neighbor Classifier on Data set of your choice. Test and Compare for Accuracy and Precision.	2	4
4	**Unsupervised Learning** Implement K-Means Clustering and Hierarchical clustering on proper data set of your choice. Compare their Convergence.	2	5
5	**Dimensionality Reduction** Principal Component Analysis-Finding Principal Components, Variance and Standard Deviation calculations of principal components.	2	6
6	**Supervised Learning and Kernel Methods** Design, Implement SVM for classification with proper data set of your choice. Comment on Design and Implementation for Linearly non separable Dataset.	2	7

Assignment 1 Study of open source analytical software

AIM : Study of platform for Implementation of Assignments

Download the open source software of your interest. Document the distinct features and functionality of the software platform. You may choose WEKA or R or Rjava.

OBJECTIVE :

To study

- Concept of open source analytical software. (WEKA & R)
- concept of statistical analysis.
- distinct features and functionality of open source software.
- Open source tools and verify execution of programs on different inputs dataset

THEORY:

- **Introduction of WEKA**

Weka is open source software under the GNU General Public License. System is developed at the University of Waikato in New Zealand. "Weka" stands for the Waikato Environment for Knowledge Analysis. The software is freely available at http://www.cs.waikato.ac.nz/ml/weka. The system is written using object oriented language Java. There are several different levels at which Weka can be used. Weka provides implementations of state-of-the-art data mining and machine learning algorithms. Weka contains modules for data preprocessing, classification, clustering and association rule extraction.

- **Introduction of R**

R is a programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years. R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. S was created by John Chambers while at Bell Labs. There are some important differences, but much of the code written for S runs unaltered. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the *R Development Core Team*, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S.R is a GNU project. The source code for the R software environment is written primarily in C, Fortran, and R.R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. R uses a command line interface; there are also several graphical front-ends for it.

Concept/Working:

- **Steps to download and configure the WEKA**

Download Weka (the stable version) from http://www.cs.waikato.ac.nz/ml/weka/

– Choose a self-extracting executable (including Java VM)

– (If you are interested in modifying/extending weka there is a developer version that includes the source code)

After download is completed, run the self extracting file to install Weka, and use the default set-ups.

- **Working of WEKA**

The general working steps are given below by considering the example of Hierarchical clustering.

1. Select a dataset for example iris.arff.
2. Select option Cluster
3. Choose cluster type: Hierarchial Cluster
4. Select cluster mode: Training Set.
5. Click on Start.

- **Features of WEKA**

Main features of Weka include:

- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 15 attribute/subset evaluators + 10 search algorithms for feature selection.
- 3 algorithms for finding association rules
- 3 graphical user interfaces

– "The Explorer" (exploratory data analysis)

– "The Experimenter" (experimental environment)

– "The KnowledgeFlow" (new process model inspired interface)

- **Steps to download and configure the R**

1. Install r-base : Write this command in Command Prompt : sudo apt-get install r-base

2. Type R on terminal/Command line to get command line for R programming.

Using IDE:

1. URL for Rstudio :

   http://www.rstudio.com/products/rstudio/download/
   Write this command in Command Prompt: sudo apt-get install r-base
   OR
2. Open Ubuntu Software Center

   Search : R-studio
   Install : R-Studio

- **Working of R**

The default panes:

- Console (entire left)
- Workspace/History (tabbed in upper right)
- Files/Plots/Packages/Help (tabbed in lower right)

1. Download package needed for that program.
2. Open RStudio.
3. Install the Required Package. Go to Packages - Click on "Install".
   You will get "Install Packages" Window.
   Install from: Package Archive File(tar.gz)
   Package archive: (Browse the path where you have stored the Package_name.tar.gz)
   Click Install.
4. In Packages - Tick the Package_name in user library. Click on Update. Click on OK.
5. Write Following Code in R Script (File - New File - R Script) and Save it as "Program1.R"

```
library(e1071)
data(iris)
set.seed(123)
cm1 <- cmeans(iris[,1:4],10)
bc1 <-
bclust(iris[,1:4],3,base.centers=20,iter.base=50,base.method="cmeans")
```

6. Click on run Icon.
7. Write following commands on 'console' which is in 'RStudio'.

```
> data("iris")
> library(class)
```

```
> library(e1071)
> pairs(iris[1:4], main = "Iris
Data(red=setosa,green=versicolor,blue=virginica)",pch= 21, bg =
c("red","green3","blue")[unclass  (iris$Species)])
> data(iris)
> summary(iris)
> classifier<-naiveBayes(iris[,1:4], iris[,5])
> table(predict(classifier, iris[,-5]), iris[,5])
```

- **Features of R**

Statistical Features

- Implement a wide variety of statistical and graphical techniques.
- R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages.
- For computationally intensive tasks, C, C++, and Fortran code can be linked and called at run time.

Programming Features

- R is an interpreted language.
- R's data structures include vectors, matrices, arrays, data frames (similar to tables in a relational database) and lists.
- R supports procedural programming with functions and, for some functions, object-oriented programming with generic functions.

Advantages/Disadvantages:

- **Advantages of WEKA.**

Free availability:

- Under the GNU General Public License
- Portability
- Fully implemented in the Java programming language and thus runs on almost any modern computing platforms
- Windows, Mac OS X and Linux
- Comprehensive collection of data preprocessing and modeling techniques
- Supports standard data mining tasks: data preprocessing, clustering, classification, regression, visualization, and feature selection .

- Easy to use GUI
- Provides access to SQL databases
- Using Java Database Connectivity and can process the result

returned by a database query.

The obvious advantage of a package like Weka is that a whole range of data preparation, feature selection and data mining algorithms are integrated. This means that only one data format is needed, and trying out and comparing different approaches becomes really easy. The package also comes with a GUI, which should make it easier to use.

- **Disadvantages of WEKA.**
  - Sequence modeling is not covered by the algorithms included in the Weka distribution.
  - Not capable of multi-relational data mining.
  - Memory bound.
  - Do not implement the newest techniques. For example the MLP implemented has a very basic training algorithm (backprop with momentum), and the SVM only uses polynomial kernels, and does not support numeric estimation. Therefore, it will be necessary to combine WEKA with some of the other tools like Netlab or SVM_torch.
  - Though the software is for free: the documentation for the GUI is quite limited.
  - Limited scaling. For difficult tasks on large datasets, the running time can become quite long, and java sometimes gives an OutOfMemory error. This problem can be reduced by using the '-mxx' option when calling java, where x is memory size (eg '50m'). For large datasets it will always be necessary to reduce the size to be able to work within reasonable time limits.
  - The GUI does not implement all the possible options. Things that could be very useful, like scoring of a test set, are not provided in the GUI, but can be called from the command line interface. So sometimes it will be necessary to switch between GUI and command line.

- **Advantages of R.**
  - R is free and open source software, allowing anyone to use and, importantly, to modify it. R is licensed under the GNU

General Public License, with copyright held by The R Foundation for Statistical Computing.

- The graphical capabilities of R are outstanding, providing a fully programmable graphics language that surpasses most other statistical and graphical packages.
- R is a programming language and environment developed for statistical analysis by practicing statisticians and researchers.

- **Disadvantages of R.**
  - R is not so easy to use for the novice.
  - Many R commands give little thought to memory management, and so R can very quickly consume all available memory.
  - The quality of some packages is less than perfect.

Any Software or Hardware Used:

- **Hardware or software required for WEKA.**

  Hardware:
  - 4GB RAM

  Software:
  - Java
  - 64-bit / 32-bits versions of Windows.
  - 64-bit / 32-bits Linux

- **Hardware or Software required for R.**

  Hardware:
  - The amount of RAM that you need is highly dependent on the work/analysis you will be doing. (More than 1 GB of RAM.)

  Software:
  - 64-bit / 32-bits versions of Windows.
  - 64-bit / 32-bits Linux

INPUT/OUTPUT:

Input:
Dataset with the attributes or features may or may not be trained. (.ARFF, .CSV, C4.5 and binary)

Output:

Graphical Representation of analyzed data from Dataset in case of R.

We get the accuracy of successful classification in weka.

OPERATIONAL STEPS REQUIRED:

- **Steps to operate WEKA**

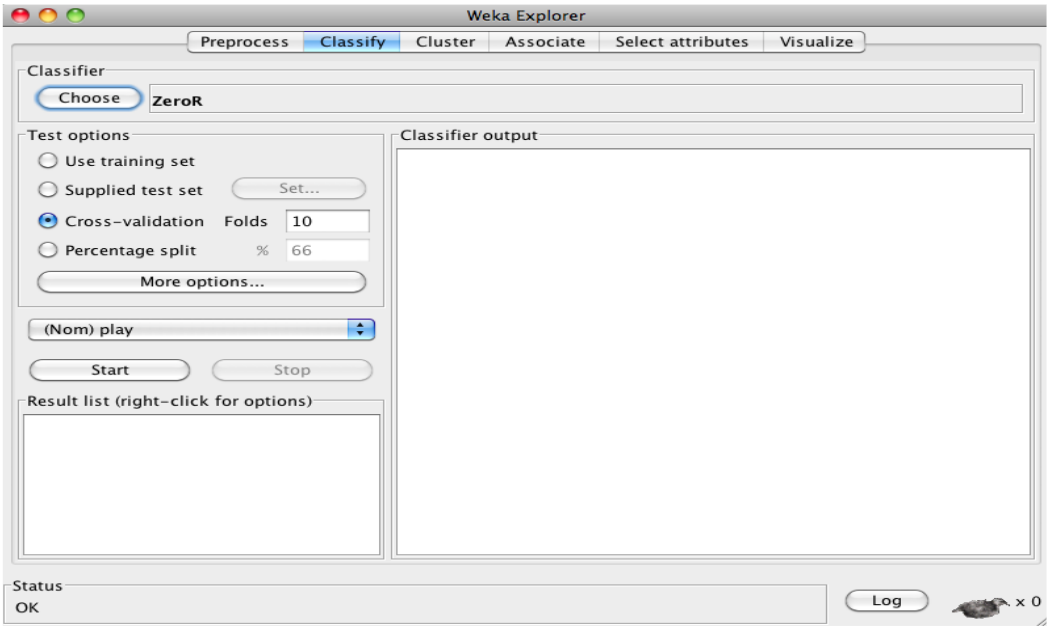
  Choose "WEKA 3.7.x" from Programs. The first interface that appears looks like the one below.

Click on "Explorer"

To load a data set from library eg. 'weather.arff.' So file click on "Open File" and browse the path for 'weather.arff'.

Select the attributes.

Under the Classify tab, click 'Choose' and select a classifier from the drop-down menu. E.g.: 'Decision Stump'



Once, a classifier is chosen, select percentage split and leave it with its default values. The default ratio is 66% for training and 34% for testing.

Click ‘Start’ to train and test the classifier.

- **Steps to operate R**

  Open RStudio.

  Go to Packages - Click on "Install".

  You will get "Install Packages" Window.

  Install from: Package Archive File(tar.gz)

  Package archive: (Browse the path where you have stored the Package_name.tar.gz)

  Click Install.

  In Packages - Tick the Package_name in user library. Click on Update. Click on OK.

  Write Code in R Script (File - New File - R Script) and Save it as "Program1.R"

  Click on run Icon.

  Write R commands on 'console' which is in 'RStudio'.

LATEST TRENDS:

- **For WEKA**

  Weka 3.7.7

- **For R**

  R-3.2.1

  Rtools33

APPLICATIONS:

- **Application of WEKA:**

The WEKA system has been applied successfully in a variety of areas including the areas of agriculture, machine learning research and education.

- **Application of R:**

R applications span the universe from theoretical computational statistics and the hard sciences such as astronomy, chemistry and genomics to practical applications in business, drug development, finance, health care, marketing, medicine and much more. Because R has nearly 5,000 packages (libraries of functions) many of which are dedicated to specific applications you don’t have to be an R genius to begin developing your own applications.

LIMITATIONS:

- **Limitations of WEKA**

  GUI is not as well documented.

  2 different Modules cannot be combined (ex. modules for both PCA and clustering without writing a Java Code).

  The Weka GUI provides several built-in 'visualization' panels but these are very limited.

  Manipulation of data sets is not easy in Weka

- **Limitations of R**

  The biggest limitation in R is the data processing model which is to load everything up in memory and process it. This not only limits the amount of data you can process but it also scales very badly for complex processes.

CONCLUSION:

Downloaded the open source softwares R-base, RStudio and WEKA. Studied the distinct features and functionality of both the software platforms. Found WEKA easier to learn but there are some limitations in case of Graphical Representations, Modifying the dataset etc. R is difficult to learn for novice but its Graphical Representation is better than WEKA.

REFERENCES:

www.cs.waikato.ac.nz/ml/weka/

https://www.cs.auckland.ac.nz/courses/compsci367s1c/tutorials/IntroductionToWeka.pdf

https://www.rstudio.com/

FAQs :

1. What is R?
2. What machines does R run on?
3. What is the current version of R?
4. How can R be obtained?
5. How can R be installed?
6. Which add-on packages exist for R?
7. How can add-on packages be installed?
8. How can add-on packages be used?
9. How can add-on packages be removed?
10. How can I create an R package?

PRACTISE ASSIGNMENTS

1. Try download and install all the available open source analytical tools and analyze its use using different datasets.

Assignment 2 Supervised Learning - Regression

AIM : Supervised Learning - Regression

Generate a proper 2-D data set of N points. Split the data set into Training Data set and Test Data set.
i) Perform linear regression analysis with Least Squares Method.
ii) Plot the graphs for Training MSE and Test MSE and comment on Curve Fitting and Generalization Error.
iii) Verify the Effect of Data Set Size and Bias-Variance Tradeoff.
iv) Apply Cross Validation and plot the graphs for errors.
v) Apply Subset Selection Method and plot the graphs for errors.
vi) Describe your findings in each case.

OBJECTIVE :

To study

- Concept of Supervised Learning - Regression
- How to download different input dataset
- Linear regression analysis with least square methods
- How to add add packages in open source analysis tools
- How to design train and test data set and plot train and test MSE graph

- Concept of curve fitting and generalization error.
- Effect of Data Set Size and Bias-Variance Tradeoff
- How to apply Cross Validation and plot the graphs for errors.
- How to apply Subset Selection Method and plot the graphs for errors

THEORY:

Part 1. Least Square Method for Linear Regression.

1This method is used to find coefficient of model parameters in linear regression. Least square method is simple but found to be more applicable. Least square method is discussed with respect to simple(Univariate) linear regression.
2Consider that we are given with some sample data as,

$(x_1, y_1), (x_2, y_2) , (x_3, y_3) , ……., (x_n, y_n)$
1 Out target is to find simple linear regression model between independent variable X and dependent variable Y as,

1 $Y = \beta_0 + \beta_1 X + \in$

2 As above equation is linear in β_0 , β_1 this is called as linear regression. THIS EQUATION MAY OR MAY NOT BE LINEAR IN 'X'.

3 β_0 , β_1 are called as parameters of the Linear Regression. These parameters can be found by two different method:
1Least Square Method.
Maximum Likelihood Estimation.

2o Both methods derives same formulae to calculate β_0 , β_1 .

4 In turn, target is to find values of β_0 and β_1, which should result in linear regression model (Regression line) that is Best fitting to given sample data.
5Let regression equation is

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

where this value $\hat{y}$ is predicted by regression line.
Residuals or Errors: The difference between actual value of Y given in training data and predicted value of Y i.e. $\hat{Y}$ predicted by linear regression.

Least square method finds values of β_0 and β_1 to result in Regression line for which Sum of Squares of all Residuals or Errors (SSE) is minimum.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$SSE = \sum_{i=1}^{n} (e_i)^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$

$$\boxed{SSE = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2}$$

1 To get minimum value of SSE for β_0 and β_1, partial derivative of SSE w.r.t. β_0 and β_1 must be equal to

$$\therefore \frac{\partial SSE}{\partial \beta_0} = 0 \qquad \therefore \frac{\partial SSE}{\partial \beta_1}$$

$$= 0$$

$$\frac{\partial SSE}{\partial \beta_0} = \frac{\partial}{\partial \beta_0} \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = 0$$

0.

$$\therefore \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\boxed{\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

Part 1.

Steps:

11. Graphically Apply Linear regression by selecting any line as regression line.
22. Find Squared Error in each case.
33. Apply least square method for linear regression.
44. Find Squared Error and compare with errors in step 2.

Sample R Code For the same:

```
rm(list=ls()) # clear all previously stored variables/functions
input <- read.csv("C:/Desktop/input.csv")
# if file can not read then down it on ur local drive and the read by giving its path
```

```
attach(input)
names(input) # shows all varaibles
head(input,8) # shows the first 8 lines of data
plot(runs~at_bats,main="Runs Vs At_bats",xlim=c(5350,5750),ylim=c(500,900))
cor(runs,at_bats) # cor = correlation. What does this value represent?

#Exercise 1: How would you describe this scatter plot? Are there any unusual

#feature? Would it be appropriate to _t a least squares linear regression model

#to predict the number of runs using the number of at-bats? (Please check all

#conditions for using the regression model.) Interpret the numerical output.

# define following plot function for plotting interactive plot

# this will perform linear regression according to line selected by user

# user can select a line by clicking any two points in scatter plot

# Select 5-8 regression line & observe the sum of squares errors for
each selected line.

plotSS <- function(x, y, showSquares = FALSE, leastSquares = FALSE)
{
plot(y~x, asp = 1, xlab=paste(substitute(x)), ylab=paste(substitute(y)))
if(leastSquares)
{
m1 <- lm(y~x)

y.hat <- m1$fit
}
else {
cat("Click any two points to make a line.\n")
pt1 <- locator(1)
points(pt1$x, pt1$y, pch = 8, col = "red")
pt2 <- locator(1)
points(pt2$x, pt2$y, pch = 8, col = "red")
pts <- data.frame("x" = c(pt1$x, pt2$x),"y" = c(pt1$y, pt2$y))
m1 <- lm(y ~ x, data = pts)
y.hat <- predict(m1, newdata = data.frame(x))
# title(paste("b0 = ", pt1$y-(pt2$y-pt1$y)/(pt2$x-pt1$x)*pt1$x, ", b1 = ", (pt2$y-
pt1$y)/(pt2$x-pt1$x)))
}
r <- y - y.hat
abline(m1)
oSide <- x - r
LLim <- par()$usr[1]
RLim <- par()$usr[2]
oSide[oSide < LLim | oSide > RLim] <- c(x + r)[oSide < LLim | oSide > RLim] # move
boxes to avoid margins
n <- length(y.hat)
for(i in 1:n)
```

```
{
lines(rep(x[i], 2), c(y[i], y.hat[i]), lty = 2, col = "blue")
if(showSquares)
{
lines(rep(oSide[i], 2), c(y[i], y.hat[i]), lty = 3, col = "orange")
lines(c(oSide[i], x[i]), rep(y.hat[i],2), lty = 3, col = "orange")
lines(c(oSide[i], x[i]), rep(y[i],2), lty = 3, col = "orange")
}
}
SS <- round(sum(r^2), 3)
cat("\r ")
print(m1)
cat("Sum of Squares: ", SS, "\n")
}
plotSS(x = at_bats, y = runs)
plotSS(x = at_bats, y = runs, showSquares = TRUE)
```

#Searching for the best predictor buy using Least square #method

```
fit1 <- lm(runs ~ at_bats)
summary(fit1)
plotSS(x = at_bats, y = runs, leastSquares = TRUE)
#observe value of parameters of linear regression and compare squared error with
# previously obtained errors.
```

Part 2. Divide data into training data set and testing data set for different sizes.

For example if data contains 1000 records then following can be different cases for training data set and test data.

Theory: Include theory on effect of Data Size on Linear Regression. How to decide minimum size of dataset?

Case	Size of training data	Size of test data	Apply Linear Regression to each case	Find Training MSE	Find Test MSE
1	100	900	Tr_MSE[1]	Ts_MSE[1]	
2	200	800	Tr_MSE[2]	Ts_MSE[2]	
3	300	700	Tr_MSE[3]	Ts_MSE[3]	
4	400	600	Tr_MSE[4]	Ts_MSE[4]	
5	500	500	Tr_MSE[5]	Ts_MSE[5]	
6	600	400	Tr_MSE[6]	Ts_MSE[6]	
7	700	300	Tr_MSE[7]	Ts_MSE[7]	
8	800	200	Tr_MSE[8]	Ts_MSE[8]	
9	900	100	Tr_MSE[9]	Ts_MSE[9]	

Plot the graph for
11. Y axis : Size of training dataset 100,200,300,400,500,600,700,800,900
X axis: Training_MSE

1
22. Y axis : Size of dataset 100,200,300,400,500,600,700,800,900
X axis: Test_MSE

IDEAL Graph for linear Regression:

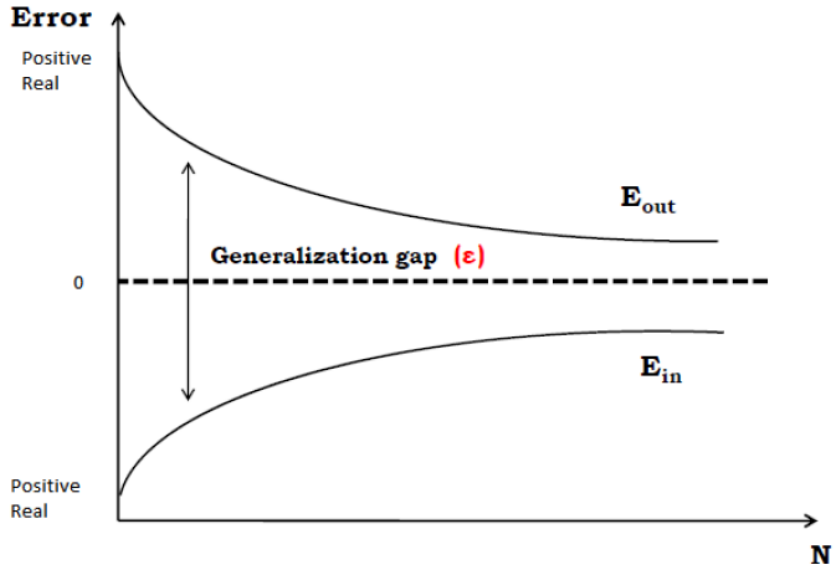


Fig: 1 Ideal Learning Curve
(Plots of Eout , Ein Vs Size(N)).

