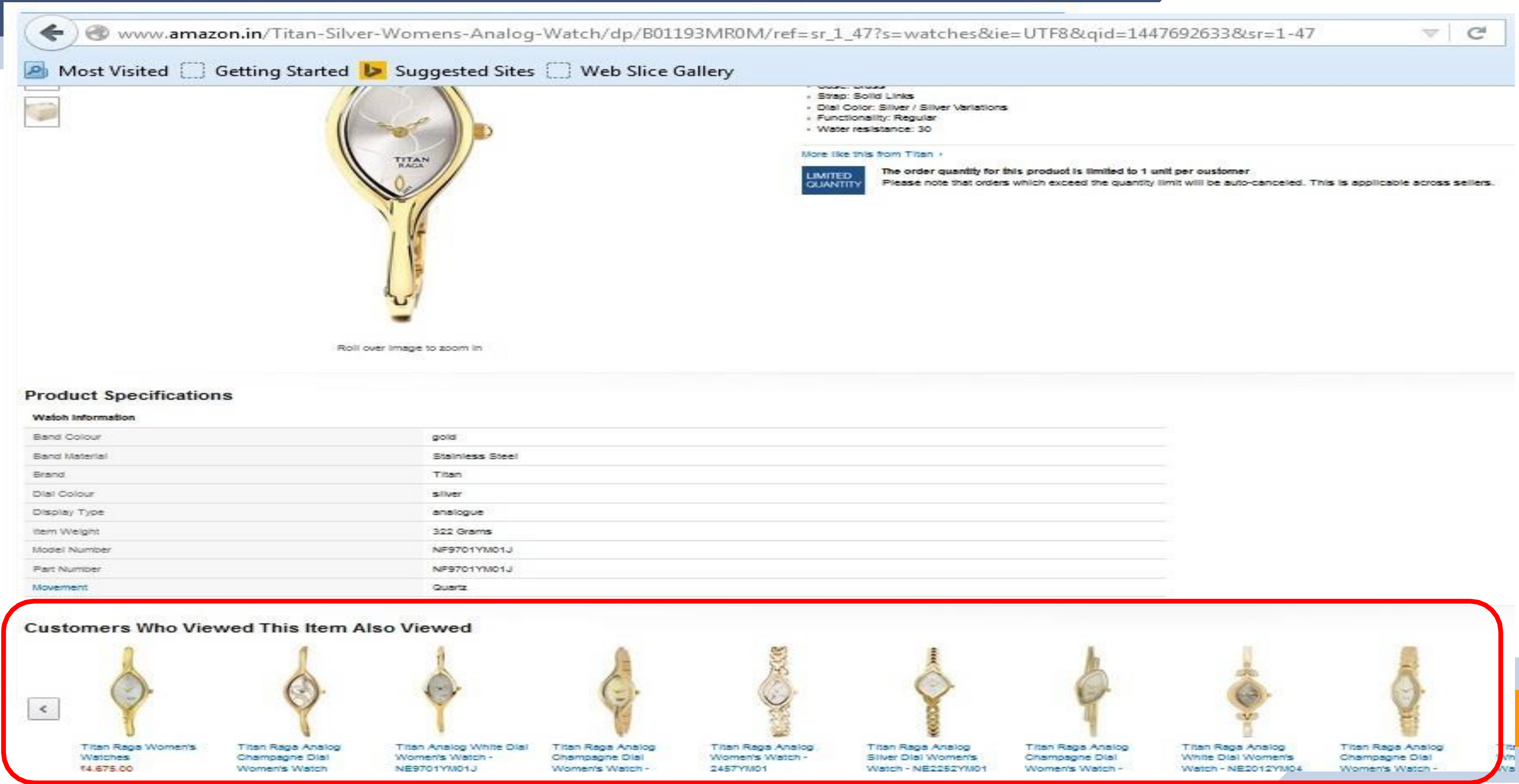
Supervised Learning & ANN

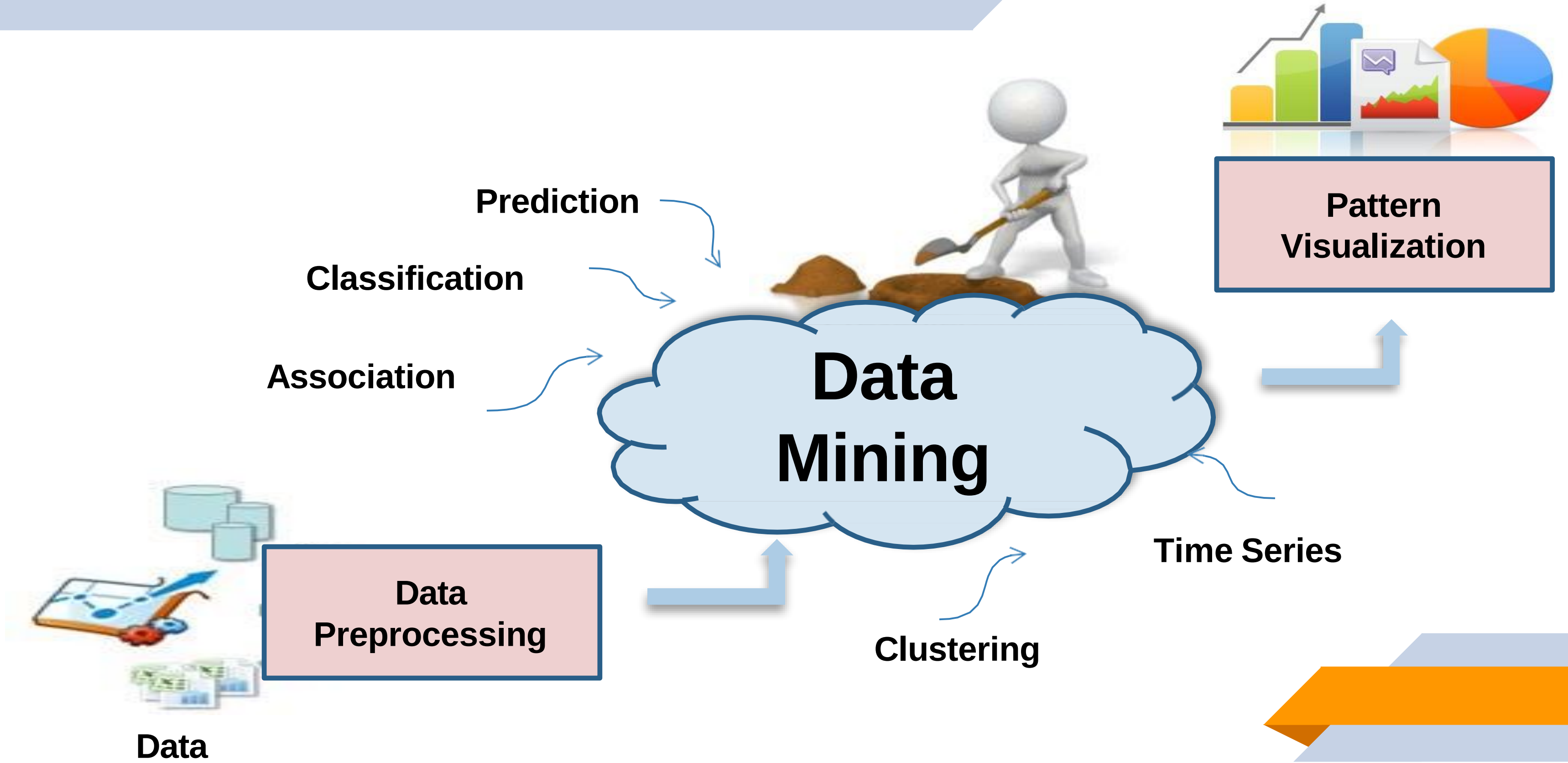
Exploration in R

Raman pal

Data Mining - Real Time Eg.



Phases in Data Analysis

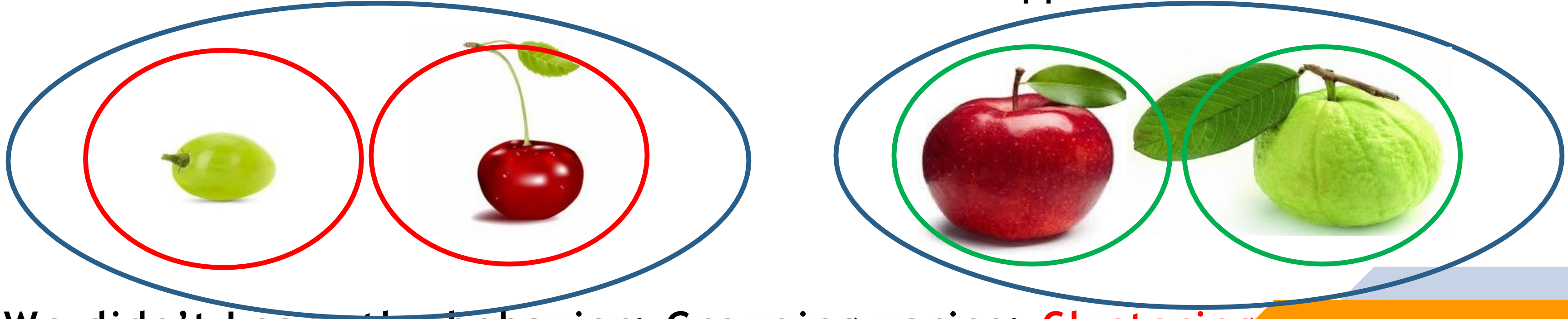


Data Mining ?

- Data mining is a process of extracting
  - Interesting
  - Trivial
  - Implicit
  - Previously unknown &
  - Potentially Useful Patterns /
  - Knowledge from Large Data sets

Descriptive (UnSupervised) Data Mining

