Laboratory Manual

For

Data Mining

(MF102)

M.Tech (IT)

SEM I

June 2011

Faculty of Technology
Dharmsinh Desai University
Nadiad
www.ddu.ac.in

Table of Contents

Experiment 1: Introduction to R Programming ... 3
Experiment 2: Data Pre-processing ... 12
Experiment 3: Data Visualization ... 13
Experiment 4: Data similarity measures ... 14
Experiment 5: Association rule mining in r ... 16
Experiment 6: Classification using Decision Tree ... 17
Experiment 7: Classification using Naive Bayes Classifier ... 18
Experiment 8: Classification using KNN ... 19
Experiment 9: K-fold cross validation and comparing models in ROC curve ... 20
Experiment 10: Clustering using k-means ... 21
Experiment 11: Classification using SVM ... 22
Experiment 12: Clustering using hierarchical clustering ... 23
References ... 25

Experiment 1

Aim: Basic R Operations: Reading/Writing data using frames, element wise operation, data selection, loops and decision-making statements.

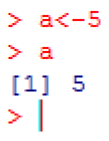
Tools/Apparatus: R studio

Introduction to R

R provides an environment to handle a large amount of data and analyse it very effectively. In R we can create or load existing database and perform various operations on data to analyse such as mathematical operations, normalizations, plotting data on to graph, etc.

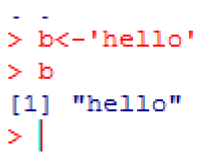
There are basically a three data types in R, as shown bellow

Numeric



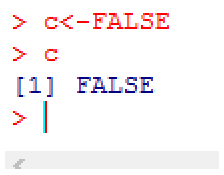
```
> a<-5
> a
[1] 5
> 
```

Character



```
> b<-'hello'
> b
[1] "hello"
> 
```

Logical



```
> c<-FALSE
> c
[1] FALSE
> 
```

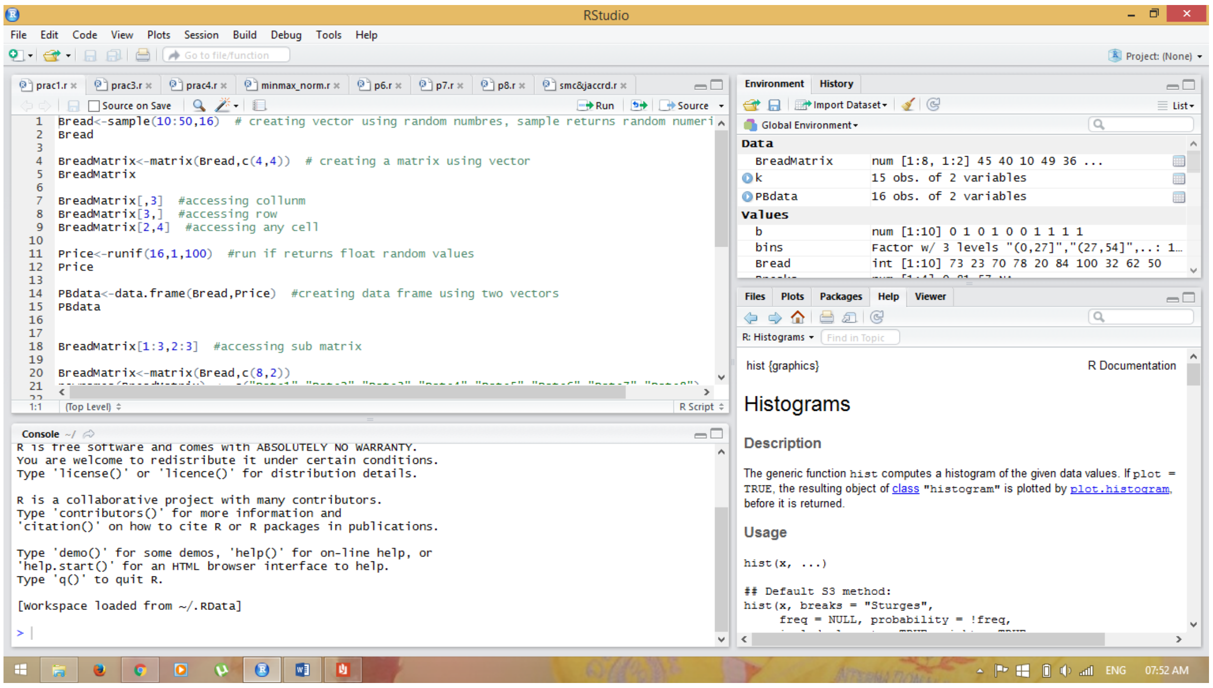
In above examples it is shown how to declare a variable and assign a value to it.