NOTE: Observe the difference between observed graph and ideal graph. “Nature of training and characteristic of dataset if matches well with the capacity of a learner (Complexity of underlined model), good generalization results can occur.” In such case the learning curves i.e plots of Eout , Ein Vs Size(N) will be ideal as shown in Fig. 1

Part 3. Include theory for K-fold Cross validation.

Theory: Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on ``new'' data. This is the basic idea for a whole class of model evaluation methods called cross validation.

K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

For a dataset of 1000 records apply 10-fold cross validation. In each case find MSE (Mean Square Error). Save them and plot against k.

Y axis : Size of training dataset 100,200,300,400,500,600,700,800,900

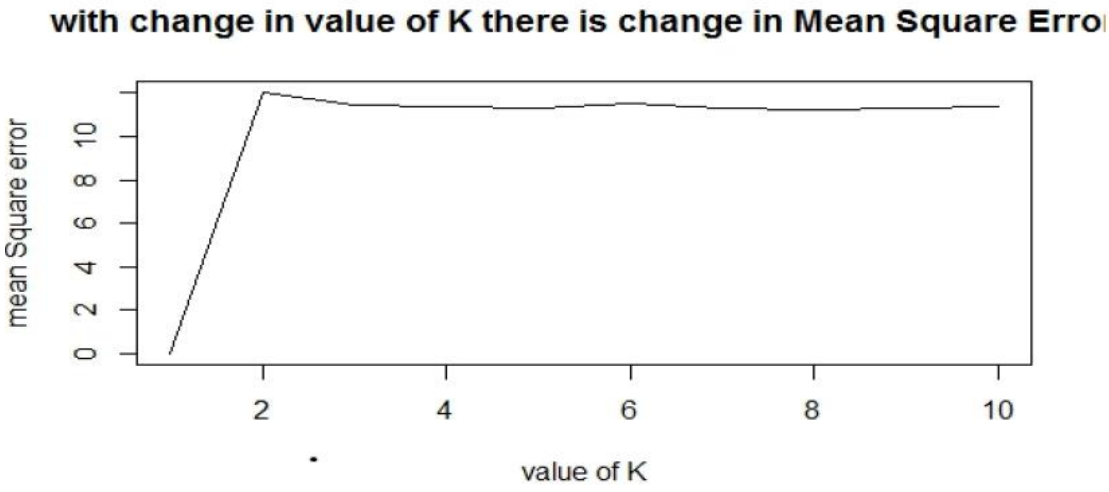
X axis: MSE

Sample R Code For the same:

```
#install.packages(MASS)
#install.packages(ISLR)

library(MASS)
#MASS is required to access data set provided by R
library(ISLR)
#ISLR is required to access data set provided by R


library(DAAG)
#DAAG is required for function cv.lm()

library("nlme")
#for some regression function.

#library("ISLR", lib.loc="~/R/win-library/3.2")
#data(): lists various data sets available in R
# let us examine cv.lm for different individual cases K=m=2,3,8
lmodel <- cv.lm(df = Auto, mpg ~ cylinders + displacement + horsepower +
weight + acceleration + year + origin, m = 2 , dots = FALSE, seed=29,
plotit=TRUE, printit=TRUE)

lmodel <- cv.lm(df = Auto, mpg ~ cylinders + displacement + horsepower +
weight + acceleration + year + origin, m = 3 , dots = FALSE, seed=29,
plotit=TRUE, printit=TRUE)

lmodel <- cv.lm(df = Auto, mpg ~ cylinders + displacement + horsepower +
weight + acceleration + year + origin, m = 8 , dots = FALSE, seed=29,
plotit=TRUE, printit=TRUE)

cv.error.10 = rep (0,10)
set.seed(17)
#seed is required for randomized selection of records.

#let us apply Kfold Cross validation for K=2,3,4,5,6,7,8,9,10
# by using this loop
for(i in 2:10)
{

  lmodel <- cv.lm(df = Auto, mpg ~ cylinders + displacement +  horsepower
+  weight + acceleration + year +  origin, m = i , dots =
```

```
                    FALSE, seed=29,
                  plotit=TRUE, printit=TRUE)

  lmodel

     #record Mean Square for each value of K

  cv.error.10[i]=attr(lmodel,"ms")

}


j<-1

for (j in 1:10)
{
  # printing MS error for each value of K
  print(cv.error.10[j])
}

X=c(1,2,3,4,5,6,7,8,9,10)
Y=cv.error.10

  # plotting MS error for each value of K against K

plot(X,Y, type='l',  xlab = "value of K" , ylab =  "mean Square error",
main = "with change in value of K there is change in Mean Square Error")
```

OUTPUT:



NOTE: Observe the effect of K on Mean Square Error.

Part 4. Include similarly theory for Subset selection.(Though implementation of Subset selection is optional, students must understand it theoretically, so that they can answer questions in ORAL)

Steps:

1.Apply Linear Regression to given data and find Mean Square Error.

2. Apply Subset Selection and for each case find Mean Square Error.

3. Conclude which subset of features when selected yields minimum Mean Square Error.

NOTE: Observe the effect Subset Selected on Mean Square Error.

FAQs :

1. **how to construct minimum mean squared error estimator from a sample data? and show that it is unbiased?**
2. **How calculate the mean of Mean Squared Errors?**
3. **How to do iterative ANOVA and extract Mean Square Values from lm object in R**
4. **How to know if a regression model generated by random forests is good? (MSE and %Var(y)) [closed]**

PRACTISE ASSIGNMENTS

2. Perform above on different input dataset.

Assignment 3 Supervised Learning - Classification

AIM : Supervised Learning - Classification

Implement Naïve Bayes Classifier and K-Nearest Neighbor Classifier on Data set of your choice. Test and Compare for Accuracy and Precision.

OBJECTIVE :

To study

- Benefits of concept of Classifier
- How to design algorithms for Naïve Bayes Classifier
- How to design algorithms for K-Nearest Neighbor Classifier
- Analysis of the above algorithms and compare Accuracy and Precision on different data sets.

THEORY:

Naive-Bayes Classification Algorithm

1. Introduction to Bayesian Classification

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems.

This Classification is named after Thomas Bayes (1702-1761), who proposed the Bayes
Theorem.

Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data.

Uses of Naive Bayes classification:

1. Naive Bayes text classification (http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html)

The Bayesian classification is used as a probabilistic learning method (Naive Bayes text classification). Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents.

2. Spam filtering (http://en.wikipedia.org/wiki/Bayesian_spam_filtering)

Spam filtering is the best known use of Naive Bayesian text classification. It makes use of a naive Bayes classifier to identify spam e-mail.

Bayesian spam filtering has become a popular mechanism to distinguish illegitimate spam email from legitimate email (sometimes called "ham" or "bacn").[4] Many modern mail clients implement Bayesian spam filtering. Users can also install separate email filtering programs. Server-side email filters, such as DSPAM, SpamAssassin, SpamBayes, Bogofilter and ASSP,
make use of Bayesian spam filtering techniques, and the functionality is sometimes embedded within mail server software itself.

3. Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering (http://eprints.ecs.soton.ac.uk/18483/)

Recommender Systems apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given resource.

It is proposed a unique switching hybrid recommendation approach by combining a Naive
Bayes classification approach with the collaborative filtering. Experimental results on two different data sets, show that the proposed algorithm is scalable and provide better
performance–in terms of accuracy and coverage–than other algorithms while at the same time
eliminates some recorded problems with the recommender systems.

4. Online applications (http://www.convo.co.uk/x02/)

This online application has been set up as a simple example of supervised machine learning and affective computing. Using a training set of examples which reflect nice, nasty or neutral sentiments, we're training Ditto to distinguish between them.

Simple Emotion Modelling, combines a statistically based classifier with a dynamical model. The Naive Bayes classifier employs single words and word pairs as features. It allocates user utterances into nice, nasty and neutral classes, labelled +1, -1 and 0 respectively. This numerical output drives a simple first-order dynamical system, whose state represents the simulated emotional state of the experiment's personification, Ditto the donkey.

1.1. Independence

1.1.1. Example:

Suppose there are two events:

M: Manuela teaches the class (otherwise it's Andrew)

S: It is sunny

"The sunshine levels do not depend on and do not influence who is teaching."

1.1.2. Theory:

From P(S | M) = P(S), the rules of probability imply:

P(~S | M) = P(~S)

P(M | S) = P(M)

P(M ^ S) = P(M) P(S)

P(~M ^ S) = P(~M) P(S)

P(M^~S) = P(M)P(~S)

P(~M^~S) = P(~M)P(~S)

1.2.3. Theory applied on previous example:

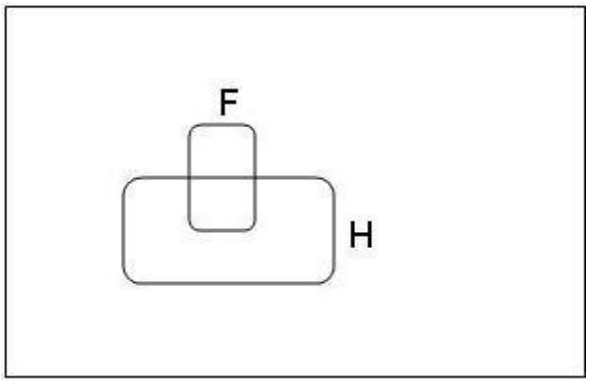
"The sunshine levels do not depend on and do not influence who is teaching." can be specified very simply:

P(S | M) = P(S)

"Two events A and B are statistically independent if the probability of A is the same value when B occurs, when B does not occur or when nothing is known about the occurrence of B"

1.2. Conditional Probability

1.2.1. Simple Example:



H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

$P(A \mid B) \; P(\neg A \mid B) = 1$

“Headaches are rare and flu is rarer, but if you’re coming down with ‘flu there’s a 50-50 chance you’ll have a headache.”

P(H|F) = Fraction of flu-inflicted worlds in which you have a headache =

$$= \frac{\text{\#worlds with flu and headache}}{\text{\#worlds with flu}} = \frac{\text{Area of “H and F” region}}{\text{Area of “F” region}} = \frac{P(H \wedge F)}{P(F)}$$

1.2.2. Theory:

P(A|B) = Fraction of worlds in which B is true that also have A true

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

Corollary:
$P(A \wedge B) = P(A|B)\, P(B)$
P(A|B)+P(¬A|B) = 1

$$\sum_{k=1}^{n} P(A \mid B_k) = 1$$

1.2.3. Detailed Example

M : Manuela teaches the class
S : It is sunny
L : The lecturer arrives slightly late.

Assume both lecturers are sometimes delayed by bad weather. Andrew is more likely to arrive late than Manuela.

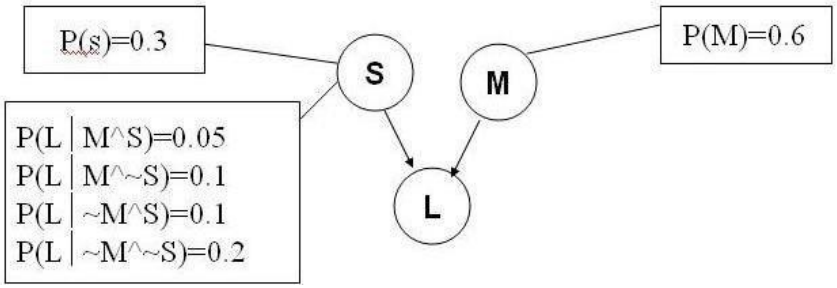
Let's begin with writing down the knowledge: P(S □ M) = P(S), P(S) = 0.3, P(M) = 0.6

Lateness is not independent of the weather and is not independent of the lecturer. Therefore

Lateness is ***dependant*** on both weather and lecturer

$P(S \mid M) = P(S)$
$P(S) = 0.3$
$P(M) = 0.6$

$P(L \mid M \wedge S) = 0.05$
$P(L \mid M \wedge \sim S) = 0.1$
$P(L \mid \sim M \wedge S) = 0.1$
$P(L \mid \sim M \wedge \sim S) = 0.2$



Read the absence of an arrow between S and M to mean "it would not help me predict M if I knew the value of S"

Read the two arrows into L to mean that if I want to know the value of L it may help me to know M and to know S.

1.3. Conditional Independence

1.3.1. Example:

Suppose we have these three events:

M : Lecture taught by Manuela
L : Lecturer arrives late
R : Lecture concerns robots

Suppose:

Andrew has a higher chance of being late than Manuela. Andrew has a higher chance of giving robotics lectures.

Once you know who the lecturer is, then whether they arrive late doesn't affect whether the lecture concerns robots.

1.3.2. Theory:

R and L are conditionally independent given M if for all x,y,z in {T,F}: P(R=x □ M=y ^ L=z) = P(R=x □ M=y)

More generally:

Let S1 and S2 and S3 be sets of variables.

Set-of-variables S1 and set-of-variables S2 are conditionally independent given S3 if for all assignments of values to the variables in the sets, P(S1's assignments S2's assignments & S3's assignments)= P(S1's assignments S3's assignments)

P(A|B) = P(A ^B)/P(B)
Therefore P(A^B) = P(A|B).P(B) – also known as Chain Rule
Also P(A^B) = P(B|A).P(A)
Therefore P(A|B) = P(B|A).P(A)/P(B)

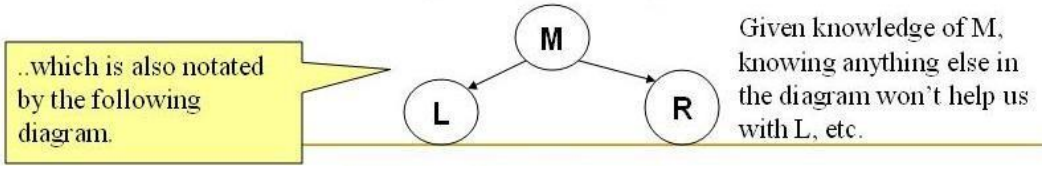
P(A,B|C) = P(A^B^C)/P(C)
= P(A|B,C).P(B^C)/P(C) – *applying chain rule*
= P(A|B,C).P(B|C)
= P(A|C).P(B|C) , *If A and B are conditionally independent given C.*
This can be extended for n values as P(A1,A2…An|C) = P(A1|C).P(A2|C)…P(An|C) *if A1, A2…An are conditionally independent given C.*

1.3.3. Theory applied on previous example:

For the previous example, we can use the following

notations: P(R M,L) = P(R M) and P(R ~M,L) =

P(R ~M)
We express this in the following way:
"R and L are conditionally independent given M"



2. Bayes Theorem

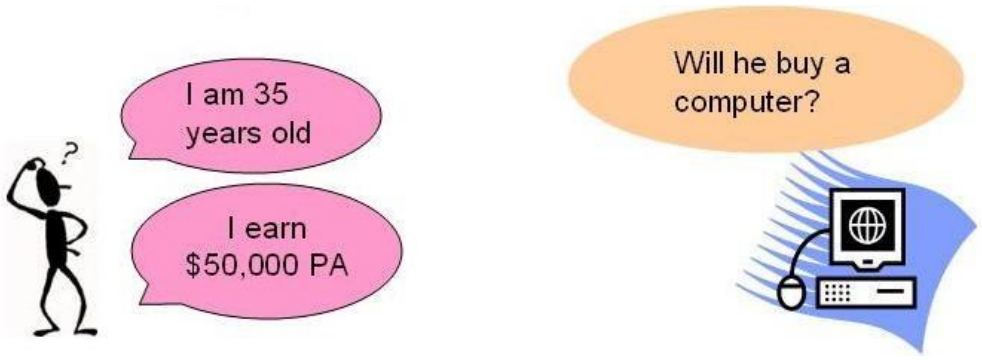
Bayesian reasoning is applied to decision making and inferential statistics that deals with probability inference. It is used the knowledge of prior events to predict future events.

Example: Predicting the color of marbles in a basket

2.1. Example:

rec	Age	Income	Student	Credit_rating	Buys_computer
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31...40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31...40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<-=30	Medium	Yes	Excellent	Yes
r12	31...40	Medium	No	Excellent	Yes
r13	31...40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

Table1: Data table



2.2. Theory:

The Bayes Theorem:

$$P(h/D) = \frac{P(D/h)}{P(h)} \\ P(D)$$

P(h) : Prior probability of hypothesis
h P(D) : Prior probability of training
data D P(h/D) : Probability of h given
D
P(D/h) : Probability of D given h

2.3. *Theory applied on previous example:*

D : 35 year old customer with an income of $50,000
PA
h : Hypothesis that our customer will buy our computer

P(h/D) : Probability that customer D will buy our computer given that we know his age and income
P(h) : Probability that any customer will buy our computer regardless of age (Prior Probability) P(D/h) : Probability that the customer is 35 yrs old and earns $50,000, given that he has bought our computer (Posterior Probability)
P(D) : Probability that a person from our set of customers is 35 yrs old and earns $50,000

2.4. Maximum A Posteriori (MAP) Hypothesis

2.4.1.
Example:

h1: Customer buys a computer =
Yes h2 : Customer buys a
computer = No
where h1 and h2 are subsets of our Hypothesis Space 'H'
P(h/D) (Final Outcome) = arg max{ P(D/h1) P(h1) , P(D/h2)
P(h2)} P(D) can be ignored as it is the same for both the terms

2.4.2.
Theory:
Generally we want the most probable hypothesis given the training data
hMAP = arg max P(h/D) (where h belongs to H and H is the hypothesis space)

$$hMAP = \arg\max \frac{P(D/h)}{P(h)} P(D)$$

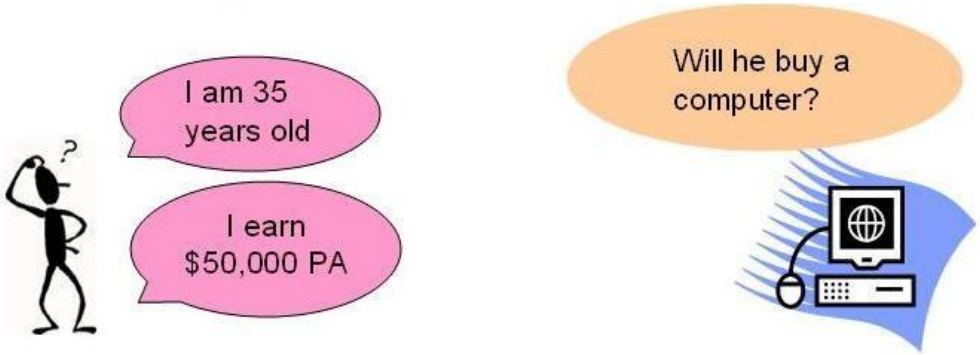
$$hMAP = \arg\max P(D/h)\, P(h)$$

2.5. Maximum Likelihood (ML) Hypothesis

2.5.1. Example:

Sl No	Age	Income	Student	Credit rating	Buys computer
1	35	Medium	Yes	Fair	Yes
2	30	High	No	Average	No
3	40	Low	Yes	Good	No
4	35	Medium	No	Fair	Yes
5	45	Low	No	Fair	Yes
6	35	High	No	Excellent	Yes
7	35	Medium	No	Good	No
8	25	Low	No	Good	No
9	28	High	No	Average	No
10	35	Medium	Yes	Average	Yes

Table 2



2.5.2. Theory:

If we assume P(hi) = P(hj) where the calculated probabilities amount to the same
Further simplification leads to:
hML = arg max P(D/hi) (where hi belongs to H)

2.5.3. Theory applied on previous example:

P (buys computer = yes) = 5/10 = 0.5
P (buys computer = no) = 5/10 = 0.5
P (customer is 35 yrs & earns $50,000) = 4/10 = 0.4
P (customer is 35 yrs & earns $50,000 / buys computer = yes) = 3/5 =0.6
P (customer is 35 yrs & earns $50,000 / buys computer = no) = 1/5 = 0.2

Customer buys a computer P(h1/D) = P(h1) * P (D/ h1) / P(D) = 0.5 * 0.6 / 0.4
Customer does not buy a computer P(h2/D) = P(h2) * P (D/ h2) / P(D) = 0.5 * 0.2 / 0.4

Final Outcome = arg max {P(h1/D) , P(h2/D)} = max(0.6, 0.2)
=> Customer buys a computer

3. Naïve Bayesian Classification

It is based on the Bayesian theorem It is particularly suited when the dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. In spite over-simplified assumptions, it often performs better in many complex real- world situations

Advantage: Requires a small amount of training data to estimate the parameters

3.1. Example

rec	Age	Income	Student	Credit_rating	Buys_computer
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31...40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31...40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<-=30	Medium	Yes	Excellent	Yes
r12	31...40	Medium	No	Excellent	Yes
r13	31...40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

X = (age= youth, income = medium, student = yes, credit_rating =

fair) A person belonging to tuple X will buy a computer?

3.2.Theory:

Derivation:

D : Set of tuples

Each Tuple is an 'n' dimensional attribute vector

X : (x1,x2,x3,…. xn)

Let there be 'm' Classes : C1,C2,C3…Cm

Naïve Bayes classifier predicts X belongs to Class Ci iff

P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i

Maximum Posteriori Hypothesis

P(Ci/X) = P(X/Ci) P(Ci) /
P(X)

Maximize P(X/Ci) P(Ci) as P(X) is constant

With many attributes, it is computationally expensive to evaluate P(X/Ci). Naïve Assumption of "class conditional independence"

$$P(X/.Ci) \approx \prod_{k=1}^{n} P(x_k/Ci)$$

P(X/Ci) = P(x1/Ci) * P(x2/Ci) *…* P(xn/ Ci)

3.3. Theory applied on previous example:

P(C1) = P(buys_computer = yes) = 9/14 =0.643
P(C2) = P(buys_computer = no) = 5/14= 0.357
P(age=youth /buys_computer = yes) = 2/9 =0.222
P(age=youth /buys_computer = no) = 3/5 =0.600
P(income=medium /buys_computer = yes) = 4/9 =0.444
P(income=medium /buys_computer = no) = 2/5 =0.400
P(student=yes /buys_computer = yes) = 6/9 =0.667
P(student=yes/buys_computer = no) = 1/5 =0.200
P(credit rating=fair /buys_computer = yes) = 6/9 =0.667
P(credit rating=fair /buys_computer = no) = 2/5 =0.400

P(X/Buys a computer = yes) = P(age=youth /buys_computer = yes) * P(income=medium /buys_computer = yes) * P(student=yes /buys_computer = yes) * P(credit rating=fair /buys_computer = yes) = 0.222 * 0.444 * 0.667 * 0.667 = 0.044

P(X/Buys a computer = No) = 0.600 * 0.400 * 0.200 * 0.400 = 0.019

Find class Ci that Maximizes P(X/Ci) * P(Ci)
=>P(X/Buys a computer = yes) * P(buys_computer = yes) = 0.028
=>P(X/Buys a computer = No) * P(buys_computer = no) = 0.007

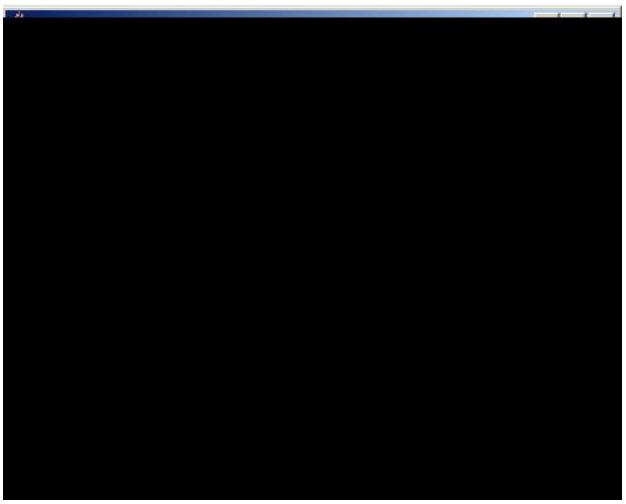
Prediction : Buys a computer for Tuple X

4. Sample running example with weka

4.1. Bayesian Network Classifiers in Weka

Let U = {x1,... ,xn}, n ~ 1 be a set of variables. A Bayesian network B over a set of variables U is a network structure BS, which is a directed acyclic graph (DAG) over U and a set of probability tables BP = {p(u|pa(u))|u 2 U} where pa(u) is the set of parents of u in BS. A Bayesian network represents a probability distributions P(U) = Q u2U p(u|pa(u)).