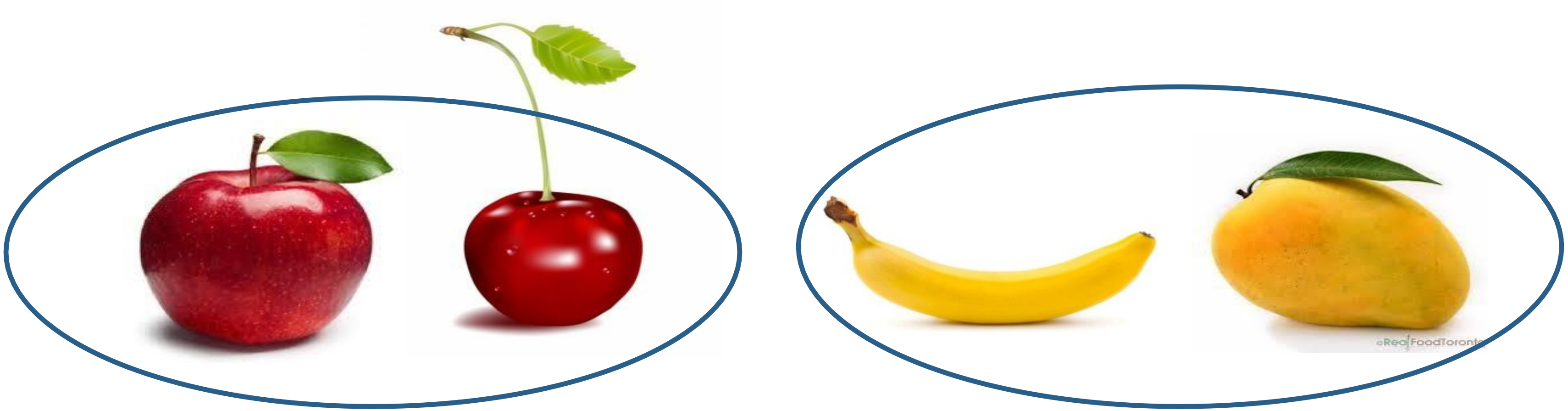
- **Group the following fruits**
- **Apple Guava Grape Cherry**
- According to Physical character, size:
  - **RED COLOR AND BIG SIZE** : Apple.
  - **RED COLOR AND SMALL SIZE** : Cherry
  - **GREEN COLOR AND BIG SIZE** : Guava
  - **GREEN COLOR AND SMALL SIZE** : Grape
- Arrange them base on the color:
  - **Size 1 GROUP** : **Grape & cherry.**
  - **Size 2 GROUP** : **Guava & Apple.**