RStudio

R is a freely available environment for statistical computing. You need to tell R to what to do using commands. RStudio provides an interface which is very easy to understand to do programs in R. The bellow image shows the screen of RStudio.

The upper left panel is workplace for writing scripts in R. It is easy to run a lines of script pressing CTRL+R. Hence you can run a complete screen by selecting all lines and pressing CTRL+R.

The lower left panel is a console, where you can write code directly to the R as a command line. And it also gives the output of the scripts.



```
Bread<-sample(10:50,16)  # creating vector using random numbres, sample returns random numeri
Bread

BreadMatrix<-matrix(Bread,c(4,4))  # creating a matrix using vector
BreadMatrix

BreadMatrix[,3]  #accessing collunm
BreadMatrix[3,]  #accessing row
BreadMatrix[2,4]  #accessing any cell

Price<-runif(16,1,100)  #run if returns float random values
Price

PBdata<-data.frame(Bread,Price)  #creating data frame using two vectors
PBdata


BreadMatrix[1:3,2:3]  #accessing sub matrix

BreadMatrix<-matrix(Bread,c(8,2))
```

```
Data
  BreadMatrix     num [1:8, 1:2] 45 40 10 49 36 ...
  k               15 obs. of 2 variables
  PBdata          16 obs. of 2 variables
Values
  b               num [1:10] 0 1 0 1 0 0 1 1 1 1
  bins            Factor w/ 3 levels "(0,27]","(27,54]",..: 1…
  Bread           int [1:10] 73 23 70 78 20 84 100 32 62 50
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> |
```

hist {graphics} R Documentation

Histograms

Description

The generic function `hist` computes a histogram of the given data values. If `plot = TRUE`, the resulting object of class `"histogram"` is plotted by `plot.histogram`, before it is returned.

Usage

```
hist(x, ...)

## Default S3 method:
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
```

The upper right panel is a workspace where you can see the all objects such as datasets and variables. Clicking on the name of a dataset in your workspace will bring up a spreadsheet of the data

The bottom right serves many purposes. It is where plots will be appear, where you manage your files (including importing files from your computer), where you install packages, and where the help information appears. Use the tabs to toggle back and forth between these screens as needed.

Installing Packages

Go to console and type following command to install any packages.

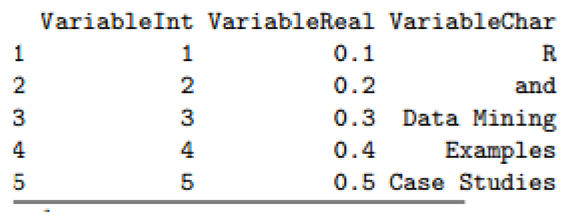
>install.packages(‘**package name**’,dependencies=TRUE)

Reading/Writing data using frames

The example below creates a dataframe df1 and save it as a .CSV file with `write.csv()`. And then, the dataframe is loaded from file to df2 with `read.csv`().

```
> var1 <- 1:5
> var2 <- (1:5) / 10
> var3 <- c("R", "and", "Data Mining", "Examples", "Case Studies")
> df1 <- data.frame(var1, var2, var3)
> names(df1) <- c("VariableInt", "VariableReal", "VariableChar")
> write.csv(df1, "./data/dummmyData.csv", row.names = FALSE)
> df2 <- read.csv("./data/dummmyData.csv")
> print(df2)
```

Output:



```
  VariableInt VariableReal VariableChar
1           1          0.1            R
2           2          0.2          and
3           3          0.3  Data Mining
4           4          0.4     Examples
5           5          0.5 Case Studies
```

Element wise Operators

You often may want to perform an operation on each element of a vector while doing a computation. For example, you may want to add two vectors by adding all of the corresponding elements. The addition (+) and subtraction (-) operators are defined to work on matrices as well as scalars. For example, if x = [1 2 3] and y = [5 6 2], then

```
>>w = x+y
w =    6     8     5
```

Multiplying two matrices element by element is a little different. The * symbol is defined as matrix multiplication when used on two matrices. Use .* to specify element-wise multiplication. So, using the x and y from above,

```
>>w = x .* y
w =   5    12     6
```

You can perform exponentiation on a vector similarly. Typing x .^ 2 squares each element of x.

```
>> w = x .^ 2
w =  1   4   9
```

Finally, you cannot use / to divide two matrices element-wise, since / and \ are reserved for left and right matrix ``division." Instead, you must use the ./ function. For example:

```
>> w = y ./ x
w =  5.0000   3.0000   0.6667
```

All of these operations work for complex numbers as well.