Below, a Bayesian network is shown for the variables in the iris data set. Note that the links between the nodes class, petallength and petalwidth do not form a directed cycle, so the graph is a proper DAG.

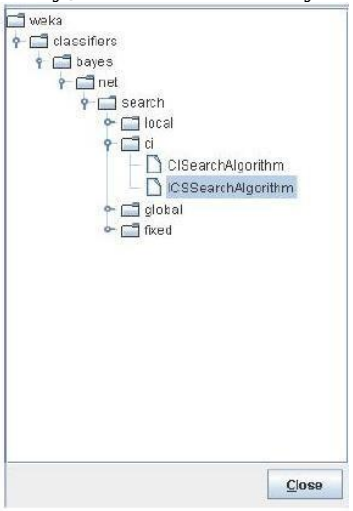


4.2. Conditional independence test based structure learning

Conditional independence tests in Weka are slightly different from the standard tests described

in the literature. To test whether variables x and y are conditionally independent given a set of variables Z, a network structure with arrows z zz y is compared with one with arrows {x

 y} z zz .y. A test is performed.



At the moment, only the ICS [9]and CI algorithm are implemented.

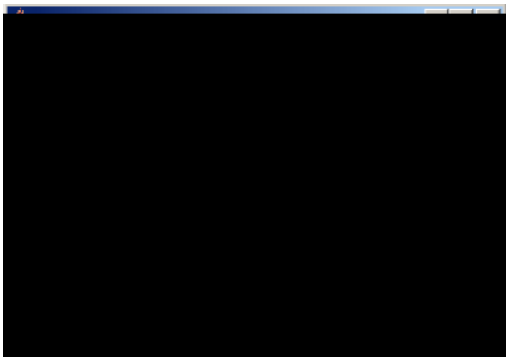
The ICS algorithm makes two steps, first find a skeleton (the undirected graph with edges if f there is an arrow in network structure) and second direct all the edges in the skeleton to get a DAG.

Starting with a complete undirected graph, we try to find conditional independencies <x, y | Z> in the data. For each pair of nodes x, y, we consider sets Z starting with cardinality 0, then 1 up to a user defined maximum. Further-more, the set Z is a subset of nodes that are neighbors of both x and y. If an independency is identified, the edge between x and y is removed from the skeleton.

The first step in directing arrows is to check for every configuration x - -z - - y where x and y not connected in the skeleton whether z is in the set Z of variables that justified removing the link between x and y (cached in the first step). If z is not in Z, we can assign direction x z y. Finally, a set of graphical rules is applied to direct the remaining arrows.

```
Rule 1: i->j--k & i-/-k => j->k
Rule 2: i->j->k & i--k => i->k
Rule 3  m
              /|\
             i | k  => m->j
     i->j<-k  \|/
               j

Rule 4  m
              / \
             i---k  => i->m & k->m
       i->j   \ /
               j
Rule 5: if no edges are directed then take a random one (first we can find)
```

The ICS algorithm comes with the following options.



Since the ICS algorithm is focused on recovering causal structure, instead of finding the optimal classifier, the Markov blanket correction can be made afterwards.

Specific options:

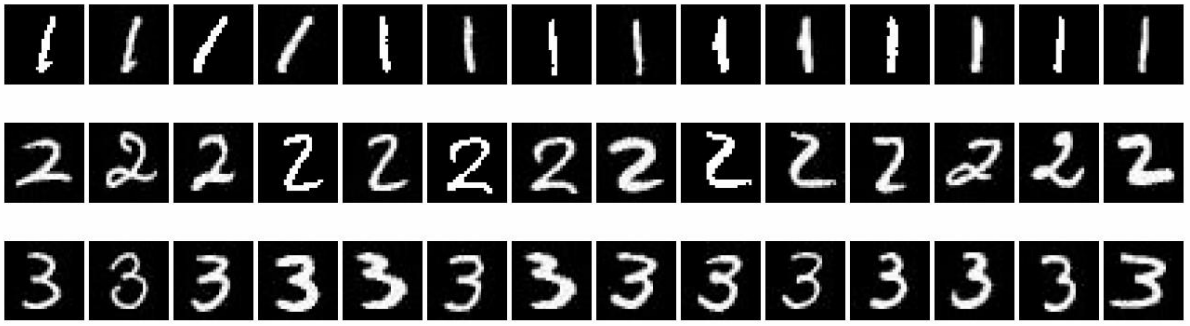
The maxCardinality option determines the largest subset of Z to be considered in conditional independence tests <x, y|Z>.

The scoreType option is used to select the scoring metric.

5. Exercises

Implement, test and interpret the results for Naïve Bayes algorithm for the following problems, using the attached input files

5.1. Being given the following binary files (imagini.zip), which represent the classes for 1, 2 and 3, you must find out the class of a digit in an image.



There will be used as attributes white pixels (value 255) and the positions of their

appearance. Algorithm:

Step 1: It is loaded the image which will be classified as being ONE, TWO or THREE

Step 2: There are loaded the images found in the folder **images.** The name of the files belonging to class ONE are: "image1_*.jpg", the ones belonging to class TWO are: "image2_*.jpg" and the ones for class THREE are : "image3_*.jpg".

Step3: It is determined the a priori probability for each class:
P(UNU) = NrTemplateInClassONE /
NumberTotalTemplates P(DOI) = NrTemplateInClassTWO /
NumberTotalTemplates P(TREI) =
NrTemplateInClassTHREE / NumberTotalTemplates

Step 4: It is determined the probability that the image from the Step 1 to be in class ONE, TWO or THREE. Let (i,j) be the position of a white pixel in the image. It is calculated the probability that the pixel having the coordinates (i, j) to be white for the class ONE, TWO and THREE.

count1i,j = 0
for k = 1,n ; n – the number of images in class ONE
if image1_k(i,j) = 255
then count1i,j =
count1i,j + 1
probability1(i,j) =count1i,j / NrTemplateInClassONE

count2i,j = 0
for k = 1,n ; n- the number of images in class TWO

if image2_k(i,j) = 255
then count2i,j =
count2i,j + 1
probability2(i,j) =count2i,j / NrTemplateInClassTWO

count3i,j = 0
for k = 1,n ; n- the number of images in class THREE
if image3_k(i,j) = 255
then count3i,j =
count3i,j + 1
probability 3(i,j) =count3i,j / NrTemplateInClassTHREE

Step 5.
The posteriori probability that the image in Step 1 to be in class ONE is:
P(T|ONE) = average (probabilitate1(i,j)); (i, j) – the position of the white pixels in the image from Step1

Step 6.
The posteriori probability that the image in Step 1 to be in class TWO is:
P(T|TWO) = average (probabilitate1(i,j)); (i, j) – the position of the white pixels in the image from Step1

Step 7:
The posteriori probability that the image in Step 1 to be in class THREE is:
P(T|THREE) = average (probabilitate1(i,j)); (i, j) – the position of the white pixels in the image from Step1

Step 8:
It is determined the probability P for each image class and it is assigned the image from Step1 to the class of images that has the greatest probability.
P(ONE|T) = P(T| ONE)*P(ONE)
P(TWO|T) = P(T| TWO)*P(TWO)
P(THREE|T) = P(T|
THREE)*P(THREE)

In order to load an image and to load pixels from an image in an array, you can use the following java code:

```
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import
javax.swing.*;
import java.util.*;
public class CImagesLoad {
        Vector<Image> images1 = new Vector<Image>();
        Vector<Image> images2 = new Vector<Image>();
        Vector<Image> images3 = new Vector<Image>();
        public String getFile(boolean isSaveDialog)
        {
                String currentDirectoryName = new File("").getAbsolutePath() +File.separator;
                try{
                        JFileChooser fc = new JFileChooser(new File(new
                        File(currentDirectoryName).getParent()));
                        int result = 0;
                        if(!isSaveDialog)
                                        result =
                                fc.showOpenDialog(null);
```

```
                    else

                            result = fc.showSaveDialog(null);
                    if(result==JFileChooser.CANCEL_OPTION) return null;
                    else { //if(result==JFileChooser.APPROVE_OPTION){
                            return fc.getSelectedFile().getAbsolutePath();
                    }
            }
            catch(Exception  e)
            {
                    return null;
            }
}
public void load_images (int
            template){ String f =
            getFile(false);
            if (f==null)
            {
                    return;
            }
            int k = 1;
            while (true)
            {
                    String curent = new java.io.File (f).getAbsolutePath  ();
                    int   pos   =   curent.lastIndexOf
                    ("\\"); curent  =  curent.substring
                    (0, pos); if (k < 10)
                    {
                            curent += "\\image" + template + "_0" + k + ".jpg";
                    }       curent += "\\image" + template + "_0" + k + ".jpg";
                     else
                    {

                    }       curent += "\\image" + template + "_" + k + ".jpg";
                    Image img = null;
                    img = new javax.swing.ImageIcon(curent).getImage();
                    if(img==null || img.getWidth(null)<=0  ||img.getHeight(null)<=0)
                    {
                            System.out.println("The  file \n" + f.toString() + "\nhas an unsupported image
                    format");
                            break;
                    }
                     else
                    {

                            k++;
                            switch (template)
                            {
                            case 1:
                                    images1.add (img);
                                    images1.add (img);
                            case    2:break;

                            case    3:images2.add (img);
                                    break;
                            default:
                                    images3.add (img);
                            }       break;
                    }
            }
                                    System.out.println("Other class");
                                    break;
```

```
        }
        public void load_pixels (Image image)
        {
                int width = image.getWidth(null);
                int height = image.getHeight(null);
                // Allocate buffer to hold the image's pixels
                int pixels[] = new int[width * height];
                // Grab pixels
                PixelGrabber pg = new PixelGrabber (image, 0, 0, width, height,
                pixels, 0, width);
                try
                {
                        pg.grabPixels();
                }
                catch (InterruptedException e)
                {
                        System.out.println ("Error image loading");
                }
        }
}
```

5.2. Modify 5.1 in order to classify images that are belonging to class 4.

5.3. Implement the example 3.1 for the following tuple X:
X = (age= youth, income = high, student = no, credit_rating = fair) Find out if a person belonging to tuple X will buy a computer

6.References

http://en.wikipedia.org/wiki/Bayesian_probability
http://en.wikipedia.org/wiki/Naive_Bayes_classifier
http://www.let.rug.nl/~tiedeman/ml05/03_bayesian_handout.pdf
http://www.statsoft.com/textbook/stnaiveb.html
http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/mlbook/ch6.pdf

Chai, K.; H. T. Hn, H. L. Chieu; "Bayesian Online Classifiers for Text Classification and Filtering", Proceedings of the 25th annual international ACM SIGIR conference on Research and Development in Information Retrieval, August 2002, pp 97-104

DATA MINING Concepts and Techniques,Jiawei Han, Micheline Kamber Morgan Kaufman Publishers, 2003

K Nearest neighbor:

1 Introduction

The purpose of the k Nearest Neighbours (kNN) algorithm is to use a database in which the data points are separated into several separate classes to predict the classification of a new sample point. This sort of situation is best motivated through examples.

Example

Suppose a bank has a database of people's details and their credit rating. These details would probably be the person's financial characteristics such as how much they earn, whether they own or rent a house, and so on, and would be used to calculate the person's credit rating. However, the process for calculating the credit rating from the person's details is quite expensive, so the bank would like to find some way to reduce this cost. They realise that by the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.

Example

Suppose a botanist wants to study the diversity of flowers growing in a large meadow. However, he does not have time to examine each flower himself, and cannot afford to hire assistants who

know about flowers to help him. Instead, he gets people to measure various characteristics of the flowers, such as the stamen size, the number of petals, the height, the colour, the size of the flower head, etc, and put them into a computer. He then wants the computer to compare them to a pre-classified database of samples, and predict what variety each of the flowers is, based on its characteristics.

The General Case

In general, we start with a set of data, each data point of which is in a know class. Then, we want to be able to predict the classification of a new data point based on the known classifications of the observations in the database. For this reason, the database is known as our training set, since it trains us what objects of the different classes look like. The process of choosing the classification of the new observation is known as the classification problem, and there are several ways to tackle it. Here, we consider choosing the classification of the new observation based on the classifications of the observations in the database which it is "most similar" to.

However, deciding whether two observations are similar or not is quite an open question. For instance, deciding whether two colours are similar is a completely different process to deciding whether two paragraphs of text are similar. Clearly, then, before we can decide whether two observations are similar, we need to find some way of comparing objects. The principle trouble with this is that our data could be of many different types - it could be a number, it could be a colour, it could be a geographical location, it could be a true/false (boolean) answer to a question, etc - which would all require different ways of measuring similarity. It seems then that this first problem is one of preprocessing the data in the database in such a way as to ensure that we can compare observations. One common way of doing this is to try to convert all our characteristics into a numerical value, such as converting colours to RGB values, converting locations to latitude and longitude, or converting boolean values into ones and zeros. Once we have everything as numbers, we can imagine a space in which each of our characteristics is represented by a different dimension, and the value of each observation for each characteristic is its coordinate in that dimension. Then, our observations become points in space and we can interpret the distance between them as their similarity (using some appropriate metric).

Even once we have decided on some way of determining how similar two observations are, we still have the problem of deciding which observations from the database are similar enough to our new observation for us to take their classification into account when classifying the new observation. This problem can be solved in several different ways, either by considering all the data points within a certain radius of the new sample point, or by taking only a certain number of the nearest points. This latter method is what we consider now in the k Nearest Neighbours Algorithm.

2 The k Nearest Neighbours Algorithm

As discussed earlier, we consider each of the characteristics in our training set as a different dimension in some space, and take the value an observation has for this characteristic to be its coordinate in that dimension, so getting a set of points in space. We can then consider the similarity of two points to be the distance between them in this space under some appropriate metric.

The way in which the algorithm decides which of the points from the training set are similar enough to be considered when choosing the class to predict for a new observation is to pick the *k* closest data points to the new observation, and to take the most common class among these. This is why it is called the k Nearest Neighbours algorithm.

The Algorithm

The algorithm (as described in [1] and [2]) can be summarised as:

1. A positive integer *k* is specified, along with a new sample
2. We select the *k* entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

An Example Using the Applet

First we set up the applet [1] as shown in Figure 1, with 40 points of two different colours in opposite corners of the screen, and 20 points of random for each colour. Now, these points will

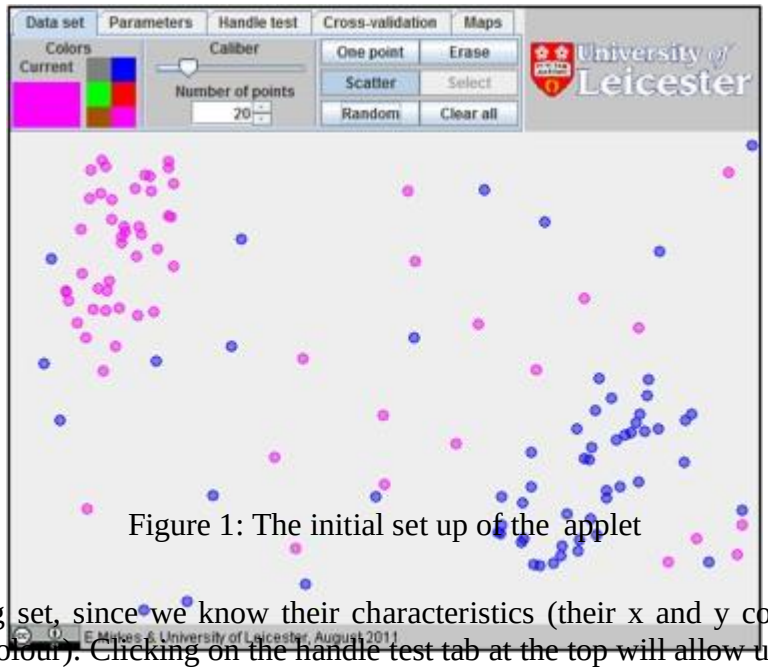


Figure 1: The initial set up of the applet

make up our training set, since we know their characteristics (their x and y coordinates) and their classification (their colour). Clicking on the handle test tab at the top will allow us to place a new test point on the screen by clicking anywhere, as shown in Figure 2. The applet finds an highlights with a star the 3 nearest neighbours of this point. Then, the most common colour among these nearest neighbours is used as the colour of the test point (indicated by the large circle).

Similarly, clicking on the map tab allows us to plot the map generated by the data, as shown in Figure 3. This colours each point on the screen by the colour which a test point at that

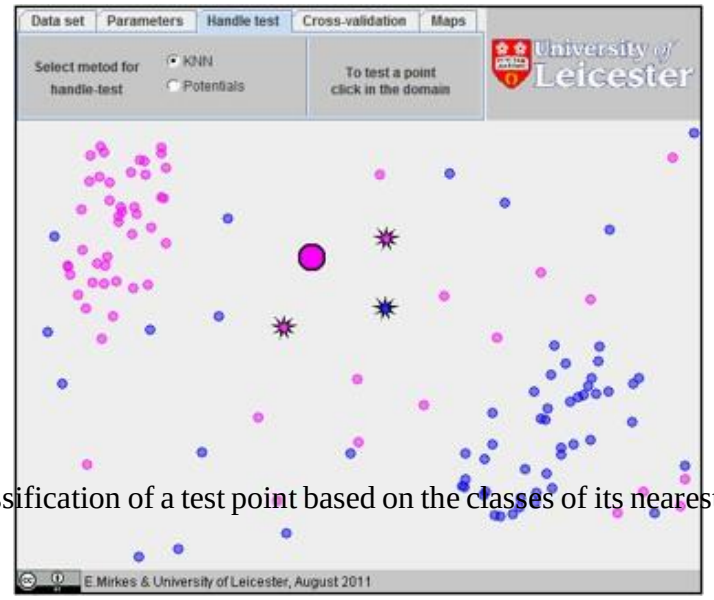


Figure 2: The classification of a test point based on the classes of its nearest neighbours

position would be coloured, so essentially gives us all the results from testing all possible test points.

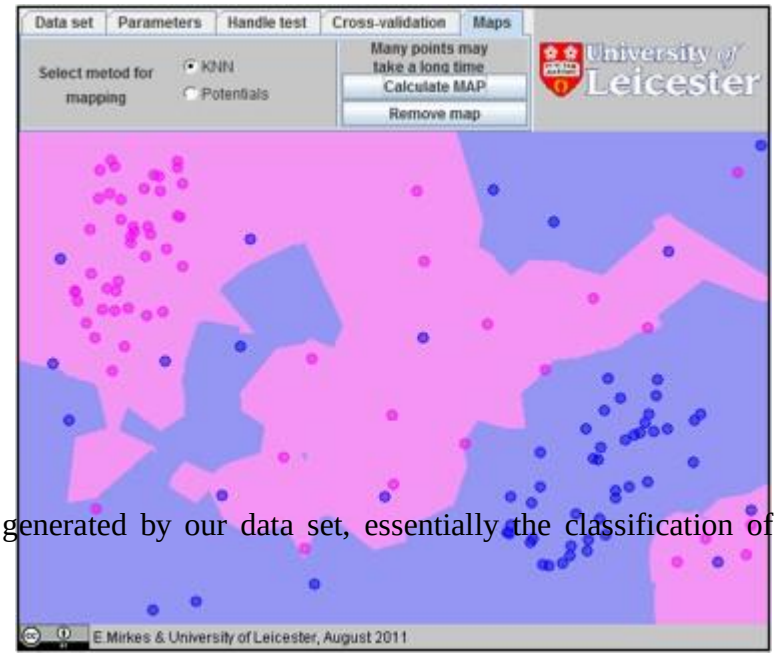


Figure 3: The map generated by our data set, essentially the classification of a test point at any location

This can be particularly useful for seeing how the algorithm's classification of a point at a certain location would compare to how we might do it by eye. For example, notice the large band of blue beside the pink cluster, and the fact that the lower right hand corner got coloured pink. This goes very much against how we would classify a point in either of those locations by eye, where we might be tempted to just split the screen down the middle diagonally, saying anything above the line was pink and anything below the line was blue.

However, if we create the map of a dataset without any noise (as shown in Figure 4), we can see that there is a much clearer division between where points should be blue and pink, and it follows the intuitive line much more closely.