- **We didn't Learn the behavior; Grouping varies; Clustering**

Predictive (Supervised) Data Mining

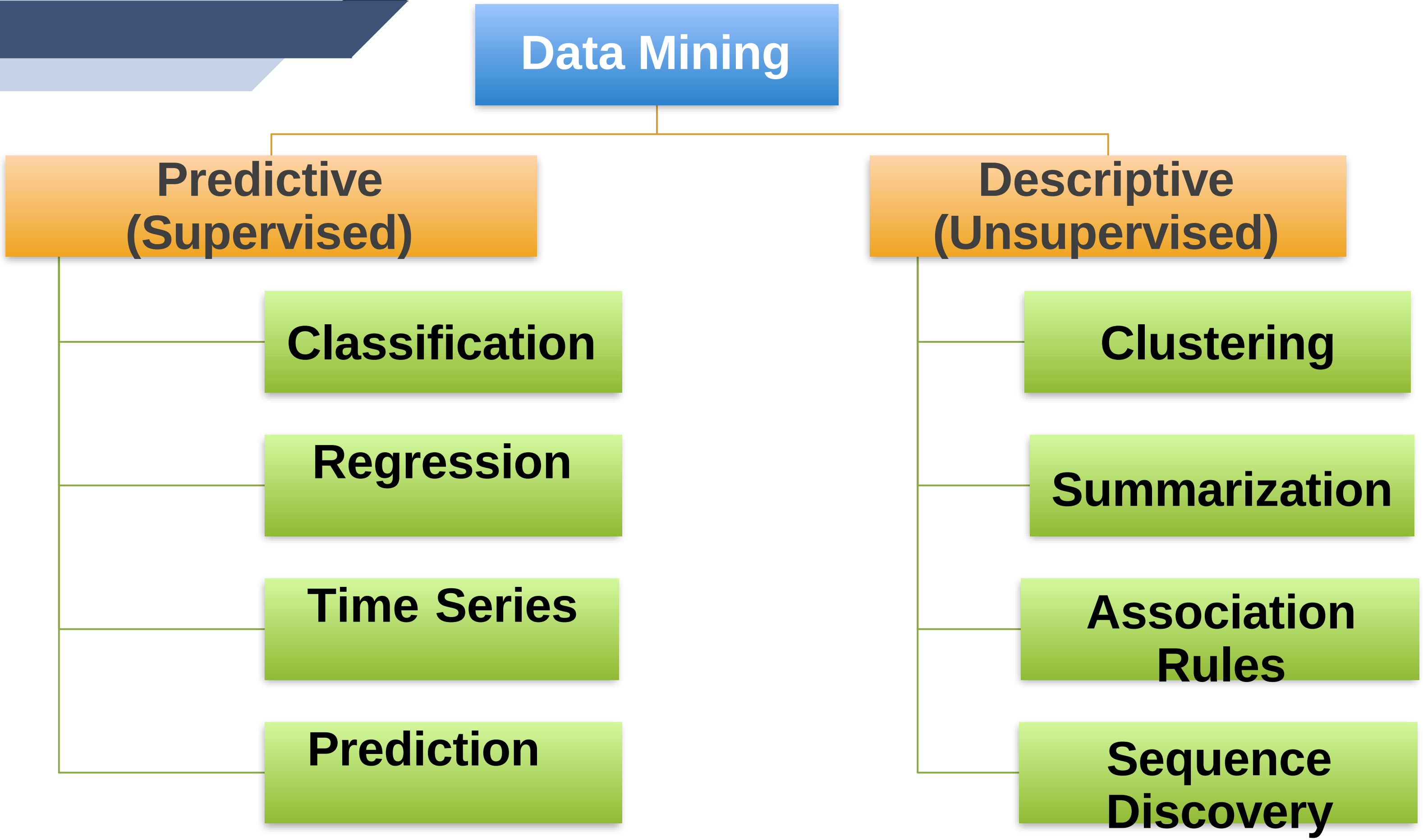
- Group the following fruits
- Apple Banana Mango Cherry



- Already Learnt the behavior; Group confidently; Classificati

Mining Tasks

- Identify a whether a patient has a specific disease or not based on Symptoms –
  - **Predictive**
- Amazon recommends some products which are bought together
  - **Descriptive**

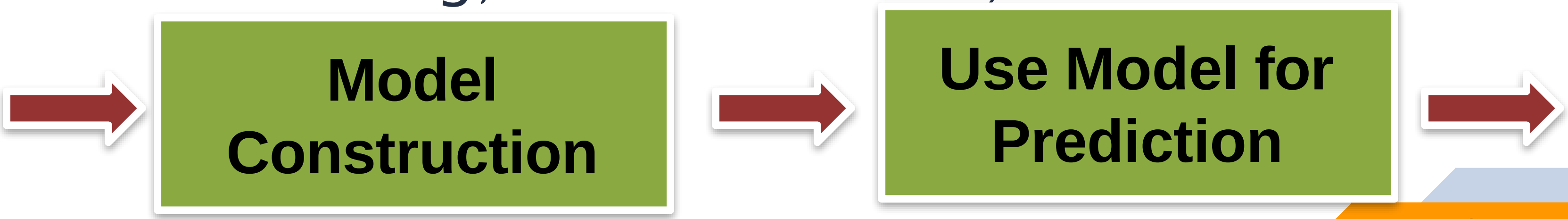

Data Mining
Predictive (Supervised)
Classification
Regression
Time Series
Prediction
Descriptive (Unsupervised)
Clustering
Summarization
Association Rules
Sequence Discovery

“Once you stop learning, you start dying”

Albert Einstein

CLASSIFICATION

- Predicts unknown class label
- Classifies the data based on training set
- <u>Applications:</u>
  - Credit Card Approval; Medical Diagnosis; Target Marketing; Fraud Detection;



CLASSIFICATION

1. Given a collection of records (Training set)
   - Each Record contains a **Set of Attributes**
   - One of the attributes is the **Class (Label)**
2. Find a **Model** for Class attribute as a function of the values of other attributes.
3. **Goal:** Previously unseen records should be assigned a class as accurately as possible.
4. A test set is used to determine the accuracy of the model.