Data selection

R has powerful indexing features for accessing object elements. These features can be used to select and exclude variables and observations. The following code snippets demonstrate ways to keep or delete variables and observations and to take random samples from a dataset.

Selecting (Keeping) Variables:

```
# select variables v1, v2, v3
myvars <- c("v1", "v2", "v3")
```

```
newdata <- mydata[myvars]

# another method
myvars <- paste("v", 1:3, sep="")
newdata <- mydata[myvars]

# select 1st and 5th thru 10th variables
newdata <- mydata[c(1,5:10)]
```

Excluding (DROPPING) Variables:

```
# exclude variables v1, v2, v3
myvars <- names(mydata) %in% c("v1", "v2", "v3")
newdata <- mydata[!myvars]

# exclude 3rd and 5th variable
newdata <- mydata[c(-3,-5)]

# delete variables v3 and v5
mydata$v3 <- mydata$v5 <- NULL
```

Selecting Observations:

```
# first 5 observations
newdata <- mydata[1:5,]

# based on variable values
newdata <- mydata[ which(mydata$gender=='F' & mydata$age > 65), ]

# or
attach(newdata)
newdata <- mydata[ which(gender=='F' & age > 65),]
detach(newdata)
```

Selection using the Subset Function:

The **subset( )** function is the easiest way to select variables and observations. In the following example, we select all rows that have a value of age greater than or equal to 20 or age less then 10. We keep the ID and Weight columns.

```
# using subset function
newdata <- subset(mydata, age >= 20 | age < 10,select=c(ID, Weight))
```

In the next example, we select all men over the age of 25 and we keep variables weight *through* income (weight, income and all columns between them).

```
#using subset function (part 2)
newdata <- subset(mydata, sex=="m" & age > 25,select=weight:income)
```

Writing Loops & Decision making statements

For Loops

In R a while takes this form, where *variable* is the name of your iteration variable, and *sequence* is a vector or list of values:

for (variable in sequence) expression

The *expression* can be a single R command - or several lines of commands wrapped in curly brackets:

for (variable in sequence) {

expression

expression

expression

}

Here is a quick trivial example, printing the square root of the integers one to ten:

```
> for (x in c(1:10)) print(sqrt(x))
[1] 1
[1] 1.414214
[1] 1.732051
[1] 2
[1] 2.236068
[1] 2.449490
[1] 2.645751
[1] 2.828427
[1] 3
[1] 3.162278
```

While Loop

In R a while takes this form, where *condition* evaluates to a boolean (True/False) and must be wrapped in ordinary brackets:

while (condition) expression

As with a for loop, *expression* can be a single R command - or several lines of commands wrapped in curly brackets:

while (condition) {

expression

}

We'll start by using a "while loop" to print out the first few Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, ... where each number is the sum of the previous two numbers. Create a new R script file, and copy this code into it:

```
a <- 0
b <- 1
print(a)
while(b < 50){
    print(b)
    temp <- a + b
    a <- b
    b <- temp
}
```

If you go to the script's "Edit" menu and pick "Run all" you should get something like this in the R command console:

```
> a <- 0
> b <- 1
> print(a)
[1] 0
> while (b < 50) {
+     print(b)
+     temp <- a + b
+     a <- b
+     b <- temp
+ }
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
```

The code works fine, but both the output and the R commands are both shown in the R command window - its a bit messy. This next version builds up the answer gradually using a vector, which it prints at the end:

```
x <- c(0,1)
while(length(x) < 10){
    position <- length(x)
    new <- x[position] + x[position-1]
    x <- c(x,new)
}
print(x)
```

To understand how this manages to append the new value to the end of the vector x, try this at the command prompt:

```
> x <- c(1,2,3,4)
> c(x,5)
[1] 1 2 3 4 5
```

Writing Functions

This following script uses the function() command to create a function (based on the code above) which is then stored as an object with the name Fibonacci:

```
Fibonacci <- function(n){
    x <- c(0,1)
    while(length(x) < n){
        position <- length(x)
        new <- x[position] + x[position-1]
        x <- c(x,new)
    }
    return(x)
}
```

Once you run this code, there will be a new function available which we can now test:

```
> Fibonacci(10)
 [1]  0  1  1  2  3  5  8 13 21 34
> Fibonacci(3)
[1] 0 1 1
> Fibonacci(2)
[1] 0 1
> Fibonacci(1)
[1] 0 1
```

That seems to work nicely - except in the case n == 1 where the function is returning the first *two* Fibonacci numbers! This gives us an excuse to introduce the if statement.