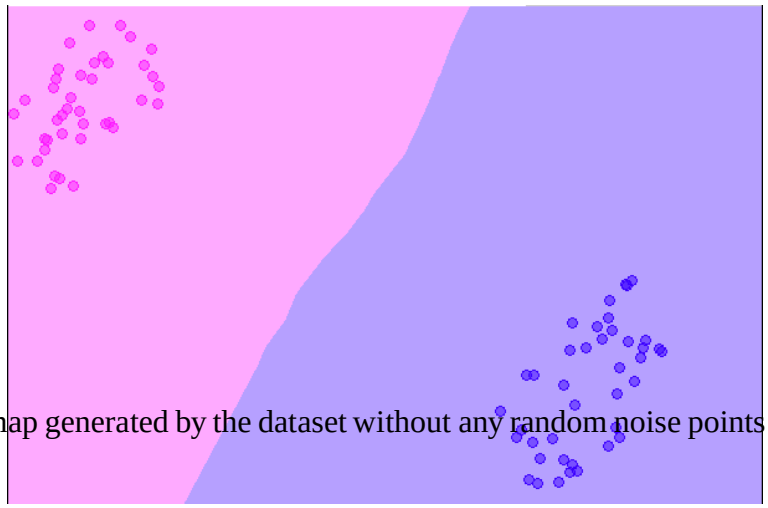


Figure 4: The map generated by the dataset without any random noise points added

The reason for this is not too hard to see. The problem is that the algorithm considers all points equally, whether they are part of a main cluster, or just a random noise point, so even just having a few unusual points can make the results very different.

However, in reality, we certainly expect to get some unusual points in our sample, so we would like to find a way to make the algorithm more robust. With a view to finding a way to make the results look more like the map in Figure 4, we first add in 20 random points from each colour, going back to where we were started. Plotting the map at this stage will produce something fairly unintuitive, like Figure 3, with seemingly random sections mapped to different colours. Instead, we try changing the value of *k*. This is achieved using the 'Parameters' tab at the top. Clearly, when choosing a new value for *k*, we do not want to pick an even number, since we could find ourselves in the awkward position of having an equal number of nearest neighbours of each colour. Instead, we pick an odd number, say 5, and plot the map. We find that as we increase the value of *k*, the map gets closer and closer to the map produced when we had no random points, as shown in Figure 5.

The reason why this happens is quite simple. If we assume that the points in the two clusters will be more dense than the random noise points, it makes sense that when we consider a large value of nearest neighbours, the influence of the noise points in rapidly reduced. However, the problem with this approach is that increasing the value of *k* means that the algorithm takes a long time to run, so if we consider a much bigger database with thousands of data points, and a lot more noise, we will need a higher value of k, and the process will take longer to run anyway.

An alternative method for dealing with noise in our results is to use cross validation. This determines whether each data point would be classified in the class it is in if it were added to the data set later. This is achieved by removing it from the training set, and running the kNN algorithm to predict a class for it. If this class matches, then the point is correct, otherwise it is deemed to be a cross-validation error. To investigate the effects of adding random noise points to the training set, we can use the applet. Setting it up as before, with two clusters in opposite corners, we can use the "Cross Validation" feature to run a cross validation on the data in our training set. This was done for 6 different data sets, and the number of noise points

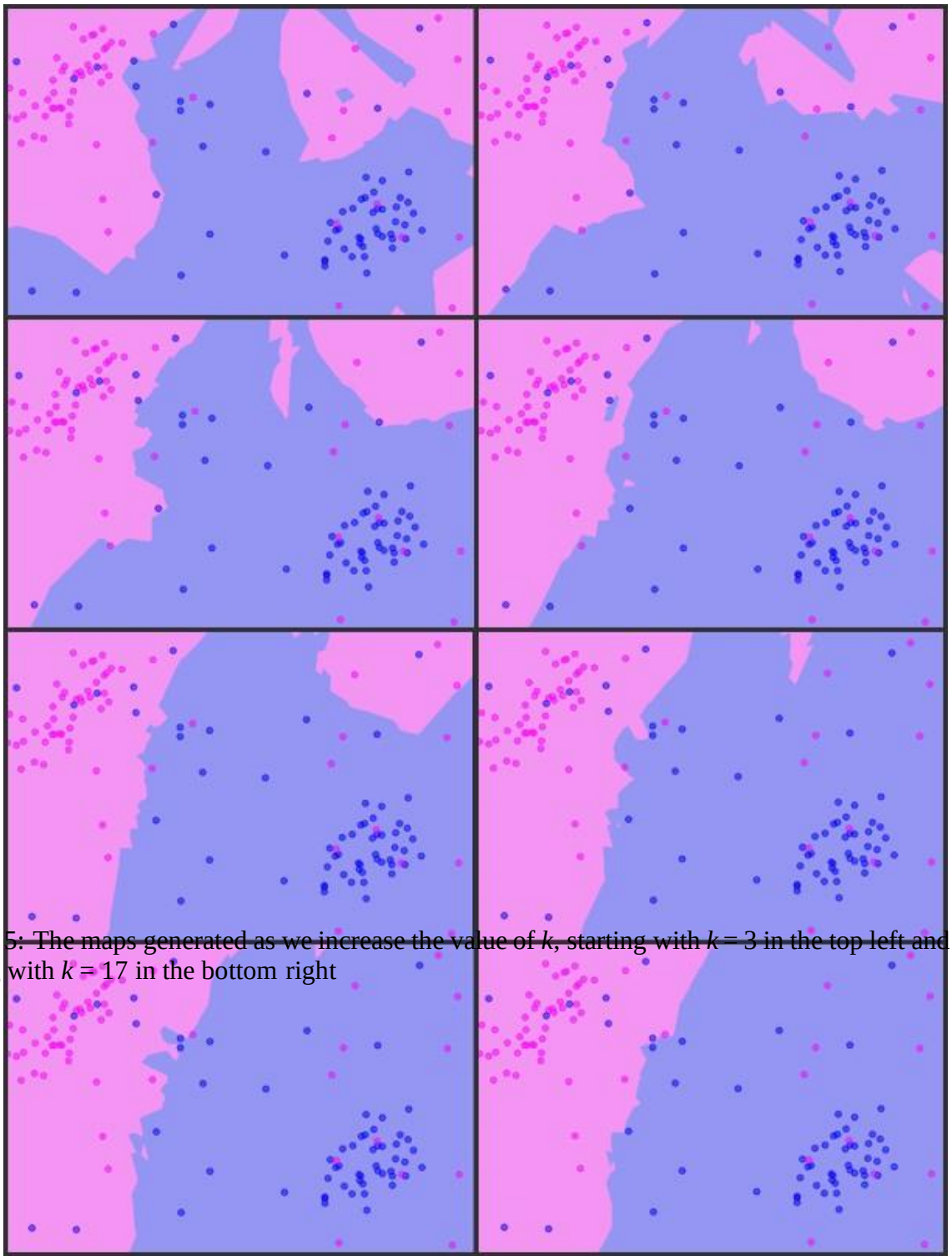


Figure 5: The maps generated as we increase the value of k, starting with $k = 3$ in the top left and ending with $k = 17$ in the bottom right

in each one was increased. These results were averaged, and the average percentage of points misclassified is shown in the table:

Number of Random Data Points	Average Percentage of Cross-Validation Errors
0	0
20	17.5
40	25
60	29.66666667
80	33.58333333
100	34.66666667
120	35.83333333
140	37.25
160	38.5

For clarity, these data are also presented in the graph in Figure 6. As can be seen, the number of mis-classifications increases as we increase the number of random noise points present in our training data. This as we would expect, since the random noise points should show up as being unexpected. Indeed, if we extrapolate the data using the general trend of the graph, we can expect that the percentage of mis-classifications would increase to approximately 50% as we continue increasing the number of random noise points. This agrees with how we would intuitively expect the cross-validation to behave on a completely random data set.

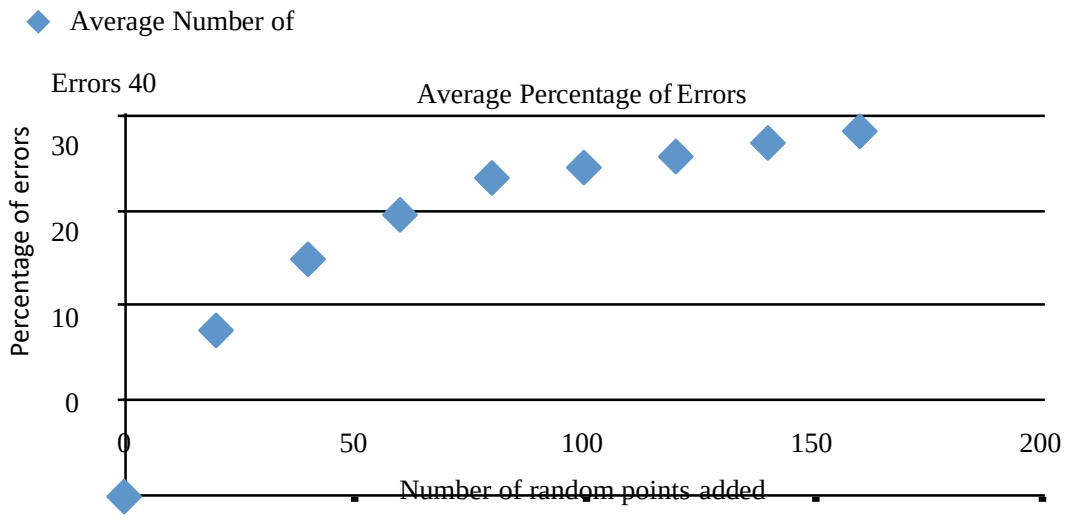


Figure 6: A graph demonstrating the effect of increasing the number of random noise points in the data set on the number of mis-classifications

It is also found that increasing the size of the training set makes classification of a new point take much longer, due simply to the number of comparisons required. This speed problem is one of the main issues with the kNN algorithm, alongside the need to find some way of comparing data of strange types (for instance, trying to compare people's favourite quotations to decide their political preferences would require us to take all sorts of metrics about the actual quotation such as information about the author, about the work it is part of, etc, that it quickly becomes very difficult for a computer, while still relatively easy for a person).

To solve the first of these problems, we can use data reduction methods to reduce the number of comparisons necessary.

3 Condensed Nearest Neighbour Data Reduction

Condensed nearest neighbour data reduction is used to summarise our training set, finding just the most important observations, which will be used to classify any new observation. This can drastically reduce the number of comparisons we need to make in order to classify a new observation, while only reducing the accuracy slightly.

The way the algorithm works is to divide the data points into 3 different types (as described in [1]):

1. **Outliers:** points which would not be recognised as the correct type if added to the database later
2. **Prototypes:** the minimum set of points required in the training set for all the other non-outlier points to be correctly recognised
3. **Absorbed points:** points which are not outliers, and would be correctly recognised based just on the set of prototype points

Then, we only need to compare new observations to the prototype points. The

algorithm to do this can be summarised as:

1. Go through the training set, removing each point in turn, and checking whether it is recognised as the correct class or not
   - If it is, then put it back in the set
   - If not, then it is an outlier, and should not be put back
2. Make a new database, and add a random point.
3. Pick any point from the original set, and see if it is recognised as the correct class based on the points in the new database, using kNN with $k = 1$
   - If it is, then it is an absorbed point, and can be left out of the new database
   - If not, then it should be removed from the original set, and added to the new database of prototypes
4. Proceed through the original set like this
5. Repeat steps 3 and 4 until no new prototypes are added

As described in [3] and [4], this algorithm can take a long time to run, since it has to keep repeating. The problem of trying to improve this algorithm to make it faster can, however, we tackled by using extended techniques, such as that described in [4]. However, once it has been run, the kNN algorithm will be much faster.

The applet can also handle CNN data reduction, and an example of its use is given in the slides.

CNN is also quite affected by noise in the training set. In order to investigate this, three data sets were produced as usual, and for each one the number of noise points was increased, and CNN was run, recording the percentage of points assigned to each type. The results from averaging these three different data sets are shown in the table below:

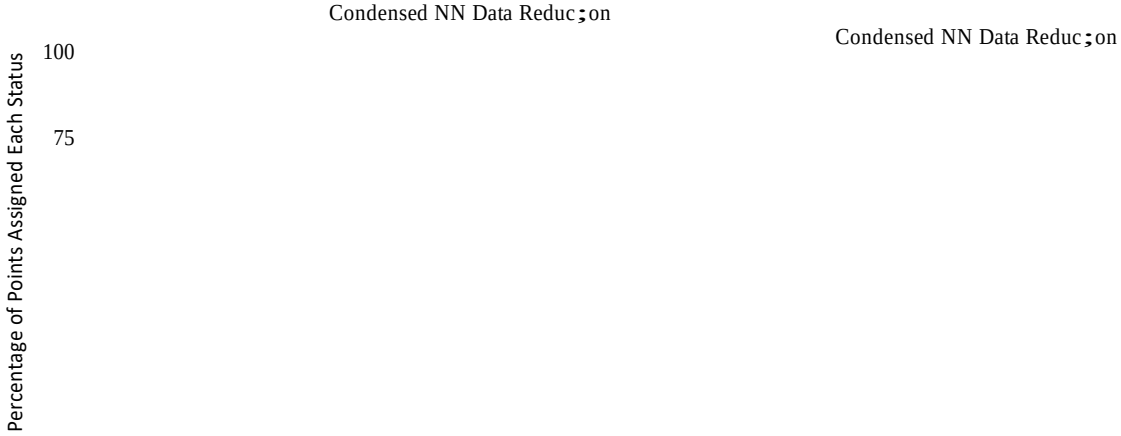
These data are also presented as a graph in Figure 7 for clarity. As can easily be seen, increasing the number of random noise points affected the results of the CNN algorithm in three main

Number of Random Data Points	Average Percentage of Cross8Valida;on	Average Percentage of Prototype Points	Average Percentage of Outlier Points	Average Percentage of Absorbed Points
0	0	3	0	98
20	17.5	16.16667	5.5	78.33333
40	25	22	9.333333	69
60	29.66666667	25.66667	8.833333	65.5
80	33.58333333	31.16667	10	59
100	34.66666667	31.16667	11.16667	57.66667
120	35.83333333	33.66667	10.5	55.66667
140	37.25	32.33333	12.33333	55.16667
160	38.5	37	11.5	52.16667

ways:

1. The percentage of points classed as outliers increased dramatically
2. The percentage of points classed as absorbed decreased
3. The percentage of points classed as prototypes increased slightly

All of these are as would be expected. The percentage of outliers increases because there are increasingly many noise points of the other colour in each cluster, which will lead them to be mis-classified. The percentage of points deemed to be prototypes increases because our data set now has a much more complex structure once we have included all these random noise points. The percentage of absorbed points therefore must decrease since the other two types are increasing (and the three are mutually exclusive).



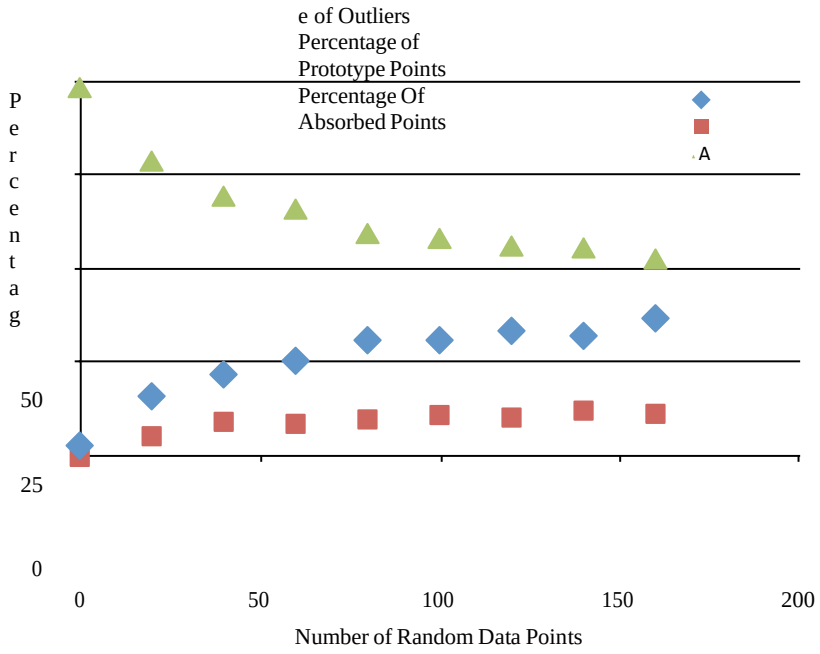


Figure 7: A graph showing the effects of increasing the number of random noise points in the training data on the percentage of points assigned each of the three primitive types by the CNN algorithm.

References

[1] E. Mirkes, *KNN and Potential Energy (Applet)*. University of Leicester. Available: http://www.math.le.ac.uk/people/ag153/homepage/KNN/KNN3.html, 2011.

[2] L. Kozma, *k Nearest Neighbours Algorithm*. Helsinki University of Technology, Available: http://www.lkozma.net/knn2.pdf, 2008.

[3] N. Bhatia et al, *Survey of Nearest Neighbor Techniques*. International Journal of Computer Science and Information Security, Vol. 8, No. 2, 2010.

[4] F. Anguilli, *Fast Condensed Nearest Neighbor Rule*. Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.

IMPLEMENTATION USING R

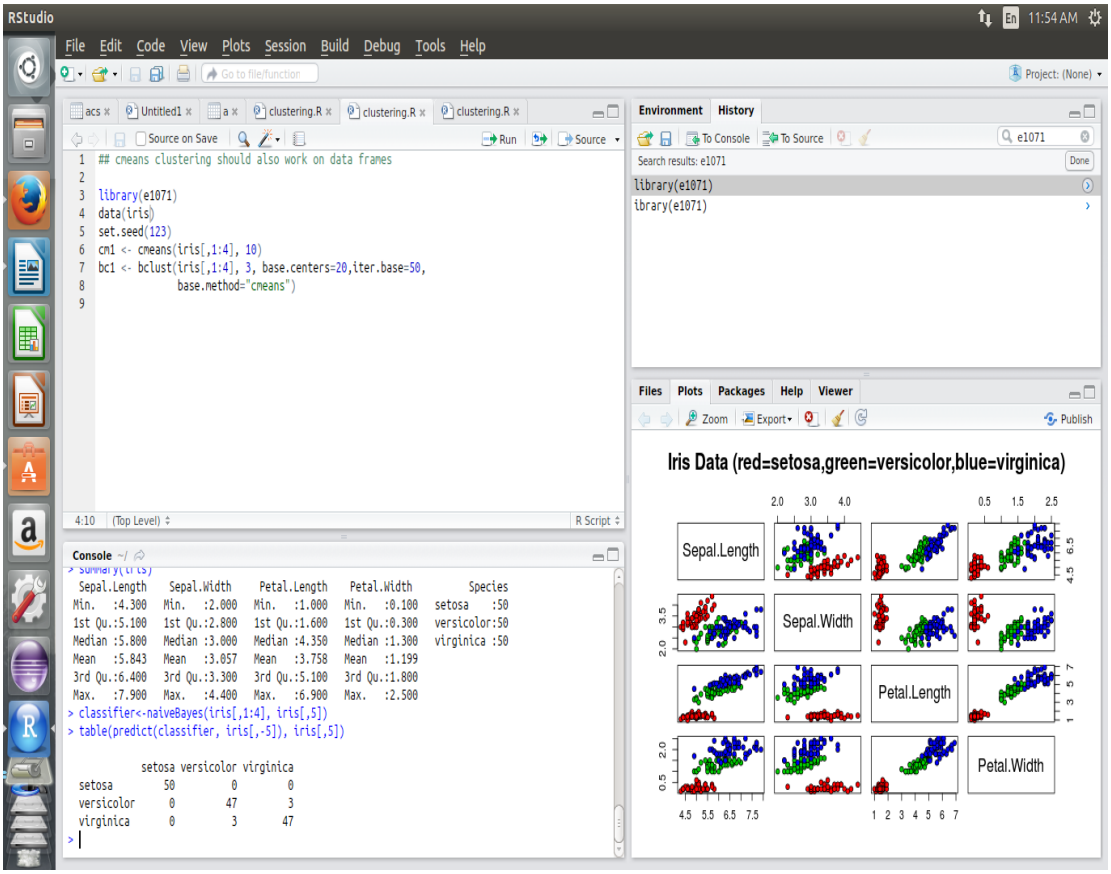
- Download Package e1071
- Add e1071 package in R

Execute the following commands

>e1071

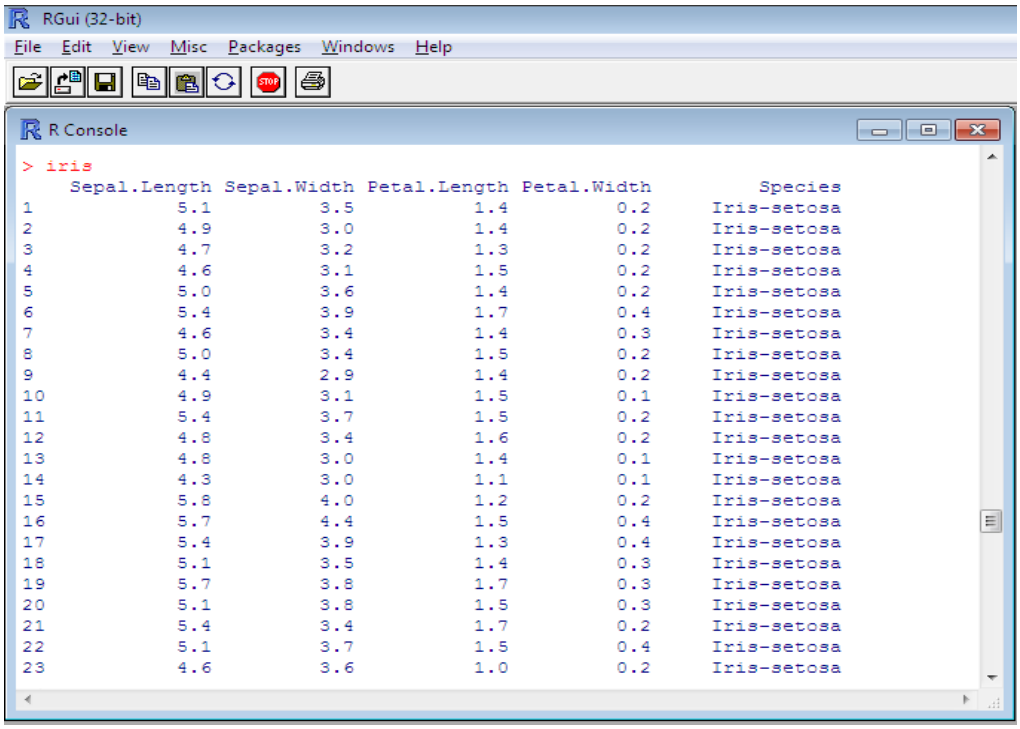
>library(e1071)