CLASSIFICATION

Usually, the given data set is divided into Training and Test sets

Training Data set

Build the Model

Testing Data set

Validate the Model

Examples of Classification

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc

Classification Techniques

- Decision Tree based Methods
- Naïve Bayes and Bayesian Belief Networks
- Artificial Neural Networks (ANN)
- Support Vector Machines (SVM)
- Rule-based Methods
- Memory based reasoning

Classification - Model Construction

Training Data

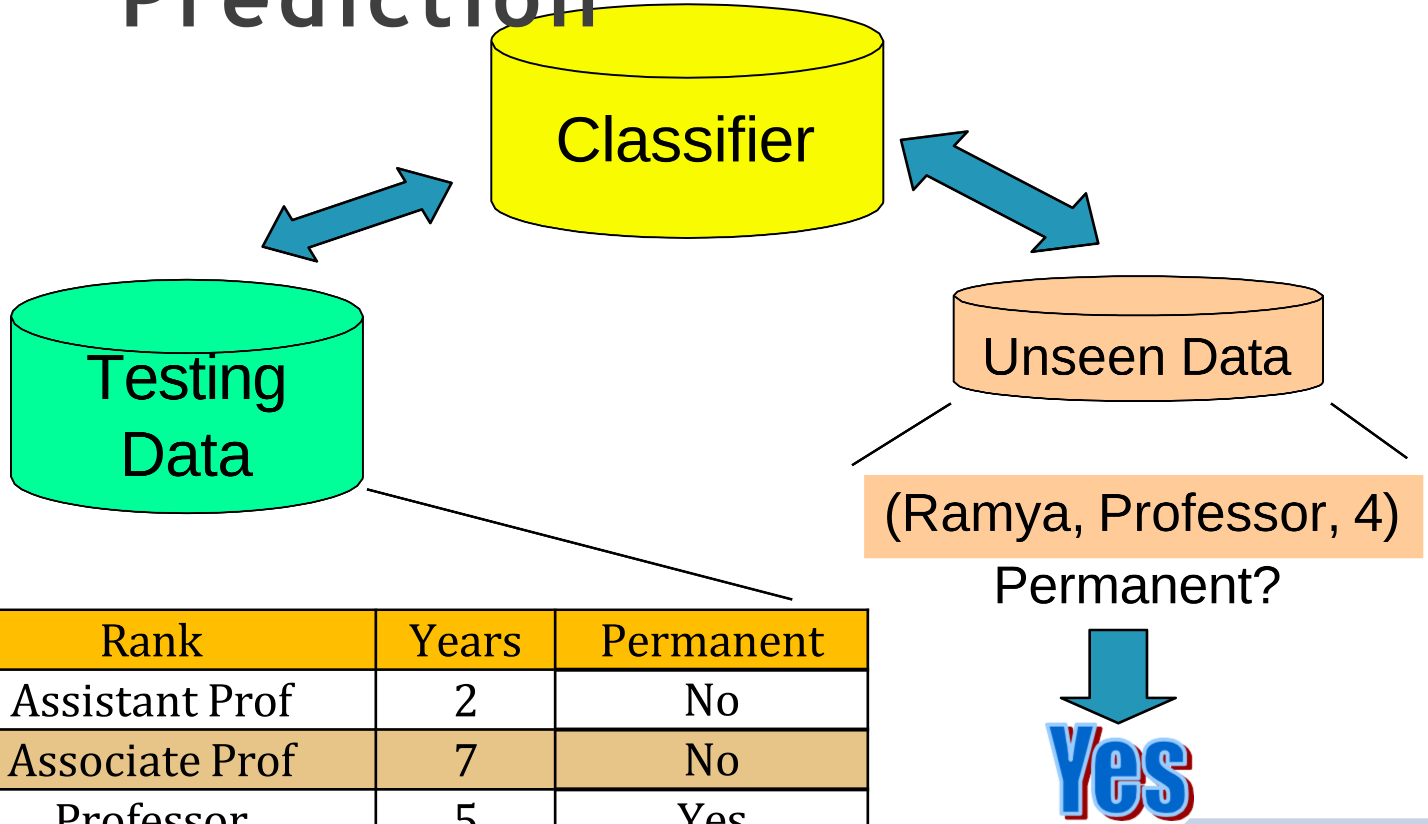
Classification Algorithms

Classifier (Model)

Name	Rank	Years	Permanent
Arun	Assistant Prof	3	No
Vijay	Assistant Prof	7	Yes
Rohan	Professor	2	Yes
Kumar	Associate Prof	7	Yes
Rahul	Assistant Prof	6	No
Raj	Associate Prof	3	No

IF Rank = ‘Professor’
OR Years > 6
THEN Permanent = ‘Yes’

Classification - Model in Prediction

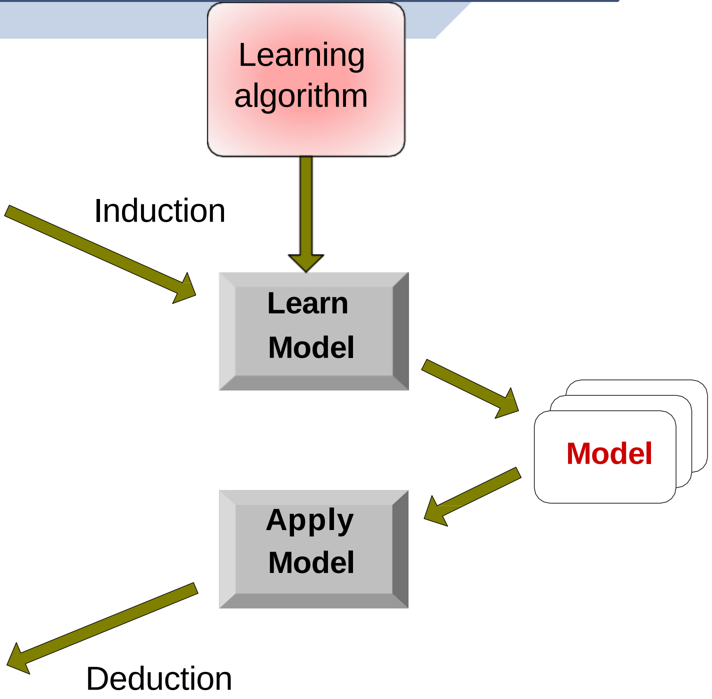


Name	Rank	Years	Permanent
Vignesh	Assistant Prof	2	No
Anand	Associate Prof	7	No
Chirstina	Professor	5	Yes
Parul	Assistant Prof	7	Yes