If Statement

In order to fix our function we can do this:

```
Fibonacci <- function(n) {
    if (n==1) return(0)
    x <- c(0,1)
    while (length(x) < n) {
        position <- length(x)
        new <- x[position] + x[position-1]
        x <- c(x,new)
    }
    return(x)
}
```

In the above example we are using the simplest possible if statement:

if (condition) expression

The if statement can also be used like this:

*if (*condition*)* expression *else* expression

And, much like the while and for loops the *expression* can be multiline with curly brackets:

```
Fibonacci <- function(n) {
    if (n==1) {
        x <- 0
    } else {
        x <- c(0,1)
        while (length(x) < n) {
            position <- length(x)
            new <- x[position] + x[position-1]
            x <- c(x,new)
        }
    }
    return(x)
}
```

Experiment 2

Aim: Data Pre-processing: Mean, Median, Mode, Standard Deviation, Quantiles, Normalizations and Noise Removal.

Procedure:

Mean

It is calculated by taking the sum of the values and dividing with the number of values in a data series.

Median

The middle most value in a data series is called the median. The **median()**function is used in R to calculate this value.

Mode

The mode is the value that has highest number of occurrences in a set of data. Unike mean and median, mode can have both numeric and character data.

R does not have a standard in-built function to calculate mode. So you create a user function to calculate mode of a data set in R. This function takes the vector as input and gives the mode value as output.

Standard Deviation

The standard deviation of an observation variable is the square root of its variance.
Use existing function **sd()** to calculate it.

Quantiles

The generic function **quantile()** produces sample quantiles corresponding to the given probabilities. The smallest observation corresponds to a probability of 0 and the largest to a probability of 1.

Normalization

Min-Max normalization
Implement your logic to normalize dataframe using following equation,

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Z-score normalization

Implement your logic to normalize dataframe using following equation,

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation

Experiment 3

Aim: Data Visualization using scatter plot , histogram, Q-Q plot, Box-Plot.

Procedure:

Load two different dataset in to dataframes.

Use **Plot()** function to plot a scatter plot.

Create bins of data and use plot function to **plot()** histogram or use existing function **hist().**

Use functions **qqplot()** and **boxplot()** to draw a Q-Q plot and box plot.

Experiment 4

Aim: Data similarity measures: Euclidian, Manhattan, Supremum, Cosine, Jaccard coefficient.

Procedure:

Euclidian distance

Consider two points in a two-dimensional space (p1,p2) and (q1,q2) , the distance between these two points is given by the formula shown at below,

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2}.$$

This is called the Euclidian Distance between two points. The same concept can be extended easily to multidimensional space. If the points are (p1,p2,p3,p4,...) and (q1,q2,q3,q4,...), the distance between the points is given by the following formula,

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

--- **dist()** function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix.

--- Implement your logic to calculate **Manhattan and Supremum distance.**

--- **Cosine Similarity** is often used when comparing two documents against each other. It measures the angle between the two vectors. If the value is zero the angle between the two vectors is 90 degrees and they share no terms. If the value is 1 the two vectors are the same except for magnitude. Cosine is used when data is sparse, asymmetric and there is a similarity of lacking characteristics.

$$cos(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|}.$$

cosine() calculates a similarity matrix between all column vectors of a matrix x. This matrix might be a document-term matrix, so columns would be expected to be documents and rows to be terms. When executed on two vectors x and y, cosine() calculates the cosine similarity between them.

--- **Jaccard Coefficient** is used to compare documents. It measures the similarity of two sets by comparing the size of the overlap against the size of the two sets. Should the two sets have only binary attributes then it reduces to the Jaccard Coefficient. As with cosine, this is useful under the same data conditions and is well suited for market-basket data.

```
         j
       1   0
      -------
  1  | a | b |
i     -------
  0  | c | d |
      -------
a = number of variables on which both objects i and j are 1
b = number of variables where object i is 1 and j is 0
c = number of variables where object i is 0 and j is 1
d = number of variables where both i and j are 0
a+b+c+d = p, the nubmer of variables.
```

you can calculate from these values Jaccard similarity coefficients between any pair of objects,

$$\frac{a}{a+b+c}$$

Experiment 5

Aim: Association rule mining in r.