>naïve bays



Implementation using R

Database : IRIS




```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width          Species
1            5.1         3.5          1.4         0.2      Iris-setosa
2            4.9         3.0          1.4         0.2      Iris-setosa
3            4.7         3.2          1.3         0.2      Iris-setosa
4            4.6         3.1          1.5         0.2      Iris-setosa
5            5.0         3.6          1.4         0.2      Iris-setosa
6            5.4         3.9          1.7         0.4      Iris-setosa
7            4.6         3.4          1.4         0.3      Iris-setosa
8            5.0         3.4          1.5         0.2      Iris-setosa
9            4.4         2.9          1.4         0.2      Iris-setosa
10           4.9         3.1          1.5         0.1      Iris-setosa
11           5.4         3.7          1.5         0.2      Iris-setosa
12           4.8         3.4          1.6         0.2      Iris-setosa
13           4.8         3.0          1.4         0.1      Iris-setosa
14           4.3         3.0          1.1         0.1      Iris-setosa
15           5.8         4.0          1.2         0.2      Iris-setosa
16           5.7         4.4          1.5         0.4      Iris-setosa
17           5.4         3.9          1.3         0.4      Iris-setosa
18           5.1         3.5          1.4         0.3      Iris-setosa
19           5.7         3.8          1.7         0.3      Iris-setosa
20           5.1         3.8          1.5         0.3      Iris-setosa
21           5.4         3.4          1.7         0.2      Iris-setosa
22           5.1         3.7          1.5         0.4      Iris-setosa
23           4.6         3.6          1.0         0.2      Iris-setosa
```

- You can make scatterplots with the **ggvis package**, for example. You first need to load the ggvis

```
>library(ggvis)
>iris %>% ggvis(~Sepal.Length, ~Sepal.Width, fill = ~Species)
  %>% layer_points()
```

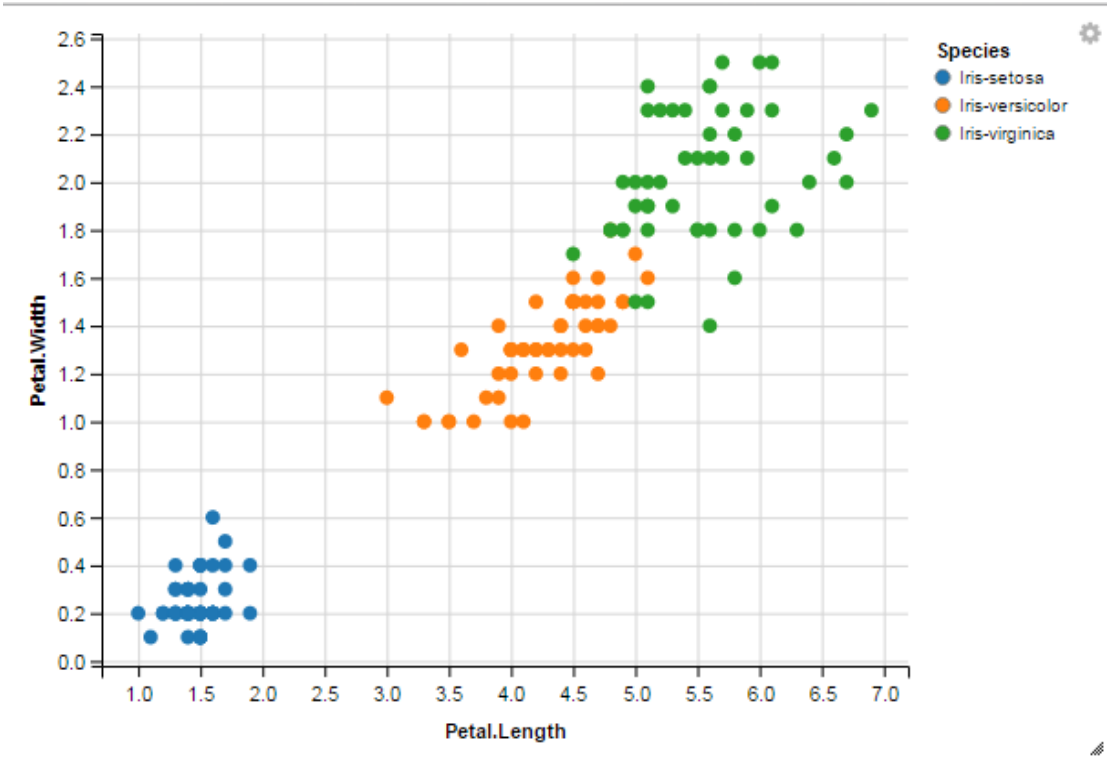
%>% is the pipe function

layer_points() is the function that creates mark objects on the graph

This command will generate a scatter graph

```
> iris %>% ggvis(~Petal.Length, ~Petal.Width, fill
  = ~Species) %>% layer_points()
```

The scatter plot that maps the petal length and the petal width.



A Quick look at the species attribute through tells you that the division of the species of the flowers is 50-50-50.

```
> table(iris$Species)

    Iris-setosa Iris-versicolor  Iris-virginica
             50              50              50
```

- R gives you the opportunity to go more in-depth with the summary() function.
- This will give you the minimum value, first quantile, median, mean, third quantile and maximum value of the data set Iris for numeric data types.

> summary(iris)

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
```

- You can also refine your summary overview by adding specific attributes to the command that was presented above:

```
> summary(iris[c("Petal.Width", "Sepal.Width")])
```

```
> summary(iris[c("Petal.Width", "Sepal.Width")])
  Petal.Width     Sepal.Width
 Min.   :0.100   Min.   :2.000
 1st Qu.:0.300   1st Qu.:2.800
 Median :1.300   Median :3.000
 Mean   :1.199   Mean   :3.054
 3rd Qu.:1.800   3rd Qu.:3.300
 Max.   :2.500   Max.   :4.400
>
```

- You can perform feature normalization, for example, by first making your own normalize function:

```
normalize <- function(x) {
        num <- x - min(x)
        denom <- max(x) - min(x)
                return (num/denom)
            }

          >YourNormalizedDataSet<-
))          as.data.frame(lapply(YourDataSet, normalize)
```

- For the Iris dataset, you would have applied the normalize argument on the four numerical attributes of the Iris data set (Sepal.Length, Sepal.Width, Petal.Length, Petal.Widthand put the results in a data frame:

```
> iris_norm <- as.data.frame(lapply(iris[1:4], normalize))
```

```
> normalize <- function(x) {
+ num <- x - min(x)
+ denom <- max(x) - min(x)
+ return (num/denom)
+ }
> iris_norm <- as.data.frame(lapply(iris[1:4], normalize))
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> summary(iris_norm)
  Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.08333
 Median :0.4167   Median :0.4167   Median :0.5678   Median :0.50000
 Mean   :0.4287   Mean   :0.4392   Mean   :0.4676   Mean   :0.45778
 3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.70833
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
>
```

- The major advantage of setting a seed is that you can get the same sequence of random numbers whenever you supply the same seed in the random number generator.

```
> set.seed(1234)
```

- The assignment of the elements is subject to probability weights of 0.67 and 0.33.

```
> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))
```

TRAINING AND TESTING DATA

```
> iris.training <- iris[ind==1, 1:4]
> iris.test <- iris[ind==2, 1:4]
```

Building Your Classifier

- After all these preparation steps, you have made sure that all your known (training) data is stored.
- No actual model or learning was performed up until this moment.
- Now, you want to find the ***k*** nearest neighbors of your training set.

- To build your classifier, you need to take the knn() function and simply add some arguments to it, like here:

```
> iris_pred <- knn(train = iris.training, test = iris.test, cl
= iris.trainLabels, k=3)
```

- You store into iris_pred the knn() function that takes as arguments the training set, the test set, the train labels and the amount of neighbours you want to find with this algorithm.
- The result of this function is a factor vector with the predicted classes for each row of the test data.

- You can retrieve the result of the knn() function by typing in the following command:

```
> iris_pred
```

- The result of this command is the factor vector with the predicted classes for each row of the test data.

```
> iris_pred
 [1] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa
 [7] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa
[13] Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor
[19] Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor
[25] Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-versicolor Iris-virginica
[31] Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica
[37] Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica
Levels: Iris-setosa Iris-versicolor Iris-virginica
>
```

FAQs :

1. **How to use KL-divergence in naive bayes classifier to weight features?**
2. **How to use log probabilities for Gaussian Naive Bayes?**
3. **How can I get feature importance for Gaussian Naive Bayes classifier?**

PRACTISE ASSIGNMENTS

1. Perform above on different input dataset.

Assignment 4 Unsupervised Learning

AIM : Unsupervised Learning: Implement K-Means Clustering and Hierarchical clustering on proper data set of your choice. Compare their Convergence.

OBJECTIVE :

To study

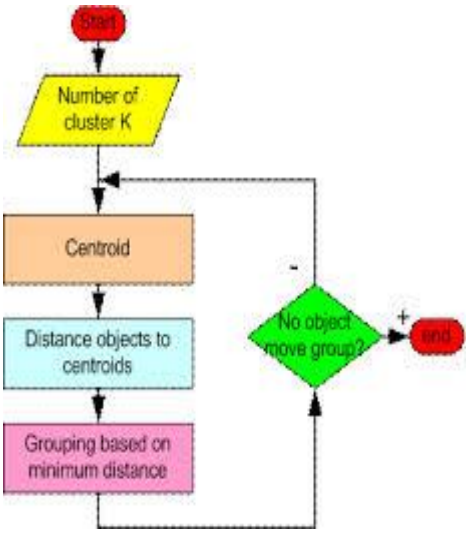
- To understand the concept of clustering.
- To implement K-means Clustering algorithm
- To implement Hierarchical clustering algorithm
- To find Convergence
- Analyze the above algorithms and verify with execution of program on different datasets

THEORY:

Introduction:

- **Introduction of Kmeans**

  K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

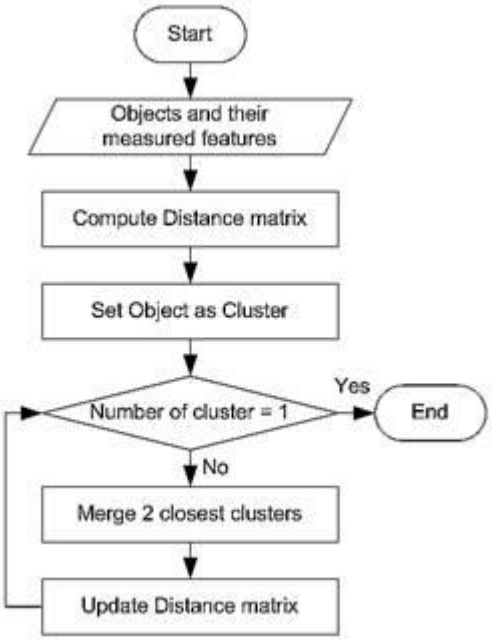


- **Introduction of Hierarchical Clustering**

In data mining and statistics, hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a

hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.



Concept/Working

- **Working of K-Means Clustering**

  The *k*-means clustering algorithm attempts to split a given anonymous data set (a set containing no information as to class identity) into a fixed number (*k*) of cluster.

  Initially *k* number of so called *centroids* are chosen. A *centroid* is a data point (imaginary or real) at the center of a cluster. In Praat each centroid is an existing data point in the given input data set, picked at random, such that all *centroids* are unique (that is, for all *centroids* c_i and c_j, $c_i \neq c_j$). These *centroids* are used to train a kNN classifier. The resulting classifier is used to classify (using $k = 1$) the data and thereby produce an initial randomized set of clusters. Each *centroid* is thereafter set to the arithmetic mean of the cluster it defines. The process of classification and *centroid* adjustment is repeated until the values of the *centroids* stabilize. The final *centroids* will be used to produce the final classification/clustering of the input data, effectively turning the set of

initially anonymous data points into a set of data points, each with a class identity.

- **Working of Hierarchical Clustering**

Given a set of N items to be clustered, and an NxN distance (or similarity) matrix, the basic process of Johnson's (1967) hierarchical clustering is this:

1. Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.
3. 3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. 4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Advantages/Disadvantages

- **Advantages of K-Means Clustering**.

1) If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.

2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

- **Disadvantages of K-Means Clustering.**

1) Difficult to predict K-Value.

2) With global cluster, it didn't work well.

3) Different initial partitions can result in different final clusters.

4) It does not work well with clusters (in the original data) of Different size and Different density

- **Advantages of Hierarchical Clustering**.

1) Hierarchical Clustering can give different partitioning depending on the level-of-resolution we are looking at.

Hierarchical clustering doesn’t need the number of clusters to be specified.

Disadvantages of Hierarchical Clustering.

Fail to take into account special characteristics of individual clusters, and thus can make incorrect merging decisions when the underlying data does not follow the assumed model, or when noise is present.

1) Hierarchical clustering can be slow (has to make several merge/split decisions).

Software Used

Weka.
RStudio.

INPUT/OUTPUT:

Input : For example: iris.arff dataset is used.
Output : Classified the given dataset.

OPERATIONAL STEPS REQUIRED:

Operational steps for K means clustering using Weka.

1. Select a dataset for example iris.arff.
2. Select option Cluster
3. Choose cluster type: Kmeans Cluster
4. Select cluster mode: Percentage Split.
5. Click on Start.

Operation steps for Hierarchical Clustering in Weka

1. Select a dataset for example iris.arff.
2. Select option Cluster
3. Choose cluster type: Hierarchial Cluster
4. Select cluster mode: Training Set.
5. Click on Start.

Operational steps for K means clustering using R.

In RStudio:

Run the following code in console in RStudio:

```
> dim(iris)
> names(iris)
> str(iris)
> attributes(iris)
> iris[1:5,]
> newiris <- iris
> newiris$Species <- NULL
> (kc <- kmeans(newiris, 3))
> table(iris$Species, kc$cluster)
> plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
> points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=8, cex=2)
```

Operation steps for Hierarchical Clustering Using R

In RStudio:

Install and Update Library:

Go to Packages - Click on "Install"
You will get "Install Packages" Window.
Install from: Package Archive File(tar.gz)
Package archive: (Browse the path where you have stored the e1071_1.6-4.tar.gz)
Click Install.
In Packages - Tick the e1701 in user library. Click on Update. Click on OK.

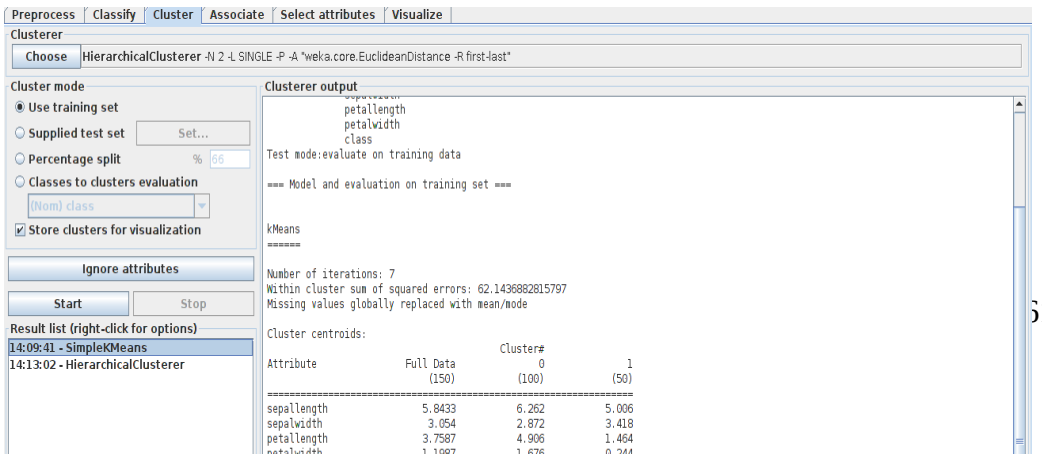
Write following in the Rscript (File - New File – Rscript) and Save it as Hirarchical.R:

```
idx <- sample(1:dim(iris)[1], 40)
irisSample <- iris[idx,]
irisSample$Species <- NULL
hc <- hclust(dist(irisSample), method="ave")
plot(hc, hang = -1, labels=iris$Species[idx])
```

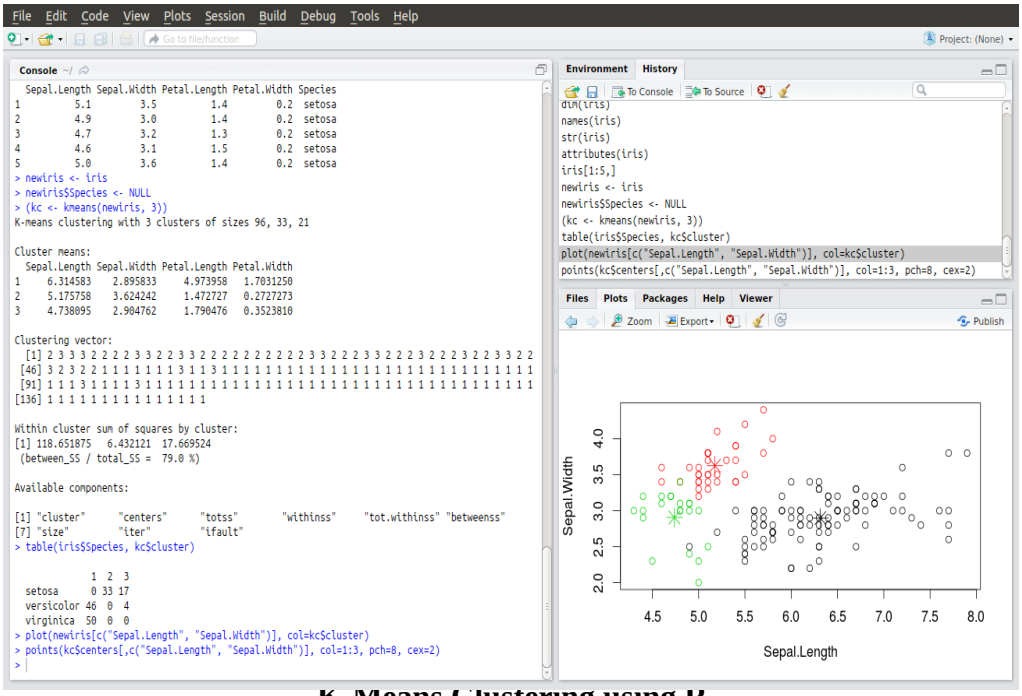
Run it (Run Icon on Tool Bar).
Run same code given above in console in RStudio.

CONCLUSION:

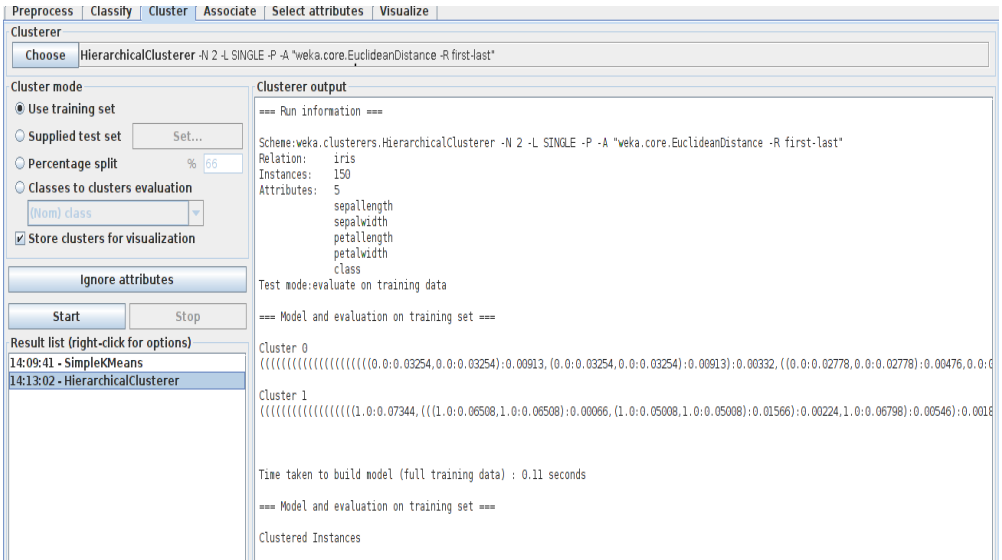
Implemented K-Means Clustering and Hierarchical clustering on proper data set.
As the number of records increase the performance of hierarchical algorithm goes decreasing and time for execution increased.
K-means algorithm also increases its time of execution but as compared to hierarchical algorithm its performance is better.
Hierarchical algorithm shows more quality as compared to k-mean algorithm.
As a general conclusion, k-mean algorithm is good for large dataset and hierarchical is good for small datasets.



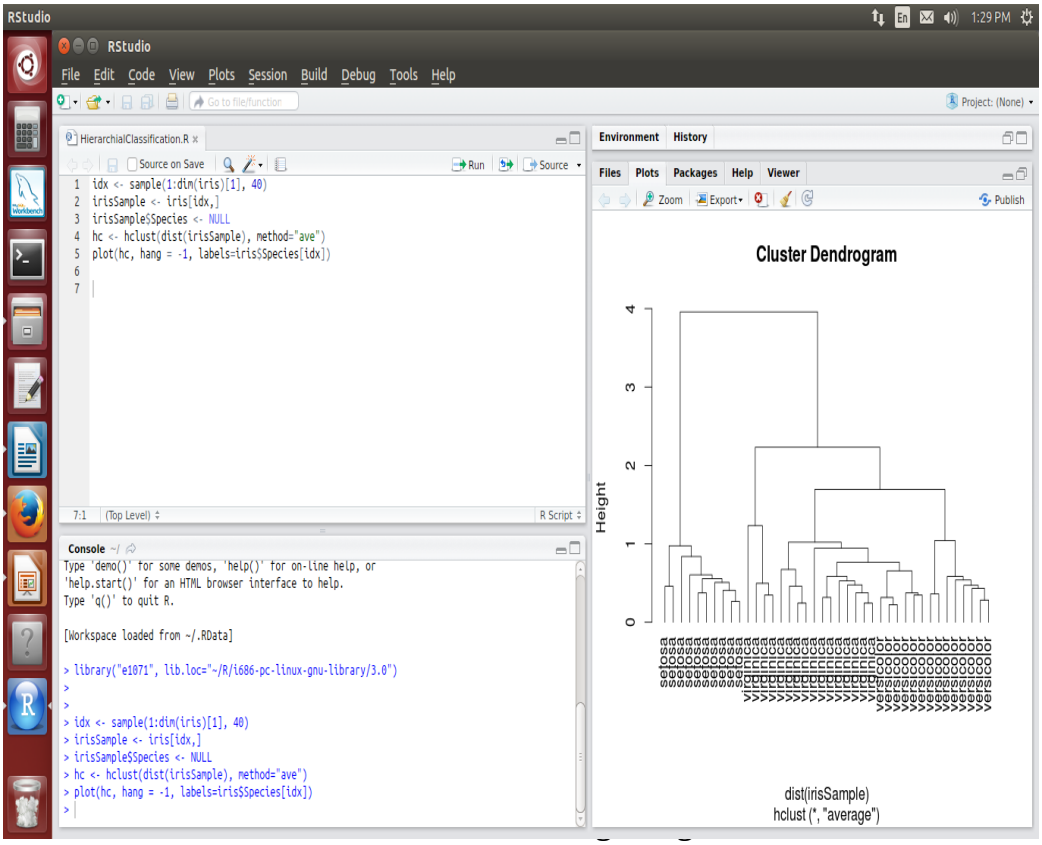
K-Means Clustering using WEKA



K-Means Clustering using R



Hierarchical Clustering using WEKA



FAQs :

1. **How to interpret the numeric values for "height" using wards clustering method**

2. **Which unsupervised classification method to use next if hierarchical clustering gave bad results?**

3. **Difference between clustering functions in R**

4. **How to cluster rows of binary data?**

5. **Difference between PROC VARCLUS and factor analysis**

6. **Cluster of Cluster analysis across correlated longitudinal data**

7. How do I make a quick Hierarchical Cluster visualization?
8. How do I adjust the view, add labels, or standardize my clustering data?
9. Why do I run out of memory on >~30k elements?

PRACTISE ASSIGNMENTS:

1. Perform above on different input dataset.

Assignment 5 Dimensionality Reduction

AIM **:** Principal Component Analysis-
Finding Principal Components, Variance and Standard Deviation calculations of principal components.

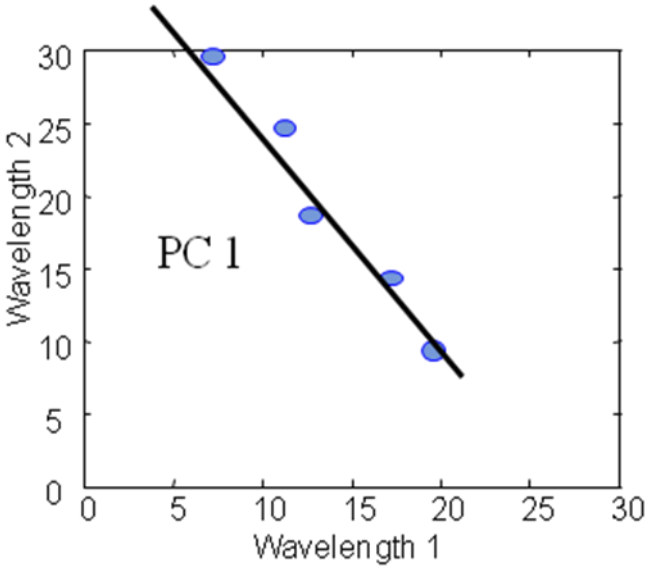
OBJECTIVE:

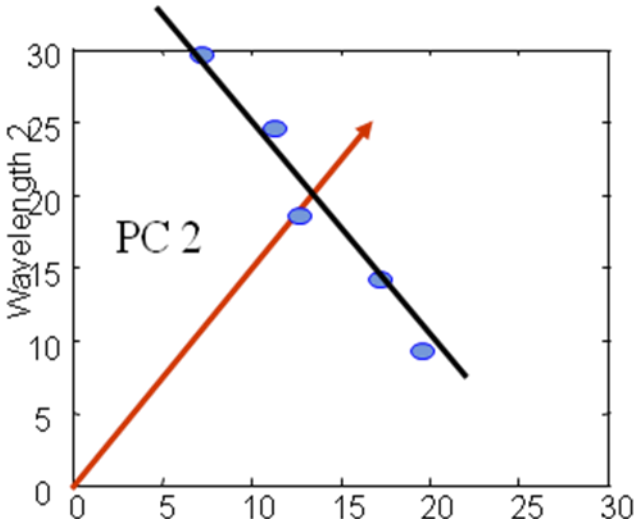
- To understand the concept of Dimensionality Reduction
- To understand what is Principal Component
- To understand what is Variance
- To understand what is Standard Deviation
- To implement method for calculating the above
- Analyze the above algorithms and verify with execution of program on different datasets

THEORY:

- Principal component analysis (PCA) refers to the process by which principal components are computed, and the subsequent use of these components in understanding the data.
- PCA is an unsupervised learning approach. Since it involves only a set of features X1, X2, . . . , Xp and no associated response Y. Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization (visualization of the observations or visualization of the variables).

- All principal components (PCs) start at the origin of the ordinate axes.
- First PC is direction of maximum variance from origin
- Subsequent PCs are orthogonal to 1st PC and describe maximum residual variance



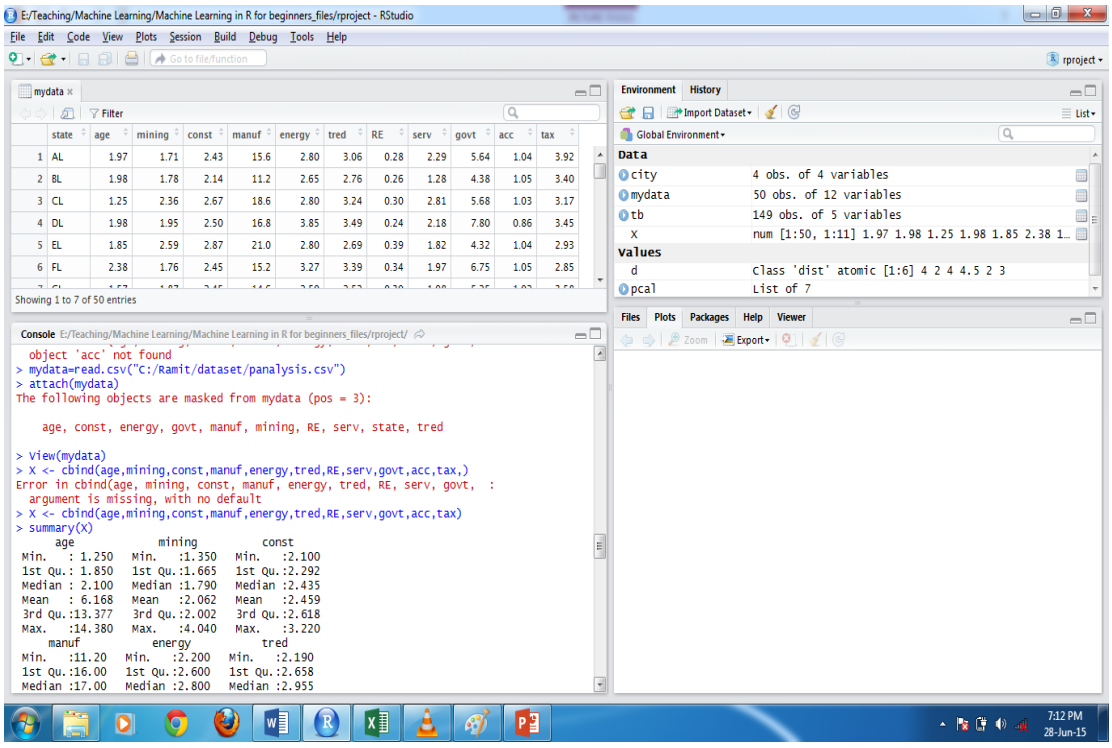


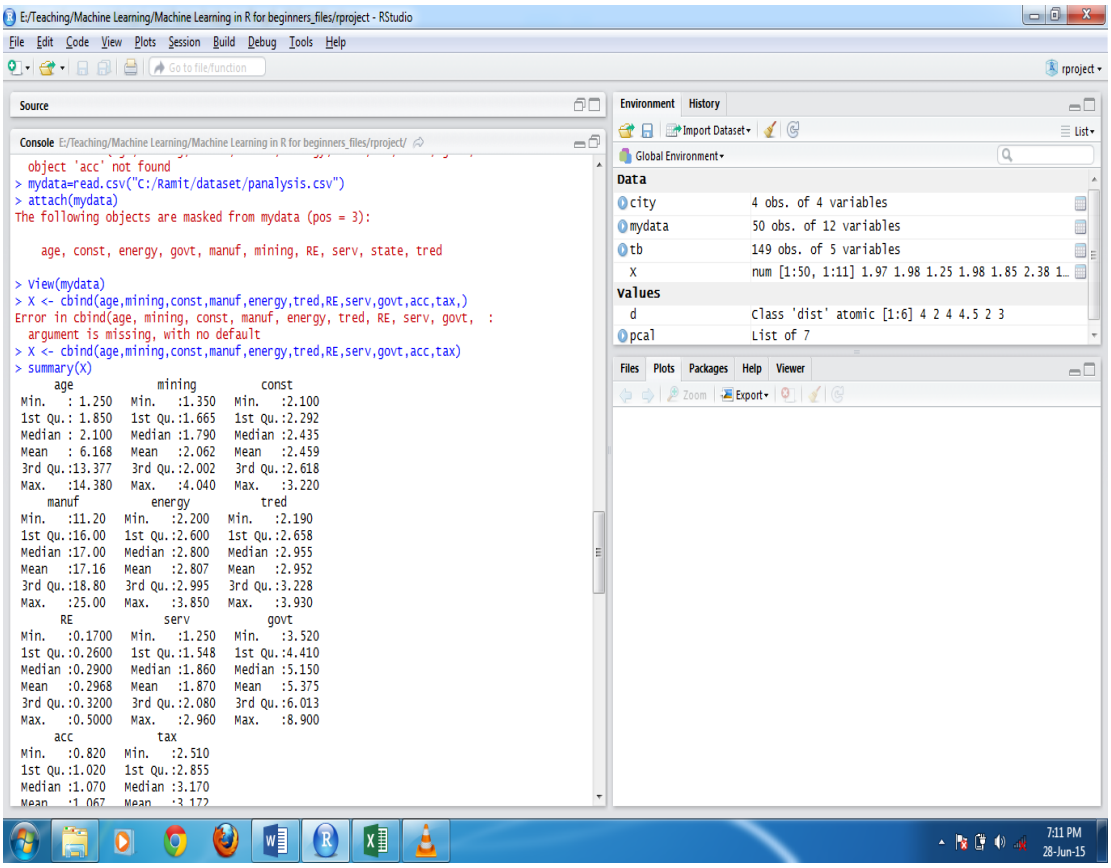
- Covariance Matrix:
  - Variables must be in same units
  - Emphasizes variables with most variance
  - Mean eigenvalue ≠1.0

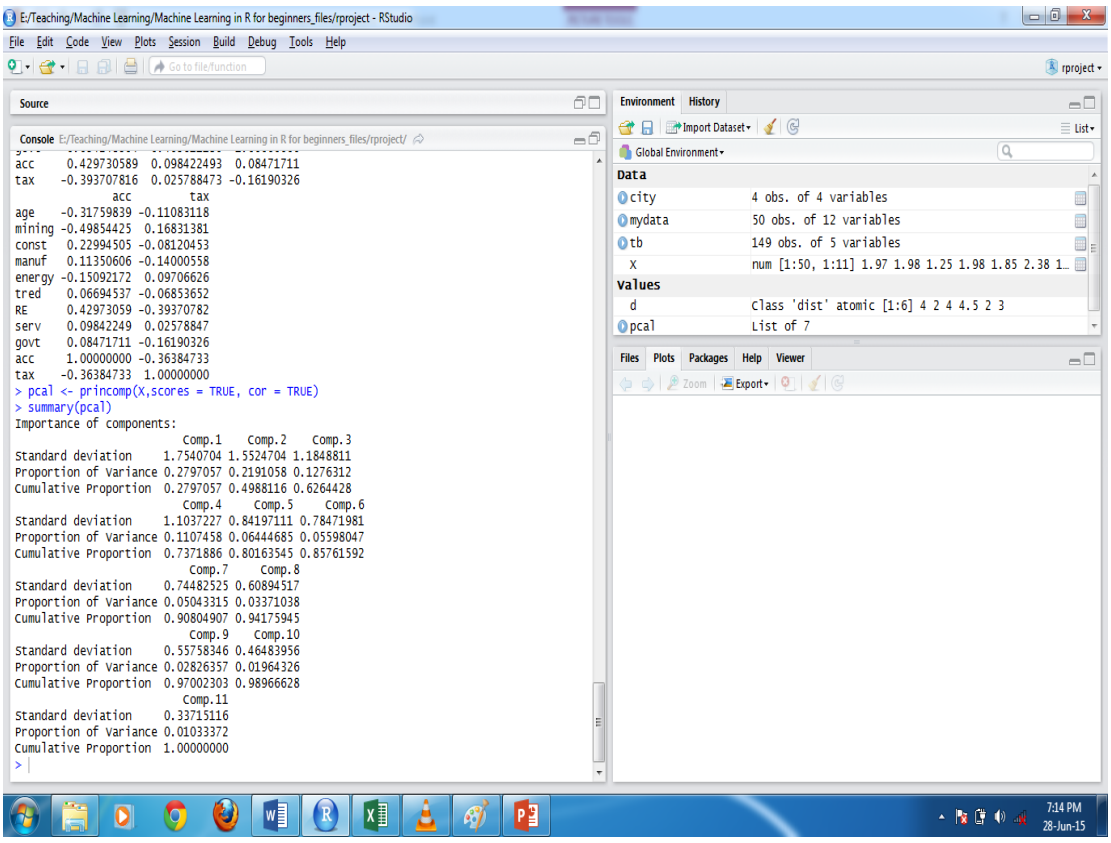
- Correlation Matrix:
  - Variables are standardized (mean 0.0, SD 1.0)
  - Variables can be in different units
  - All variables have same impact on analysis
  - Mean eigenvalue = 1.0

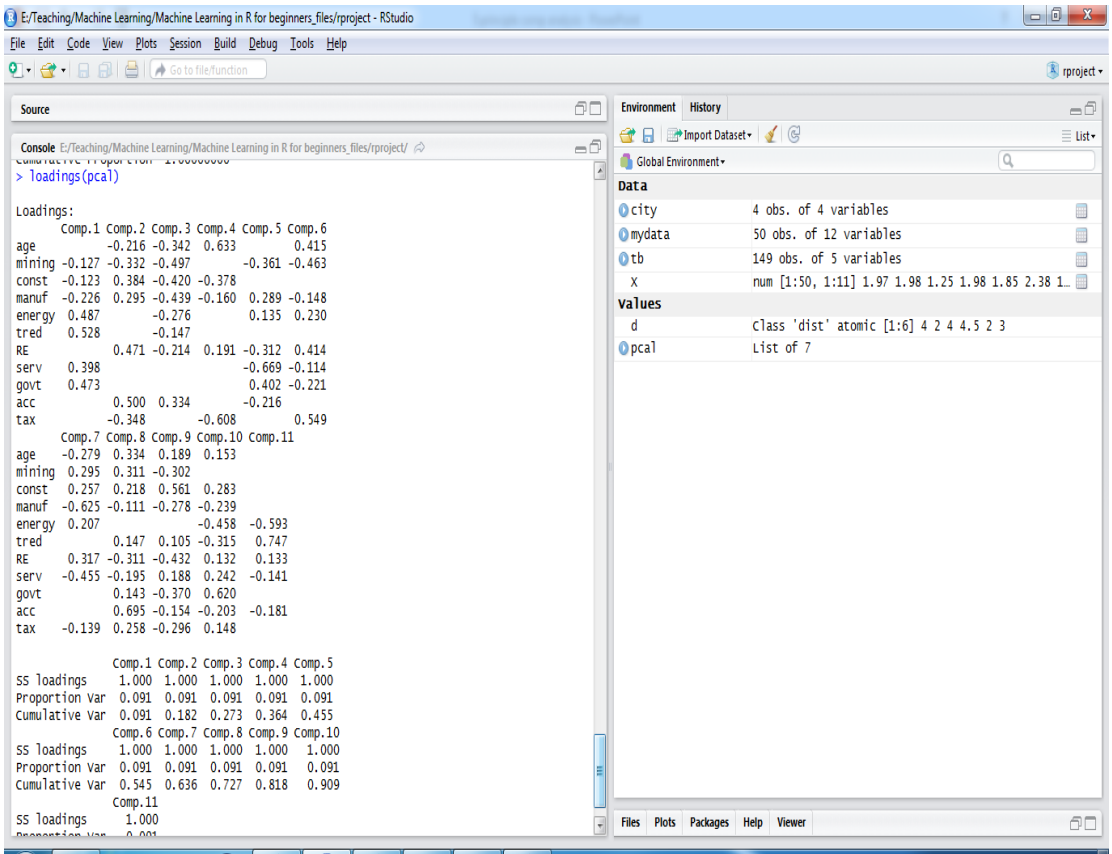
Steps to perform PCA

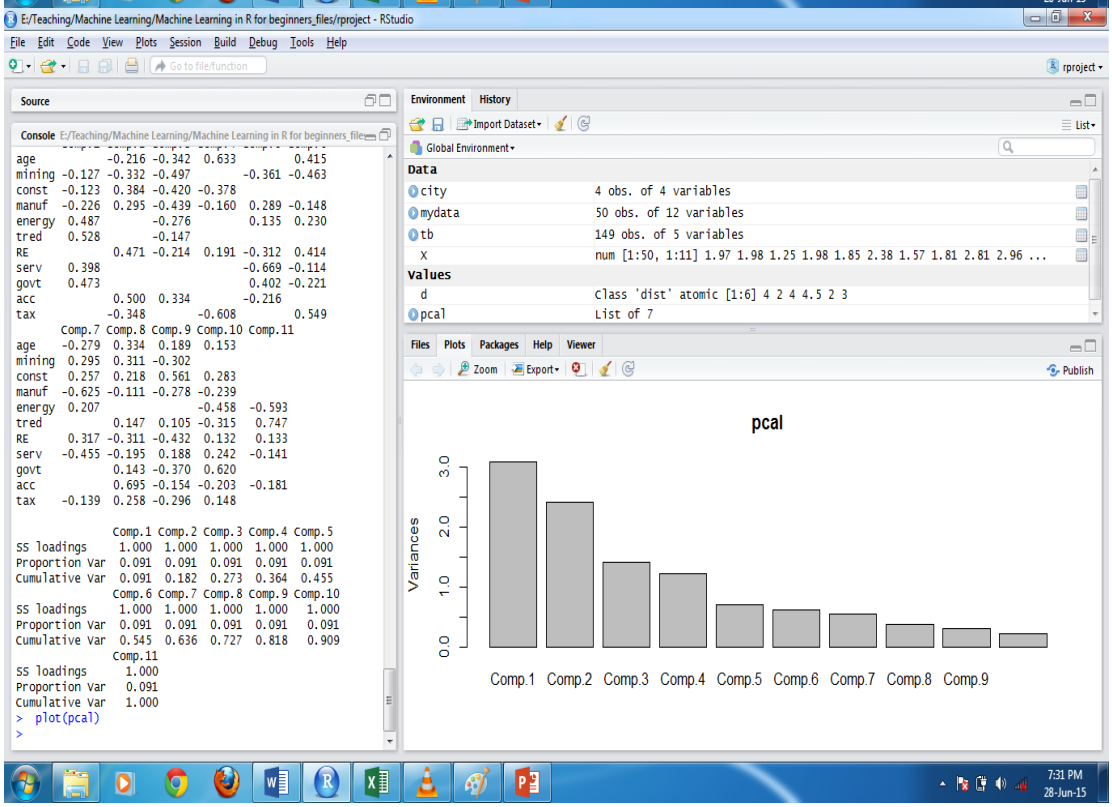
```
mydata<- read.csv("C:/Econometrics/Data/pca_gsp.csv")
attach(mydata)