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



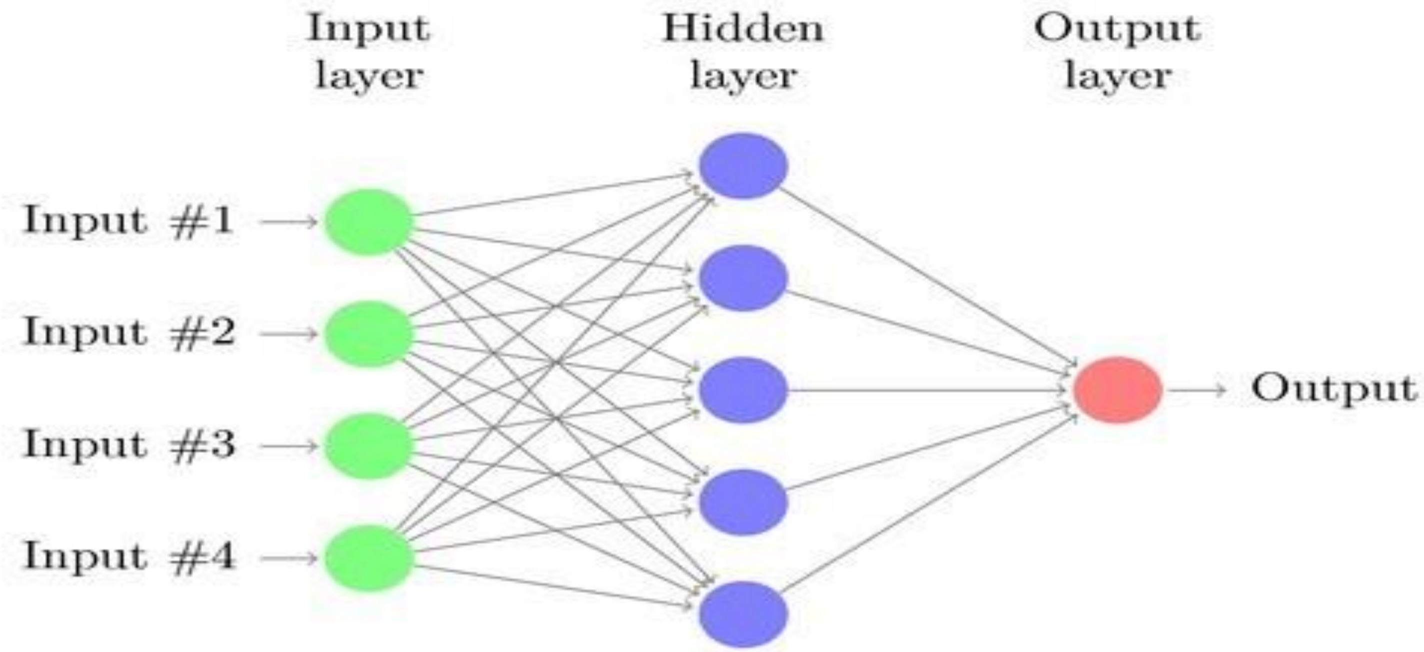
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Artificial Neural Network

- Artificial Neural Network was Introduced to Model the Way in which the human brain Performs a Particular Task or Function of Interest
- These networks
  - Learn from experience
  - Generalize from previous examples to new ones and
  - Abstract essential characteristics from inputs containing irrelevant data.
- The Feed-forward Artificial Neural Network (FNN) is used for classification of the Web Pages
- The training of the network is a **classification problem** where the weights are incremented or decremented in

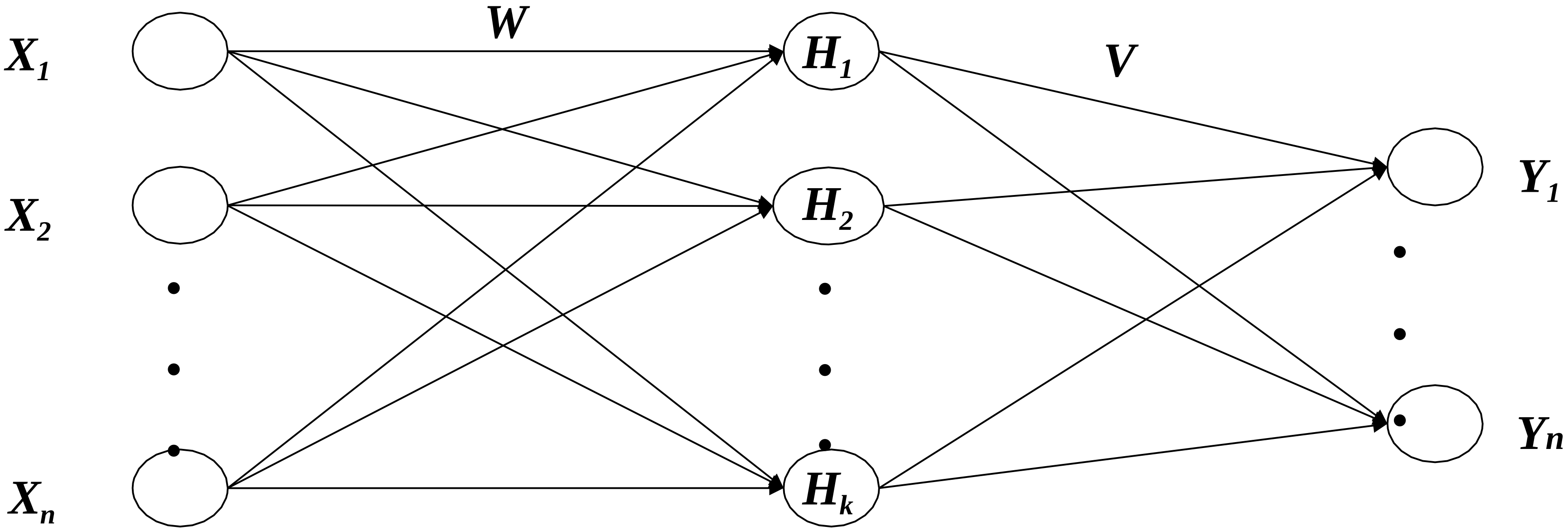
Artificial Neural Networks



Feedforward Neural Network

- Consisting of a number of neurons which are connected by weighted links.
- Neurons are organized in several layers namely
  - An input layer
  - One or more hidden layers and
  - An output layer.
- Input layer receives an external input, and passes it via weighted connections to the units in the first hidden layer.

Feedforward Neural Network



- The neurons in that hidden layer compute their activations and pass them to neurons in succeeding layer and so on.