Theory:

Association rules are rules presenting association or correlation between itemsets.

Support $(A \Rightarrow B) = P(A \cup B)$

Confidence $(A \Rightarrow B) = P(B|A)$

$= P(A \cup B)/ P(A)$

Lift $(A \Rightarrow B) = \text{confidence}(A \Rightarrow B)/ P(B)$

$= P(A \cup B) /P(A)P(B)$

where P(A) is the percentage (or probability) of cases containing A.

APRIORI

A level-wise, breadth-first algorithm which counts transactions to find frequent itemsets and then derive association rules from them.

Procedure:

apriori() in package **arules**.

Read all transactions in to any variable. And use **apriori()** function to generate rules and then use **inspect()** function to generate desired output.

Experiment 6

Aim: Classification using Decision Tree.

Theory:

A decision tree is a flow chart like tree structure where each internal node(nonleaf) denotes a test on the attribute, each branch represents an outcome of the test ,and each leaf node(terminal node)holds a class label. Decision trees can be easily converted into classification rules.

Procedure:

Use **rpart** library.

Read all data.

Use **rpart()** function to generate decision tree and specify that which attribute will be the class label and which attributes needed to be classified.

Plot dicision tree using **plot()**.

Experiment 7

Aim: Classification using Naive Bayes Classifier.

Theory:

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$\underset{\text{Posterior Probability}}{P(c \mid x)} = \frac{\overset{\text{Likelihood}}{P(x \mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Procedure:

Use **E1071** library.

Read all training data.

Create model using **naiveBayes()** function on training data.

Use that model to predict the class label of test data.

Experiment 8

Aim: Classification using KNN.

Theory:

k nearest neighbours is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbours. This algorithms segregates unlabelled data points into well-defined groups.

The **knn ()** function needs to be used to train a model for which we need to install a package ‘**class**’. The knn() function identifies the k-nearest neighbours using Euclidean distance where k is a user-specified number.

Procedure:

Read training data.

Train a model using knn() on training data and predict a test data.

Now for evaluating a model you need to compare predicted class label of test data with actual class label of test data for that use **CrossTable()** present in **gmodels** library.

Experiment 09

Aim: K-fold cross validation and comparing models in ROC curve.

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*.

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.

Procedure:

Use **ROCR** library for Roc curve.

Read any dataset into data.

Divide all data into k subsets.

Apply your logic to generate k models using any classification method.

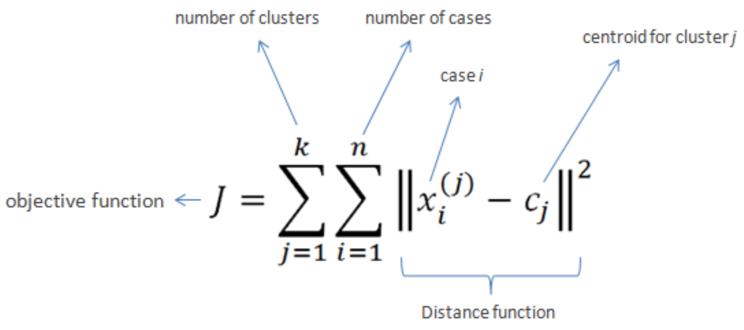
Predict the performance of each model using tpr and fpr and plot it in a roc curve.

Experiment 10

Aim: Clustering using k-means.

Theory:

K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly *k* different clusters of greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:



$$J = \sum_{j=1}^{k}\sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

Procedure:

Read a dataset into data for clustering.

Use **kmeans()** function to cluster the data in to number of specified clusters.

Use **table()** function to compare the results and plot the all clusters on graph.