# Define variables
X <- cbind(Ag, Mining, Constr, Manuf, Manuf_nd, Transp, Comm, Energy, TradeW, Trade

# Descriptive statistics
summary(X)
cor(X)

# Principal component analysis
pca1 <- princomp(X, scores=TRUE, cor=TRUE)
summary(pca1)

# Loadings of principal components
loadings(pca1)
#pca1$loadings

# Scree plot of eigenvalues
plot(pca1)
screeplot(pca1, type="line", main="Scree Plot")

# Biplot of score variables
biplot(pca1)

# Scores of the components
pca1$scores[1:10,]
```

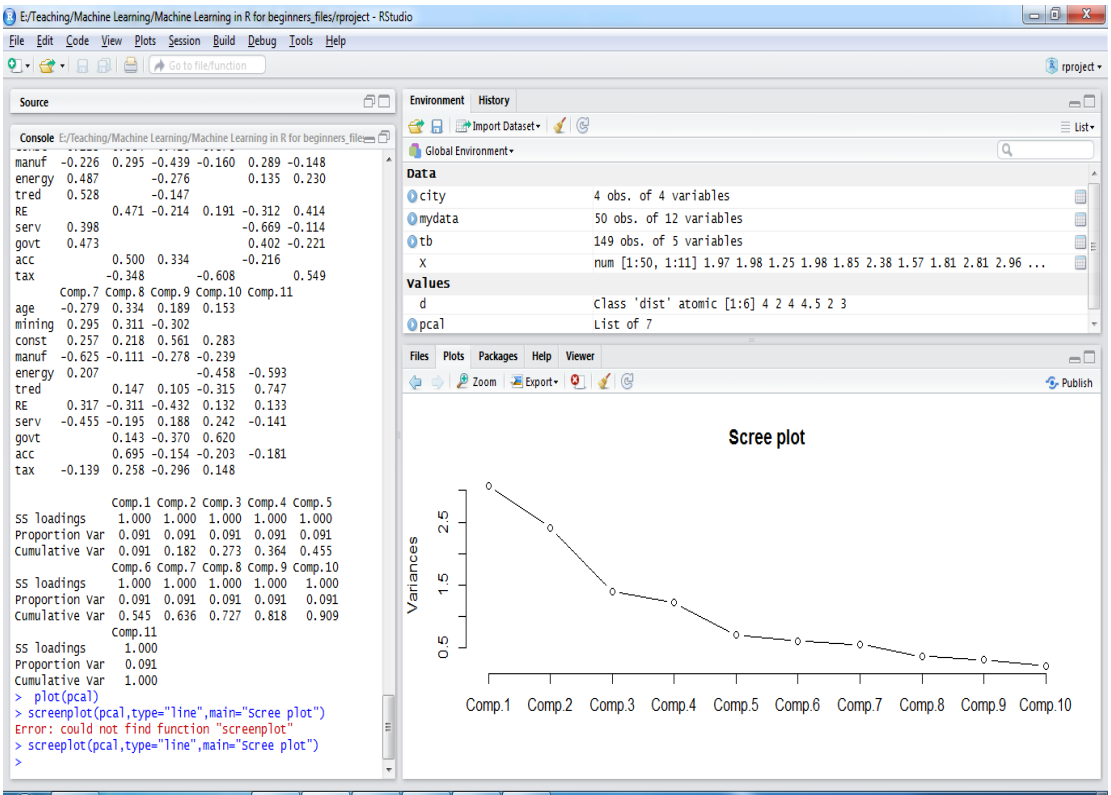


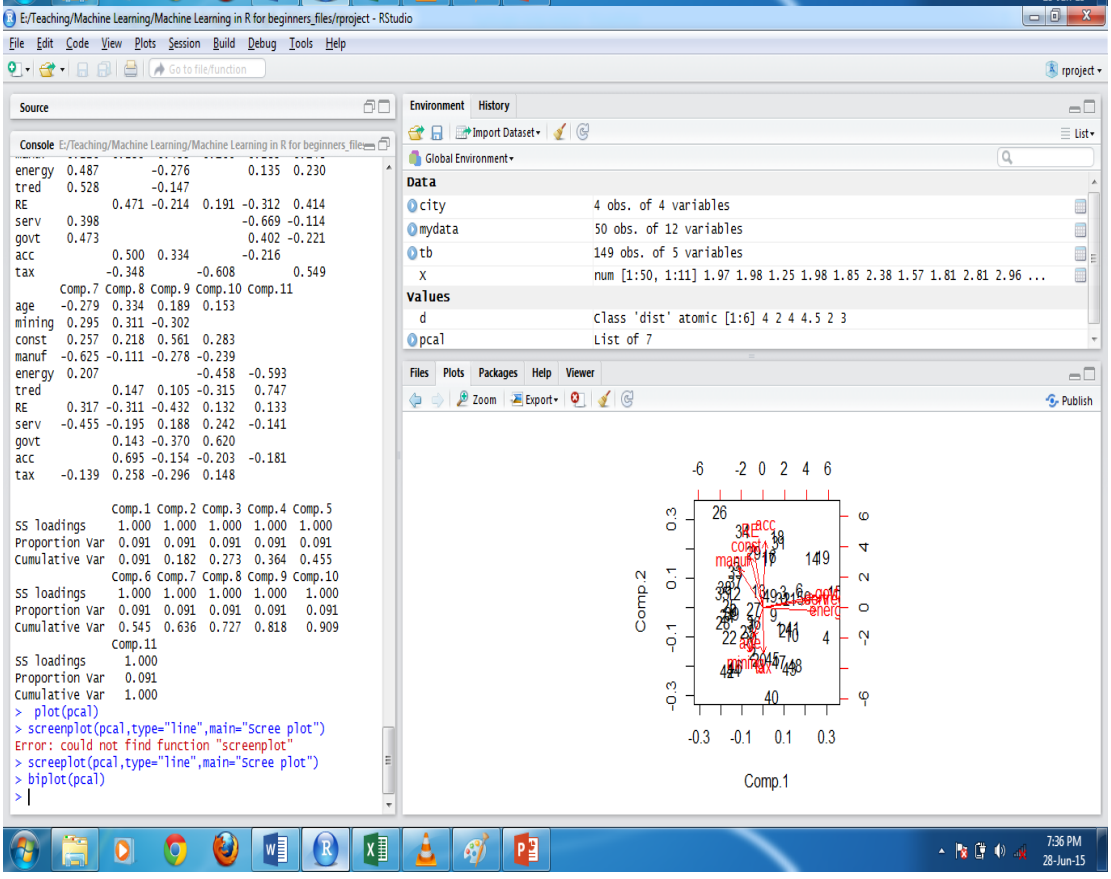

E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject/
object 'acc' not found
> mydata=read.csv("C:/Ramit/dataset/panalysis.csv")
> attach(mydata)
The following objects are masked from mydata (pos = 3):
age, const, energy, govt, manuf, mining, RE, serv, state, tred
> View(mydata)
> X <- cbind(age,mining,const,manuf,energy,tred,RE,serv,govt,acc,tax,)
Error in cbind(age, mining, const, manuf, energy, tred, RE, serv, govt, :
argument is missing, with no default
> X <- cbind(age,mining,const,manuf,energy,tred,RE,serv,govt,acc,tax)
> summary(X)
age mining const
Min. : 1.250 Min. :1.350 Min. :2.100
1st Qu.: 1.850 1st Qu.:1.665 1st Qu.:2.292
Median : 2.100 Median :1.790 Median :2.435
Mean : 6.168 Mean :2.062 Mean :2.459
3rd Qu.:13.377 3rd Qu.:2.002 3rd Qu.:2.618
Max. :14.380 Max. :4.040 Max. :3.220
manuf energy tred
Min. :11.20 Min. :2.200 Min. :2.190
1st Qu.:16.00 1st Qu.:2.600 1st Qu.:2.658
Median :17.00 Median :2.800 Median :2.955
Mean :17.16 Mean :2.807 Mean :2.952
3rd Qu.:18.80 3rd Qu.:2.995 3rd Qu.:3.228
Max. :25.00 Max. :3.850 Max. :3.930
RE serv govt
Min. :0.1700 Min. :1.250 Min. :3.520
1st Qu.:0.2600 1st Qu.:1.548 1st Qu.:4.410
Median :0.2900 Median :1.860 Median :5.150
Mean :0.2968 Mean :1.870 Mean :5.375
3rd Qu.:0.3200 3rd Qu.:2.080 3rd Qu.:6.013
Max. :0.5000 Max. :2.960 Max. :8.900
acc tax
Min. :0.820 Min. :2.510
1st Qu.:1.020 1st Qu.:2.855
Median :1.070 Median :3.170
Environment History
Import Dataset
Global Environment
Data
city 4 obs. of 4 variables
mydata 50 obs. of 12 variables
tb 149 obs. of 5 variables
X num [1:50, 1:11] 1.97 1.98 1.25 1.98 1.85 2.38 1...
Values
d Class 'dist' atomic [1:6] 4 2 4 4.5 2 3
pcal List of 7
Files Plots Packages Help Viewer
Zoom Export
7:11 PM
28-Jun-15


E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject/
acc 0.429730589 0.098422493 0.08471711
tax -0.393707816 0.025788473 -0.16190326
acc tax
age -0.31759839 -0.11083118
mining -0.49854425 0.16831381
const 0.22994505 -0.08120453
manuf 0.11350606 -0.14000558
energy -0.15092172 0.09706626
tred 0.06694537 -0.06853652
RE 0.42973059 -0.39370782
serv 0.09842249 0.02578847
govt 0.08471711 -0.16190326
acc 1.00000000 -0.36384733
tax -0.36384733 1.00000000
> pcal <- princomp(X,scores = TRUE, cor = TRUE)
> summary(pcal)
Importance of components:
Comp.1 Comp.2 Comp.3
Standard deviation 1.7540704 1.5524704 1.1848811
Proportion of Variance 0.2797057 0.2191058 0.1276312
Cumulative Proportion 0.2797057 0.4988116 0.6264428
Comp.4 Comp.5 Comp.6
Standard deviation 1.1037227 0.84197111 0.78471981
Proportion of Variance 0.1107458 0.06444685 0.05598047
Cumulative Proportion 0.7371886 0.80163545 0.85761592
Comp.7 Comp.8
Standard deviation 0.74482525 0.60894517
Proportion of Variance 0.05043315 0.03371038
Cumulative Proportion 0.90804907 0.94175945
Comp.9 Comp.10
Standard deviation 0.55758346 0.46483956
Proportion of Variance 0.02826357 0.01964326
Cumulative Proportion 0.97002303 0.98966628
Comp.11
Standard deviation 0.33715116
Proportion of Variance 0.01033372
Cumulative Proportion 1.00000000
Environment History
Import Dataset
Global Environment
Data
city 4 obs. of 4 variables
mydata 50 obs. of 12 variables
tb 149 obs. of 5 variables
X num [1:50, 1:11] 1.97 1.98 1.25 1.98 1.85 2.38 1...
Values
d Class 'dist' atomic [1:6] 4 2 4 4.5 2 3
pcal List of 7
Files Plots Packages Help Viewer
Zoom Export
7:14 PM
28-Jun-15


E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject/
> loadings(pcal)
Loadings:
Environment History
Import Dataset
Global Environment
Data
city 4 obs. of 4 variables
mydata 50 obs. of 12 variables
tb 149 obs. of 5 variables
X num [1:50, 1:11] 1.97 1.98 1.25 1.98 1.85 2.38 1...
Values
d Class 'dist' atomic [1:6] 4 2 4 4.5 2 3
pcal List of 7
Files Plots Packages Help Viewer
7:30 PM
28-Jun-15


E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/
> plot(pcal)
Environment History
Import Dataset
Global Environment
Data
city 4 obs. of 4 variables
mydata 50 obs. of 12 variables
tb 149 obs. of 5 variables
X num [1:50, 1:11] 1.97 1.98 1.25 1.98 1.85 2.38 1.57 1.81 2.81 2.96 ...
Values
d Class 'dist' atomic [1:6] 4 2 4 4.5 2 3
pcal List of 7
Files Plots Packages Help Viewer
Zoom Export Publish
pcal
Variances
0.0 1.0 2.0 3.0
Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
7:31 PM
28-Jun-15


E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/
manuf -0.226 0.295 -0.439 -0.160 0.289 -0.148
energy 0.487 -0.276 0.135 0.230
tred 0.528 -0.147
RE 0.471 -0.214 0.191 -0.312 0.414
serv 0.398 -0.669 -0.114
govt 0.473 0.402 -0.221
acc 0.500 0.334 -0.216
tax -0.348 -0.608 0.549
Comp.7 Comp.8 Comp.9 Comp.10 Comp.11
age -0.279 0.334 0.189 0.153
mining 0.295 0.311 -0.302
const 0.257 0.218 0.561 0.283
manuf -0.625 -0.111 -0.278 -0.239
energy 0.207 -0.458 -0.593
tred 0.147 0.105 -0.315 0.747
RE 0.317 -0.311 -0.432 0.132 0.133
serv -0.455 -0.195 0.188 0.242 -0.141
govt 0.143 -0.370 0.620
acc 0.695 -0.154 -0.203 -0.181
tax -0.139 0.258 -0.296 0.148
Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
SS loadings 1.000 1.000 1.000 1.000 1.000
Proportion Var 0.091 0.091 0.091 0.091 0.091
Cumulative Var 0.091 0.182 0.273 0.364 0.455
Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
SS loadings 1.000 1.000 1.000 1.000 1.000
Proportion Var 0.091 0.091 0.091 0.091 0.091
Cumulative Var 0.545 0.636 0.727 0.818 0.909
Comp.11
SS loadings 1.000
Proportion Var 0.091
Cumulative Var 1.000
> plot(pcal)
> screenplot(pcal,type="line",main="Scree plot")
Error: could not find function "screenplot"
> screeplot(pcal,type="line",main="Scree plot")
Environment History
Import Dataset
Global Environment
Data
city 4 obs. of 4 variables
mydata 50 obs. of 12 variables
tb 149 obs. of 5 variables
X num [1:50, 1:11] 1.97 1.98 1.25 1.98 1.85 2.38 1.57 1.81 2.81 2.96 ...
Values
d Class 'dist' atomic [1:6] 4 2 4 4.5 2 3
pcal List of 7
Files Plots Packages Help Viewer
Zoom Export Publish
Scree plot
Variances
0.5 1.5 2.5
Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
7:35 PM
28-Jun-15


E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/rproject - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console E:/Teaching/Machine Learning/Machine Learning in R for beginners_files/
energy 0.487 -0.276 0.135 0.230
tred 0.528 -0.147
RE 0.471 -0.214 0.191 -0.312 0.414
serv 0.398 -0.669 -0.114
govt 0.473 0.402 -0.221
acc 0.500 0.334 -0.216
tax -0.348 -0.608 0.549
Comp.7 Comp.8 Comp.9 Comp.10 Comp.11
age -0.279 0.334 0.189 0.153
mining 0.295 0.311 -0.302
const 0.257 0.218 0.561 0.283
manuf -0.625 -0.111 -0.278 -0.239
energy 0.207 -0.458 -0.593
tred 0.147 0.105 -0.315 0.747
RE 0.317 -0.311 -0.432 0.132 0.133
serv -0.455 -0.195 0.188 0.242 -0.141
govt 0.143 -0.370 0.620
acc 0.695 -0.154 -0.203 -0.181
tax -0.139 0.258 -0.296 0.148
Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
SS loadings 1.000 1.000 1.000 1.000 1.000
Proportion Var 0.091 0.091 0.091 0.091 0.091
Cumulative Var 0.091 0.182 0.273 0.364 0.455
Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
SS loadings 1.000 1.000 1.000 1.000 1.000
Proportion Var 0.091 0.091 0.091 0.091 0.091
Cumulative Var 0.545 0.636 0.727 0.818 0.909
Comp.11
SS loadings 1.000
Proportion Var 0.091
Cumulative Var 1.000
> plot(pcal)
> screenplot(pcal,type="line",main="Scree plot")
Error: could not find function "screenplot"
> screeplot(pcal,type="line",main="Scree plot")
> biplot(pcal)
Environment History
Import Dataset
Global Environment
Data
city 4 obs. of 4 variables
mydata 50 obs. of 12 variables
tb 149 obs. of 5 variables
X num [1:50, 1:11] 1.97 1.98 1.25 1.98 1.85 2.38 1.57 1.81 2.81 2.96 ...
Values
d Class 'dist' atomic [1:6] 4 2 4 4.5 2 3
pcal List of 7
Files Plots Packages Help Viewer
Zoom Export Publish
-6 -2 0 2 4 6
Comp.2
-0.3 -0.1 0.1 0.3
Comp.1
7:36 PM
28-Jun-15

FAQs :

1. What is the **Recommendations for a final project on Principal Component Analysis**

2. **How to apply distance-based clustering or dimensionality reduction for too many samples**

3. **Is large scale PCA even possible?**

4. **Evaluate output of different dimensionality reduction methods**

5. Which are the **Techniques for plotting PCA projections in more than three dimensions**

PRACTISE ASSIGNMENTS

1. Perform above on different input dataset.

Assignment 6 Supervised Learning and Kernel Methods

AIM : Write a program to Design, Implement SVM for classification on IRIS data set. Comment on Design and Implementation for Linearly non separable Dataset.

OBJECTIVE:

- To understand the designing of SVM classifier
- Analyze the above algorithms for linearly non separable Dataset.

THEORY:

SVM:

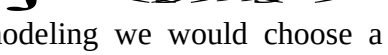
Machine Learning is considered as a subfield of Artificial Intelligence and it is concerned with the development of techniques and methods which enable the computer to learn. In simple terms development of algorithms which enable the machine to learn and perform tasks and activities. Machine learning overlaps with statistics in many ways. Over the period of time many techniques and methodologies were developed for machine learning tasks.

Support Vector Machine (SVM) was first heard in 1992, introduced by Boser, Guyon, and Vapnik in COLT-92. Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support

Vector machines can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Support vector machine was initially popular with the NIPS community and now is an active part of the machine learning research around the world. SVM becomes famous when, using pixel maps as input; it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task. It is also being used for many applications, such as hand writing analysis, face analysis and so forth, especially for pattern classification and regression based applications. The foundations of Support Vector Machines (SVM) have been developed by Vapnik and gained popularity due to many promising features such as better empirical performance. The formulation uses the Structural Risk Minimization (SRM) principle, which has been shown to be superior, , to traditional Empirical Risk Minimization (ERM) principle, used by conventional neural networks. SRM minimizes an upper bound on the expected risk, where as ERM minimizes the error on the training data. It is this difference which equips SVM with a greater ability to generalize, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to solve regression problems.

Statistical Learning Theory

The statistical learning theory provides a framework for studying the problem of gaining knowledge, making predictions, making decisions from a set of data. In simple terms, it enables the choosing of the hyper plane space such a way that it closely represents the underlying function in the target space.

In statistical learning theory the problem of supervised learning is formulated as follows. We are given a set of training data $\{(\mathbf{x}_1,y_1)... (\mathbf{x}_l,y_l)\}$ in $R^n \times R$ sampled according to unknown probability distribution $P(\mathbf{x},y)$, and a loss function $V(y,f(\mathbf{x}))$ that measures the error, for a given $\mathbf{x}$, $f(\mathbf{x})$ is "predicted" instead of the actual value y. The problem consists in finding a function f that minimizes the expectation of the error on new data that is, finding a function f that minimizes the expected error: 
In statistical modeling we would choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space. More on statistical learning theory can be found on introduction to statistical learning theory.

Learning and Generalization

Early machine learning algorithms aimed to learn representations of simple functions. Hence, the goal of learning was to output a hypothesis that performed the correct classification of the training data and early learning algorithms were designed to find such an accurate fit to the data. The ability of a hypothesis to correctly classify data not in the training set is known as its generalization. SVM performs better in term of not over generalization when the neural networks might end up over generalizing easily. Another thing to observe is to find where to make the best trade-off in trading complexity with the number of epochs; the illustration

brings to light more information about this. The below illustration is made from the class notes.

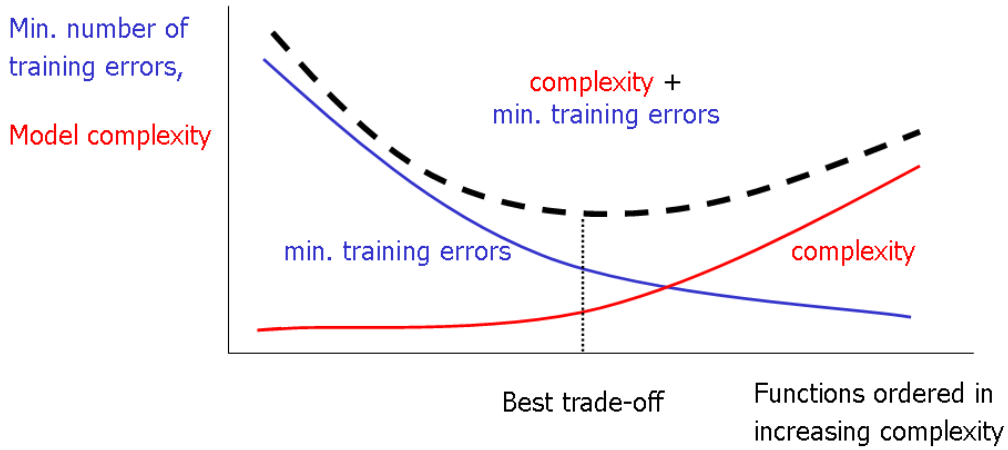


Figure 1: Number of Epochs Vs Complexity.

Introduction to SVM: Why SVM?

Firstly working with neural networks for supervised and unsupervised learning showed good results while used for such learning applications. MLP's uses feed forward and recurrent networks. Multilayer perceptron (MLP) properties include universal approximation of continuous nonlinear functions and include learning with input-output patterns and also involve advanced network architectures with multiple inputs and outputs.

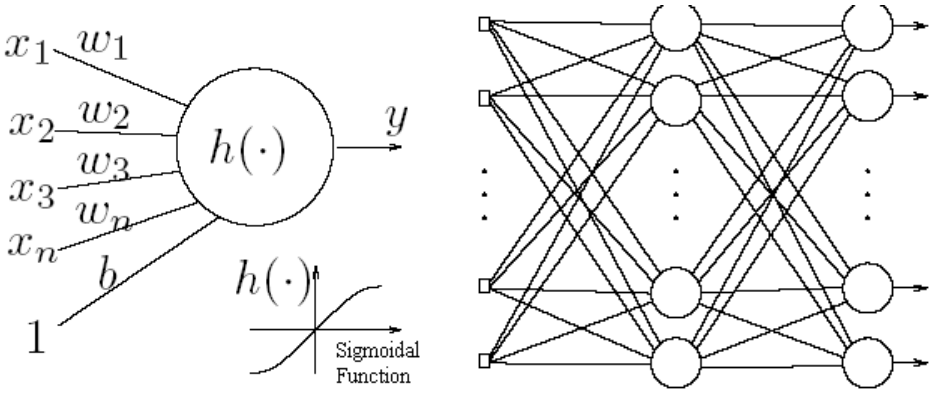


Figure 2: a] Simple Neural Network b]Multilayer Perceptron. [10][11]. These are simple visualizations just to have a overview as how neural network looks like.

There can be some issues noticed. Some of them are having many local minima and also finding how many neurons might be needed for a task is another issue which determines whether optimality of that NN is reached. Another thing to note is that even if the neural network solutions used tends to converge, this may not result in a unique solution. Now let us look at another example where we plot the data and try to classify it and we see that there are many hyper planes which can classify it. But which one is better?

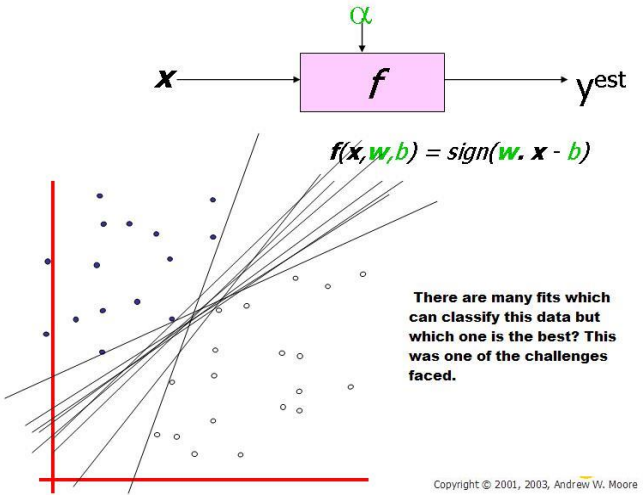


Figure 3: Here we see that there are many hyper planes which can be fit in to classify the data but which one is the best is the right or correct solution. The need for SVM arises. (Taken Andrew W. Moore 2003).

From above illustration, there are many linear classifiers (hyper planes) that separate the data. However only one of these achieves maximum separation. The reason we need it is because if we use a hyper plane to classify, it might end up closer to one set of datasets compared to others and we do not want this to happen and thus we see that the concept of maximum margin classifier or hyper plane as an apparent solution. The next illustration gives the maximum margin classifier example which provides a solution to the above mentioned problem.

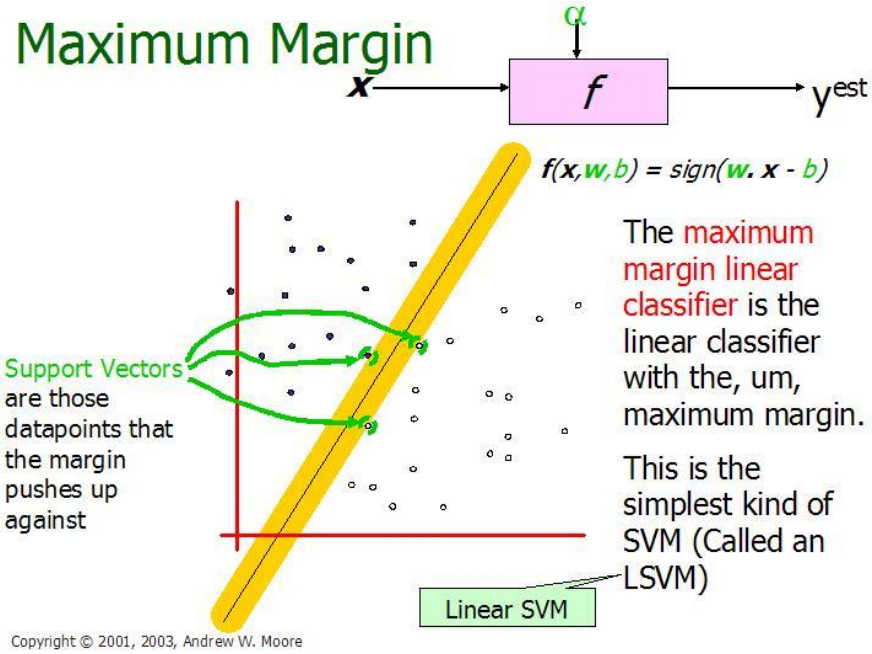


Figure 4: Illustration of Linear SVM. (Taken from Andrew W. Moore slides 2003).

Expression for Maximum margin is given as :

The above illustration is the maximum linear classifier with the maximum range. In this context it is an example of a simple linear SVM classifier. Another

interesting question is why maximum margin? There are some good explanations which include better empirical performance. Another reason is that even if we've made a small error in the location of the boundary this gives us least chance of causing a misclassification. The other advantage would be avoiding local minima and better classification. Now we try to express the SVM mathematically and for this tutorial we try to present a linear SVM. The goals of SVM are separating the data with hyper plane and extend this to non-linear boundaries using kernel trick. For calculating the SVM we see that the goal is to correctly classify all the data. For mathematical calculations we have,

[a] If $Y_i = +1$; $wx+b \geq 1$
[b] If $Y_i = -1$; $wx_i + b \leq 1$
[c] For all i; $y_i (w_i + b) \geq 1$

In this equation x is a vector point and w is weight and is also a vector. So to separate the data [a] should always be greater than zero. Among all possible hyper planes, SVM selects the one where the distance of hyper plane is as large as possible. If the training data is good and every test vector is located in radius r from training vector. Now if the chosen hyper plane is located at the farthest possible from the data. This desired hyper plane which maximizes the margin also bisects the lines between closest points on convex hull of the two datasets. Thus we have [a], [b] & [c].

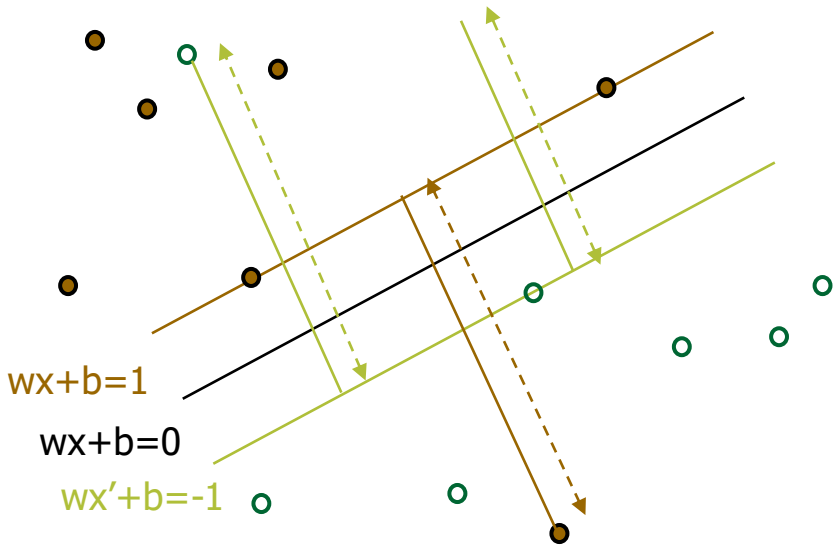


Figure 5: Representation of Hyper planes. [9]

Distance of closest point on hyperplane to origin can be found by maximizing the x as x is on the hyper plane. Similarly for the other side points we have a similar scenario. Thus solving and subtracting the two distances we get the summed distance from the separating hyperplane to nearest points. Maximum Margin = M = 2 / ||w||

Now maximizing the margin is same as minimum. Now we have a quadratic optimization problem and we need to solve for w and b. To solve this we need to optimize the quadratic function with linear constraints. The solution involves constructing a dual problem and where a Langlier's multiplier α_i is associated. We need to find w and b such that $\Phi(w) = \frac{1}{2} |w'||w|$ is minimized;

And for all $\{(x_i, y_i)\}$: $y_i (w * x_i + b) \geq 1$.

Now solving: we get that w =$\Sigma\alpha_i * x_i$; $b = y_k$- w *x_k for any x_k such that $\alpha k \neq 0$

Now the classifying function will have the following form: $f(x) = \Sigma\alpha_i y_i x_i * x +$ b

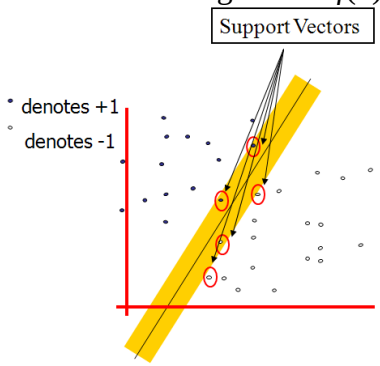


Figure 6: Representation of Support Vectors (Copyright © 2003, Andrew W. Moore)

SVM Representation

In this we present the QP formulation for SVM classification. This is a simple representation only.

SV classification:

$$\min_{f,\xi} \|f\|_K^2 + C\sum_{i=1}^{l}\xi_i \qquad y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \text{ for all } i \quad \xi_i \geq 0$$

SVM classification, Dual formulation:

$$\max_{\alpha} \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \qquad 0 \leq \alpha_i \leq C, \text{ for all } i;$$

$$\sum_{i=1}^{l}\alpha_i y_i = 0$$

Variables ξ_i are called slack variables and they measure the error made at point $(\mathbf{x}_i, y_i)$. Training SVM becomes quite challenging when the number of training points is large. A number of methods for fast SVM training have been proposed.

Soft Margin Classifier

In real world problem it is not likely to get an exactly separate line dividing the data within the space. And we might have a curved decision boundary. We might have a hyperplane which might exactly separate the data but this may not be desirable if the data has noise in it. It is better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers. This is handled in a different way; here we hear the term slack variables being introduced. Now we have, $y_i(w'x + b) \geq 1 - S_k$. This allows a point to be a small distance S_k on the wrong side of the hyper plane without violating the constraint. Now we might end up having huge slack variables which allow any line to separate the data, thus in such scenarios we have the Lagrangian variable introduced which penalizes the large slacks.

$$\min L = \tfrac{1}{2} w'w - \sum \lambda_k (y_k (w'x_k + b) + s_k - 1) + \alpha \sum s_k$$

Where reducing α allows more data to lie on the wrong side of hyper plane and would be treated as outliers which give smoother decision boundary.

Kernal Trick

Let's first look at few definitions as what is a kernel and what does feature space mean?

Kernel: If data is linear, a separating hyper plane may be used to divide the data. However it is often the case that the data is far from linear and the datasets are inseparable. To allow for this kernels are used to non-linearly map the input data to a high-dimensional space. The new mapping is then linearly separable. A very simple illustration of this is shown below in figure 7.

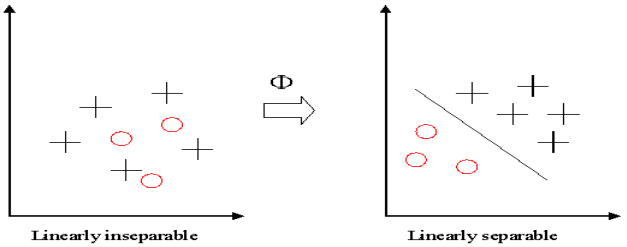


Figure 7: Why use Kernels?

This mapping is defined by the Kernel: $K(x,y) = \Phi(x) \cdot \Phi(y)$

Feature Space: Transforming the data into feature space makes it possible to define a similarity measure on the basis of the dot product. If the feature space is chosen suitably, pattern recognition can be easy .

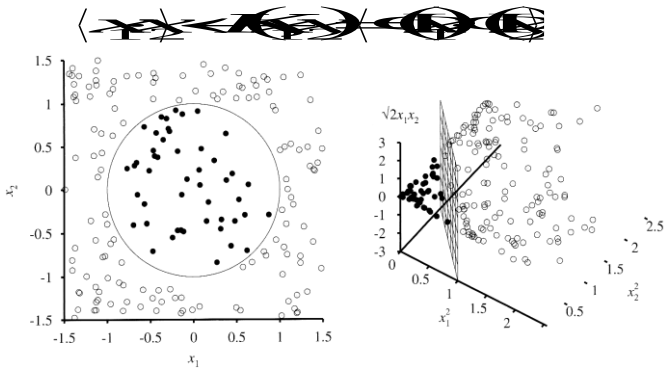


Figure 8: Feature Space Representation .

Now getting back to the kernel trick, we see that when w,b is obtained the problem is solved for a simple linear scenario in which data is separated by a hyper plane. The Kenral trick allows SVM's to form nonlinear boundaries. Steps involved in kernel trick are given below.

[a] The algorithm is expressed using only the inner products of data sets. This is also called as dual problem.

[b] Original data are passed through non linear maps to form new data with respect to new dimensions by adding a pair wise product of some of the original data dimension to each data vector.

[c] Rather than an inner product on these new, larger vectors, and store in tables and later do a table lookup, we can represent a dot product of the data after doing non linear mapping on them. This function is the kernel function. More on kernel functions is given below.

Kernal Trick: Dual Problem

First we convert the problem with optimization to the dual form in which we try to eliminate w, and a Lagrangian now is only a function of λ_i. There is a mathematical solution for it but this can be avoided here as this tutorial has instructions to minimize the mathematical equations, I would describe it instead. To solve the problem we should maximize the L_D with respect to λ_i.

Kernal Trick: Inner Product summarization

Here we see that we need to represent the dot product of the data vectors used. The dot product of nonlinearly mapped data can be expensive. The kernel trick just picks a suitable function that corresponds to dot product of some nonlinear mapping instead. Some of the most commonly chosen kernel functions are given below in later part of this tutorial. A particular kernel is only chosen by trial and error on the test set, choosing the right kernel based on the problem or application would enhance SVM's performance.

Kernel Functions

The idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner product does not need to be evaluated in the feature space. We want the function to perform mapping of the attributes of the input space to the feature space. The kernel function plays a critical role in SVM and its performance. It is based upon reproducing Kernel Hilbert Spaces.

$$K(x, x') = \langle \phi(x), \phi(x') \rangle,$$

If K is a symmetric positive definite function, which satisfies Mercer's Conditions,

$$K(x, x') = \sum_{m}^{\infty} a_m \phi_m(x) \phi_m(x'), \quad a_m \geq 0,$$

$$\iint K(x, x') g(x) g(x') dx dx' > 0, \quad g \in L_2$$

Then the kernel represents a legitimate inner product in feature space. The training set is not linearly separable in an input space. The training set is linearly separable in the feature space. This is called the "Kernel trick".

The different kernel functions are listed below:

1] *Polynomial:* A polynomial mapping is a popular method for non-linear modeling. The second kernel is usually preferable as it avoids problems with the hessian becoming Zero.

$$K(x, x') = \langle x, x' \rangle^d.$$

$$K(x, x') = \left(\langle x, x' \rangle + 1 \right)^d.$$

2] *Gaussian Radial Basis Function*: Radial basis functions most commonly with a Gaussian form

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2} \right)$$

3] *Exponential Radial Basis Function*: A radial basis function produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right)$$

4] *Multi-Layer Perceptron*: The long established MLP, with a single hidden layer, also has a valid kernel representation.

$$K(x, x') = \tanh\left(\rho\langle x, x'\rangle + \varrho\right)$$

Controlling Complexity in SVM: Trade-offs

SVM is powerful to approximate any training data and generalizes better on given datasets. The complexity in terms of kernel affects the performance on new datasets. SVM supports parameters for controlling the complexity and above all SVM does not tell us how to set these parameters and we should be able to determine these Parameters by Cross-Validation on the given datasets. The diagram given below gives a better illustration.

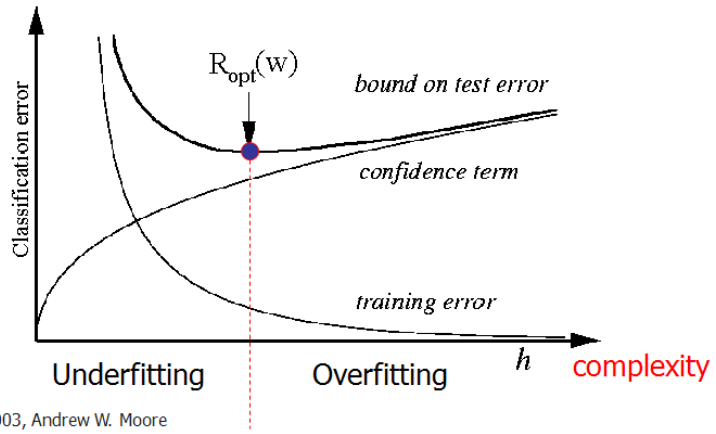


Figure 9: How to control complexity. Note the legend is not described as they are sample plotting to make understand the concepts involved.

SVM for Classification

SVM is a useful technique for data classification. Even though it's considered that Neural Networks are easier to use than this, however, sometimes unsatisfactory results are obtained. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

SVM for Regression

SVMs can also be applied to regression problems by the introduction of an alternative loss function. The loss function must be modified to include a distance measure. The regression can be linear and non linear. Linear models mainly consist of the following loss functions, e-intensive loss functions, quadratic and Huber loss function. Similarly to classification problems, a non-linear model is usually required to adequately model data. In the same manner as the non-linear SVC approach, a non-linear mapping can be used to map the data into a high dimensional feature space where linear regression is performed. The kernel approach is again employed to address the curse of dimensionality. In the

regression method there are considerations based on prior knowledge of the problem and the distribution of the noise. In the absence of such information Huber's robust loss function, has been shown to be a good alternative.

Applications of SVM

SVM has been found to be successful when used for pattern classification problems. Applying the Support Vector approach to a particular practical problem involves resolving a number of questions based on the problem definition and the design involved with it. One of the major challenges is that of choosing an appropriate kernel for the given application. There are standard choices such as a Gaussian or polynomial kernel that are the default options, but if these prove ineffective or if the inputs are discrete structures more elaborate kernels will be needed. By implicitly defining a feature space, the kernel provides the description language used by the machine for viewing the data. Once the choice of kernel and optimization criterion has been made the key components of the system are in place. Let's look at some examples.

The task of text categorization is the classification of natural text documents into a fixed number of predefined categories based on their content. Since a document can be assigned to more than one category this is not a multi-class classification problem, but can be viewed as a series of binary classification problems, one for each category. One of the standard representations of text for the purposes of information retrieval provides an ideal feature mapping for constructing a Mercer kernel. Indeed, the kernels somehow incorporate a similarity measure between instances, and it is reasonable to assume that experts working in the specific application domain have already identified valid similarity measures, particularly in areas such as information retrieval and generative models.

Traditional classification approaches perform poorly when working directly because of the high dimensionality of the data, but Support Vector Machines can avoid the pitfalls of very high dimensional representations. A very similar approach to the techniques described for text categorization can also be used for the task of image classification, and as in that case linear hard margin machines are frequently able to generalize well. The first real-world task on which Support Vector Machines were tested was the problem of hand-written character recognition. Furthermore, multi-class SVMs have been tested on these data. It is interesting not only to compare SVMs with other classifiers, but also to compare different SVMs amongst themselves.

Strength and Weakness of SVM:

1. The major strengths of SVM are the training is relatively easy. No local optimal, unlike in neural networks. It scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly. The weakness includes the need for a good kernel function

Implemntation of SVM in R:

Generate toy data:

- #First generate a set of positive and negative examples from 2 Gaussians.