- Finally neurons in the output layer will give output for the input value.

- In this network, the signal travel in only one direction i.e., no feedback and thus the output of any layer does not affect the same layer.

Training

- It can be viewed as a nonlinear optimization problem in which the goal is to find a set of weights that minimizes the network errors.
- Training of the adaptable weights of the network may be performed for the whole network or by proceeding in a layer–by–layer manner.
- Training the network may be
  - supervised or unsupervised training method.

1. Supervised Training

It requires **training pair** consisting of **input vector** and **desired output vector**.

- This performs **training at offline** i.e., the training phase of the network is distinct from operation phase.

- This is the most frequently used technique in the field of network.

2. Unsupervised Training

- It requires **no target vector** for the output, it solely has only **input vectors.**

- This performs **training at online** i.e., the training and operation phase of the network occurs at the same time.

- The training algorithm modifies network weights to produce output vectors that are consistent.

Backpropagation(BP) Algo rithm

- This is a basic Supervised training algorithm for training multi-layered feedforward neural networks.
- The basic idea of this algorithm is the **repeated application of the chain rule to compute the influence of each weight** in the network with respect to an arbitrary error function.
- The problem associated with this method is slow convergence and local minima entrapment.

Error

$$MSE = (1/N)(\bullet_{j=1,N} \bullet_{i=1,m} (y_i - d_i)^2)$$

where,

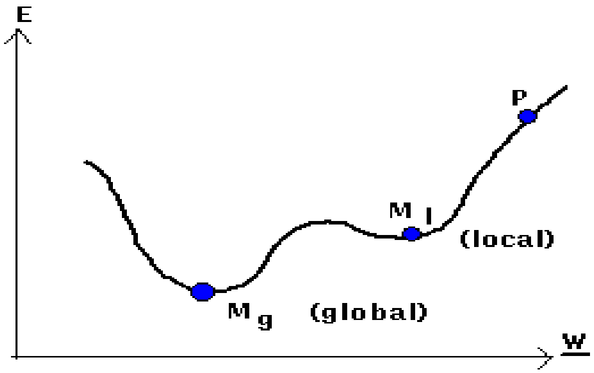
y – represents output layer neuron output

d – represents desired value for that.

Local & Global Minimum

The smallest overall value of a set , function over its entire range is **"Global Minimum"**

- A relative minimal value within some neighborhood that may not be a Global minimum is **"Local Minimum"**



- Local minimum corresponds to a partial solution for a network in response to the training data.

Neural Network: Classification

File: creditset

```
install.packages("neuralnet")
library(neuralnet)
creditset<-read.csv("E:/Materials/SIT Tumkur/Session V Classification/creditset.csv")
View(creditset)
```

```
   There is a binary version available (and will be installed) but
   the source version is later:
          binary source
neuralnet   1.32   1.33

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.1/neuralnet_1.32.z
ip'
Content type 'application/zip' length 58727 bytes (57 Kb)
opened URL
downloaded 57 Kb

package 'neuralnet' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\inurture1\AppData\Local\Temp\Rtmp8UzOWw\downloaded_packages
> |
```

BP Algorithm

1. Generate random numbers for the weight vectors in the range $[-1.5 \;\; 1.5]$
2. Feed input, For each input pattern

   compute output of the feedforward network, find the sum of absolute covariances using the formula,

   $$F_j = (1/N)\sum_{k=1}^{M} \left| \sum_{t=1}^{N} (y_{j,t} - y_j)(e_{k,t} - e_k) \right|, \; j=1,\ldots H$$

3. Repeat steps 1 and 2 for H candidate hidden units.

Algorithm

5. For each output layer

Calculate the output error and

Calculate the nonlinear output error using (5) & (6)

$$e^l_{1j} = f^{-1}(d^l_j) - net^l_j \qquad (5)$$

$$e^l_{2j} = (d^l_j) - y^l_j \qquad (6)$$

6. Calculate the weights to be added for the output layer using (8)

$$\Delta w^l_{ji} = \mu\lambda e^l_{1j} y_i^{l-1} + \mu f^{1}(net^l_j) e^l_{2j} y_i^{l-1} \qquad (8)$$

$$w_{ji} = w_{ji} + \Delta w_{ji} \qquad (13)$$

7. Update the weights of the output layer using (13)

Algorithm

8. Calculate the weights to be added for the hidden layer using (10)

$$\Delta w^{l}_{ji} = \mu \lambda e^{l}_{1j} y^{l-1}_{i} + \mu\ f^{1}(net^{l}_{j}) e^{l}_{2j} y^{l-1}_{i} \qquad (8)$$

9. Update the weights of the hidden layer using (13)

$$w_{ji} = w_{ji} + \Delta w_{ji} \qquad (13)$$

10. Repeat steps to change weights until the desired accuracy is obtained

Bring neuralnet in action

```
> library(neuralnet)
Loading required package: grid
Loading required package: MASS
Warning message:
package 'neuralnet' was built under R version 3.1.3
>
```

Create training and testing data sets

```
> trainset<- creditset[1:800,]
> testset<- creditset[801:2000,]
```