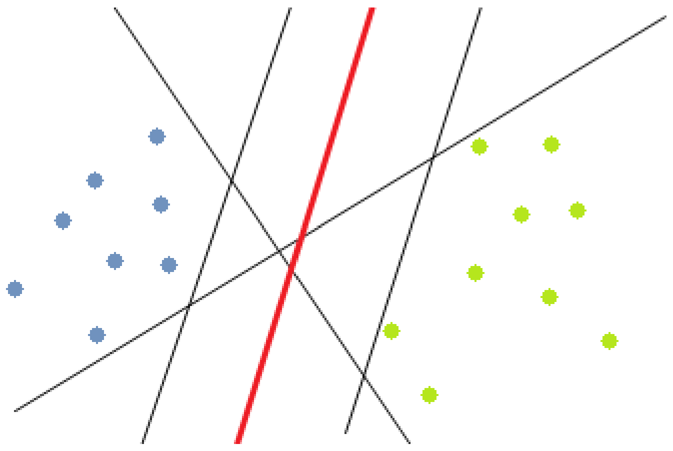
Experiment 11

Aim: Classification using SVM.

Theory:

svm is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation.

A classification task usually involves training and test sets which consist of data instances. Each instance in the training set contains one target value (class label) and several attributes (features). The goal of a classifier is to produce a model able to predict target values of data instances in the testing set, for which only the attributes are known. Without loss of generality, the classification problem can be viewed as a two-class problem in which one's objective is to separate the two classes by a function induced from available examples. The goal is to produce a classifier that generalizes well, i.e. that works well on unseen examples. The below picture is an example of a situation in which various linear classifiers can separate the data. However, only one maximizes the distance between itself and the nearest example of each class (i.e. the margin) and for that is called the optimal separating hyperplane. It is intuitively expected that this classifier generalizes better than the other options. The basic idea of SVM classifier uses this approach, i.e. to choose the hyperplane that has the maximum margin.



Procedure:

Use **E1071** library.

Use **svm()** function to generate a model using training data in which specify the attribute which is class label.

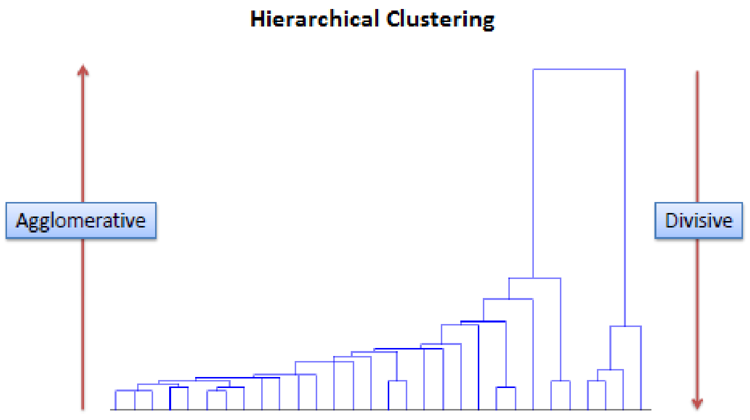
Test that model to predict test data.

Experiment 12

Aim: Clustering using hierarchical clustering.

Theory:

Hierarchical clustering involves creating clusters that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a hierarchy. There are two types of hierarchical clustering, *Divisive* and*Agglomerative*.



Divisive method

In this method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation.

Agglomerative method

In this method we assign each observation to its own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters. Finally, repeat steps 2 and 3 until there is only a single cluster left. The related algorithm is shown below.

```
Given:
A set X of objects $\{x_1,...,x_n\}$
A distance function $dist(c_1,c_2)$
for $i = 1$ to $n$
    $c_i = \{x_i\}$
end for
$C = \{c_1,...,c_n\}$
$l = n+1$
while $C$.size > 1 do
    – $(c_{min1},c_{min2})$ = minimum $dist(c_i,c_j)$ for all $c_i,c_j$ in $C$
    – remove $c_{min1}$ and $c_{min2}$ from $C$
    – add $\{c_{min1},c_{min2}\}$ to $C$
    – $l = l + 1$
end while
```

Before any clustering is performed, it is required to determine the proximity matrix containing the distance between each point using a distance function. Then, the matrix is updated to display the distance between each cluster. The following three methods differ in how the distance between each cluster is measured.

Procedure:

Use **hclust()** function to generate a hierarchical tree by specifying the distance method.

Then cut the tree into specified number of clusters.

Plot all clusters using **plot().**

References

Reference Books

- Data Mining: Concepts and Techniques, Third Edition

  Jiawei Han, Micheline Kamber, Jian Pei, Morgan Kaufmann , 2011.
- Data mining,
  Pieter Adriaans & Dolf Zantinge, Pearson Education Asia(2001).

Web References

- http://www.saedsayad.com/data_mining_map.htm
- http://www.rdatamining.com
- http://www.r-bloggers.com/