```
n <- 150              # number of data points
p <- 2                # dimension
sigma <- 1            # variance of the distribution
meanpos <- 0          # centre of the distribution of positive examples
meanneg <- 3          # centre of the distribution of negative examples
npos <- round(n/2)    # number of positive examples
nneg <- n-npos        # number of negative examples
```

- # Generate the positive and negative examples

```
xpos<-atrix(rnorm(npos*p,mean=meanpos,sd=sigma),   npos,p)
xneg<- matrix(rnorm(nneg*p,mean=meanneg,sd=sigma), npos,p)
x <- rbind(xpos,xneg)
```

- # Generate the labels

```
y <- matrix(c(rep(1,npos),rep(-1,nneg)))
```

- # Visualize the data

```
plot(x,col=ifelse(y>0,1,2))
legend("topleft",c('Positive','Negative'),col=seq(2),pch=1,text.col=seq(2))
```

- Now we split the data into a training set (80%) and a test set (20%):

```
## Prepare a training and a test set ##
ntrain<-round(n*0.8)          # number of training examples
tindex<-sample(n,ntrain)      # indices of training samples
xtrain <- x[tindex,]
xtest <- x[-tindex,]
ytrain <- y[tindex]
ytest <- y[-tindex]
istrain=rep(0,n)
istrain[tindex]=1
```

- # Visualize

```
plot(x,col=ifelse(y>0,1,2),pch=ifelse(istrain==1,1,2))
legend("topleft",c('Positive Train','Positive Test','Negative Train','Negative Test'),
col=c(1,1,2,2),pch=c(1,2,1,2),text.col=c(1,1,2,2))
```

2. **Train a SVM**

Now we train a linear SVM with parameter C=100 on the training set

- # load the kernlab package

```
library(kernlab)
```

- # train the SVM

```
svp<-ksvm(xtrain,ytrain,type="C- svc",kernel='vanilladot',C=100,scaled=c())
```

Look and understand what svp contains

- # General summary

```
svp
```

- # Attributes that you can access

```
attributes(svp)
```

- # For example, the support vectors

```
alpha(svp)
alphaindex(svp)
b(svp)
```

- # Use the built-in function to pretty-plot the classifier

```
plot(svp,data=xtrain)
```

3. **Predict with a SVM**

Now we can use the trained SVM to predict the label of points in the test set, and we analyze the results using variant metrics.

- # Predict labels on test

```
ypred = predict(svp,xtest)
table(ytest,ypred)
```

- # Compute accuracy

```
sum(ypred==ytest)/length(ytest)
```

- # Compute at the prediction scores

```
ypredscore = predict(svp,xtest,type="decision")
```

- # Check that the predicted labels are the signs of the scores

```
table(ypredscore > 0,ypred)
```

- # Package to compute ROC curve, precision-recall etc...

```
library(ROCR)
pred <- prediction(ypredscore,ytest)
```

- # Plot ROC curve

```
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf)
```

- # Plot precision/recall curve

```
perf <- performance(pred, measure = "prec", x.measure = "rec")
plot(perf)
```

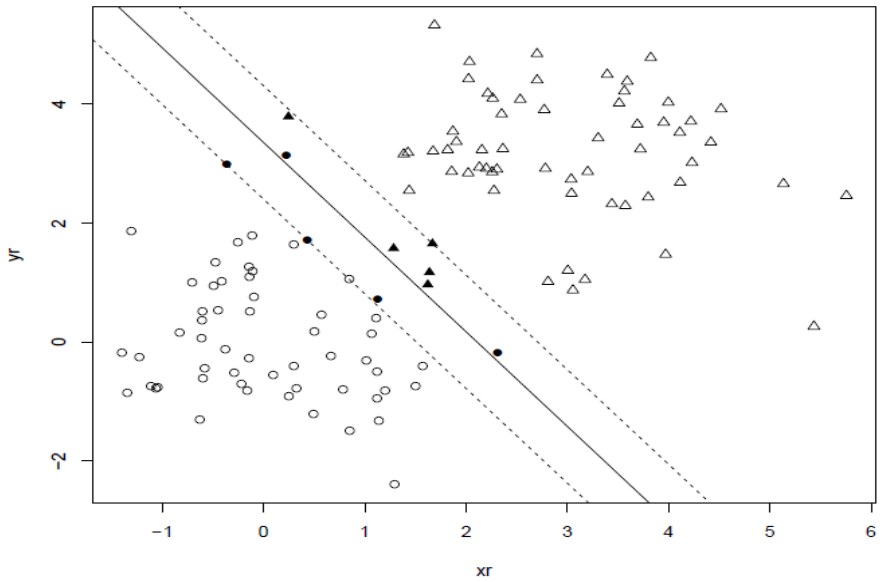


Figure: A linear SVM with decision boundary f(x) = 0. Dotted lines correspond to the level sets f(x) = 1 and f(x) =-1 Support vectors are in black.

FAQs:

1. How to calculate decision boundary from support vectors?
2. What is the difference between SVM and CRF models in the context of NER?
3. How to select best cross validated SVM (support vector machine) model when using K fold CV (5)?
4. Advantages and disadvantages of machine learning hyperparameter optimizers [on hold]
5. Why an observation might have predicted class probabilities differ from its predicted response in a svm classifier? How can I use both?
6. What is the meaning of the relevance labels in SVM-Rank?
7. Is this training dataset enough for training and testing classification model?

8. How do I use weight vector of SVM and logistic regression for feature importance?

PRACTISE ASSIGNMENTS

1. Perform above on different input dataset.