Machine Learn

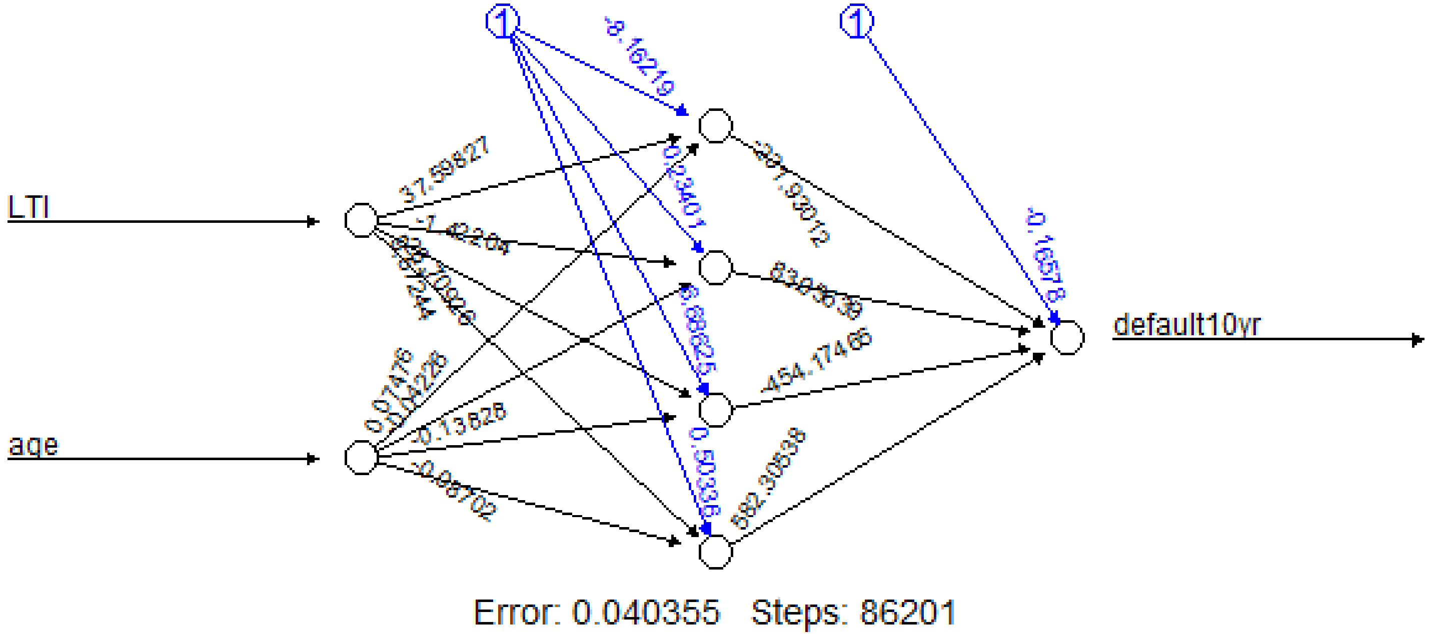
```
> trainset<- creditset[1:800,]
> testset<- creditset[801:2000,]
```

Let machine learn

Creditnet <- neuralnet(default10yr~LTI+age, trainset, hidden=4, lifesign="minimal", linear.output=FALSE, threshold=0.1)

```
> creditnet<-neuralnet(default10yr~LTI + age, trainset, hidden = 4, lifesign
 = "minimal", linear.output = FALSE, threshold = 0.1)
hidden: 4    thresh: 0.1    rep: 1/1    steps:   71925  error: 0.08578  time
: 53.05 secs
> creditnet<-neuralnet(default10yr~LTI + age, trainset, hidden = 4, lifesign
 = "minimal", linear.output = FALSE, threshold = 0.1)
hidden: 4    thresh: 0.1    rep: 1/1    steps:   12232  error: 0.3743   time
: 8.01 secs
> creditnet<-neuralnet(default10yr~LTI + age, trainset, hidden = 4, lifesign
 = "minimal", linear.output = FALSE, threshold = 0.1)
hidden: 4    thresh: 0.1    rep: 1/1    steps:    6389  error: 0.75657  time
: 4.23 secs
> creditnet<-neuralnet(default10yr~LTI + age, trainset, hidden = 4, lifesign
 = "minimal", linear.output = FALSE, threshold = 0.1)
hidden: 4    thresh: 0.1    rep: 1/1    steps:     515  error: 11.77955 time
: 0.33 secs
> creditnet<-neuralnet(default10yr~LTI + age, trainset, hidden = 4, lifesign
 = "minimal", linear.output = FALSE, threshold = 0.1)
hidden: 4    thresh: 0.1    rep: 1/1    steps:   86201  error: 0.04036  time
: 56.43 secs
>
```

```
> plot(creditnet, rep = "best")
```



Preparing for results

```
> temp_test<- subset(testset, select = c("LTI", "age"))
> creditnet.results<- compute(creditnet, temp_test)
> results<- data.frame(actual = testset$default10yr, prediction=
creditnet.results$net.result)
```

Results

```
> results[100:115,]
    actual        prediction
900      0   6.361327751e-71
901      0   1.943534917e-32
902      0  3.377113112e-119
903      1   1.000000000e+00
904      0   3.463046655e-17
905      0   3.466677914e-85
906      0   2.505847008e-52
907      1   9.999999317e-01
908      0   5.119846572e-04
909      0   2.671280086e-96
910      0   1.045441835e-14
911      1   1.000000000e+00
912      0  1.804893653e-119
913      1   1.000000000e+00
914      0   9.093234050e-86
915      0   1.139237172e-29
>
```

Lets round them up

```
> results$prediction<- round(results$prediction)
> results[100:115,]
    actual prediction
900      0          0
901      0          0
902      0          0
903      1          1
904      0          0
905      0          0
906      0          0
907      1          1
908      0          0
909      0          0
910      0          0
911      1          1
912      0          0
913      1          1
914      0          0
915      0          0
>
```

Classification Results

```
> table(results$actual, results$prediction)

       0    1
  0 1035    1
  1    0  164
```

Neural Network: Prediction
File: mtcars

```
> mtcars <- read.csv("C:/Users/inurture1/Desktop/session7_NN/mtc
ars.csv")
>    View(mtcars)
> install.packages("neuralnet")

  There is a binary version available (and will be
  installed) but the source version is later:
          binary source
neuralnet   1.32   1.33

trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.1/neur
alnet_1.32.zip'
Content type 'application/zip' length 58727 bytes (57 Kb)
opened URL
downloaded 57 Kb

package 'neuralnet' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\inurture1\AppData\Local\Temp\RtmpeUQOOH\downloa
ded_packages
> library(neuralnet)
Loading required package: grid
Loading required package: MASS
Warning message:
package 'neuralnet' was built under R version 3.1.3
```

Create training and testing data sets

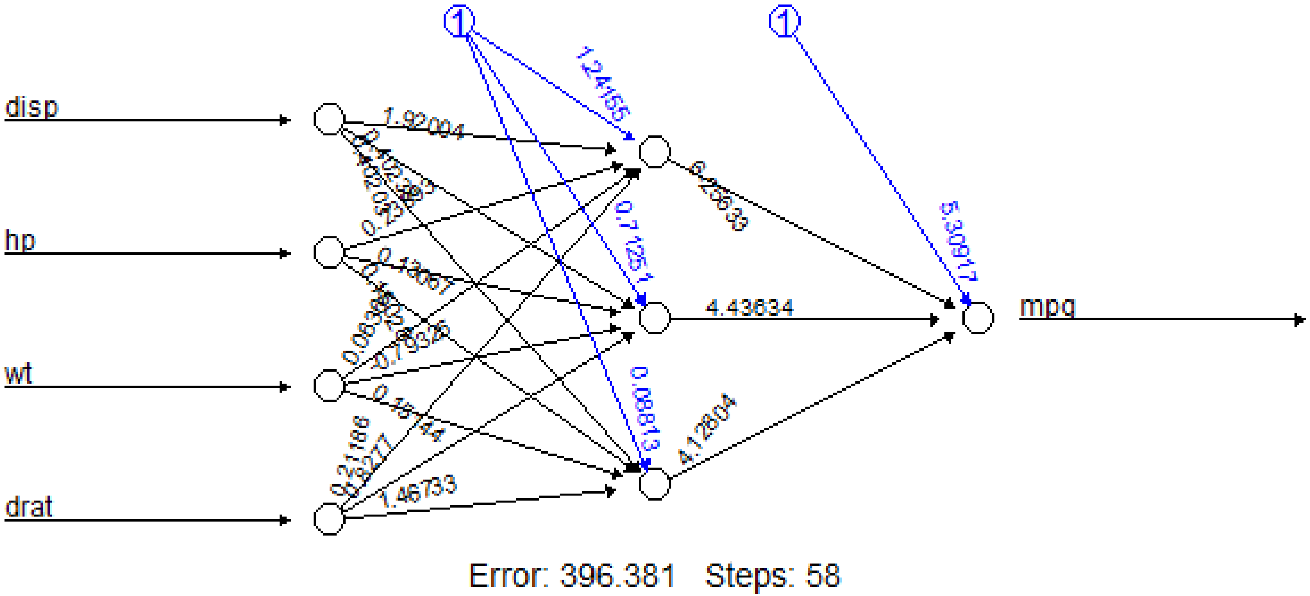
```
> trainset<-mtcars[1:20,]
> testset<-mtcars[21:32,]
```

Build NN Predictive Model

```
> mtcarsnet<-neuralnet(mpg~disp+hp+wt+drat, trainset, hidden = 3
, lifesign = "minimal")
hidden: 3    thresh: 0.01    rep: 1/1    steps:      58 error: 3
96.381          time: 0.02 secs
```

Network Diagram

```
> plot(mtcarsnet, rep = "best")
```



Lets find results

```
> temp_test<-subset(testset, select = c("disp", "hp", "wt", "dra
t"))
> mtcarsnet.results<- compute(mtcarsnet, temp_test)
> results<- data.frame(actual = testset$mpg, prediction = mtcars
net.results$net.result)
```

Results

```
> results
    actual  prediction
21    21.5 20.12987887
22    15.5 20.12987887
23    15.2 20.12987887
24    13.3 20.12987887
25    19.2 20.12987887
26    27.3 20.12987887
27    26.0 20.12987887
28    30.4 20.12987887
29    15.8 20.12987887
30    19.7 20.12987887
31    15.0 20.12987887
32    21.4 20.12987887
```

RMSE

```
> error <- results$actual - results$prediction
> rmse<- function(error)
+ {
+ sqrt(mean(error^2))
+ }
> rmse(error)
[1] 5.27047195
> 
```

Thank you

k.arunadeviselvi@gmail.com

Role of external experts in Curriculum Design

BoS Members	Representatives from University, Alumni, Academia and Industry Dean, Head and all the Faculty members of the department
Meeting	Yearly once
Process	Need Analysis Feedback from Alumni, Students Feedback from Faculty Members, Industry
Impact	Improved Placements - Curriculum in par with Industry Improved Examination Results - Evaluation Criteria More Practical Components

Student Achievements - Research

Paper Presentations

28

Best Paper Awards

06

52

Participation

Seminar / Conference / Meet-ups

22

Paper Publications

Peer Reviewed Journals

Bayesian Classification: Why?

1. Probabilistic learning: Calculate **explicit probabilities** for hypothesis, among the most practical approaches to certain types of learning problems

2. Incremental: Each training example can **incrementally increase/decrease the probability that a hypothesis is correct**. Prior knowledge can be combined with observed data.

3. Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities

4. Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can

Bayesian Theorem: Basics

- Let X be a data sample whose class label is unknown
- Let H be a Hypothesis that X belongs to class C
- For classification problems, determine $P(H|X)$: the probability that the hypothesis holds given the observed data sample X
- $P(H)$: prior probability of hypothesis H (i.e. the initial probability before we observe any data, reflects the background knowledge)
- $P(X)$: probability that sample data is observed
- $P(X|H)$: probability of observing the sample X, given that the hypothesis holds

Bayesian Theorem

- Given training data *X*, posteriori probability of a hypothesis *H*, *P(H|X)* follows the Bayes theorem

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

- Informally, this can be written as

    posteriori = likelihood x prior / evidence

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayesian Classifierr

- A simplified assumption: Attributes are conditionally independent:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- The product of occurrence of say 2 elements x_1 and x_2, given the current class is *C*, is the product of the probabilities of each element taken separately, given the same class $P([y_1,y_2], C) = P(y_1, C) * P(y_2, C)$
- No dependence relation between attributes
- Greatly reduces the computation cost, only count the class distribution.
- Once the probability $P(X|C_i)$ is known, assign *X* to the class with maximum $P(X|C_i) * P(C_i)$

Training Dataset

Class:
C1:buys_computer= 'yes'
C2:buys_computer= 'no'

Data sample
X =(age<=30,
Income=medium,
Student=yes
Credit_rating=
Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier

- Easy to implement
- Good results obtained in most of the cases

Disadvantages

- Assumption: class conditional independence, therefore loss of accuracy
- Practically, dependencies exist among variables
- E.g., hospitals: patients: Profile: age, family history etc

Symptoms: fever, cough etc., Disease: lung cancer, diabetes etc

- Dependencies among these cannot be modeled by Naïve Bayesian Classifier

=0.6

Naïve Bayesian Classifier

P(income="medium" | buys_computer="yes") =

=0.444

P(income="medium" | buys_computer="no") = 2/5 = 0.4

P(student="yes" | buys_computer="yes) =

P(student="yes" | buys_computer="no") =

P(credit_rating="fair" | buys_computer="yes

=0.667

P(credit_rating="fair" | buys_computer="no") = 2/5 =0.4

X=(age<=30 , income =medi

P(X|Ci) :

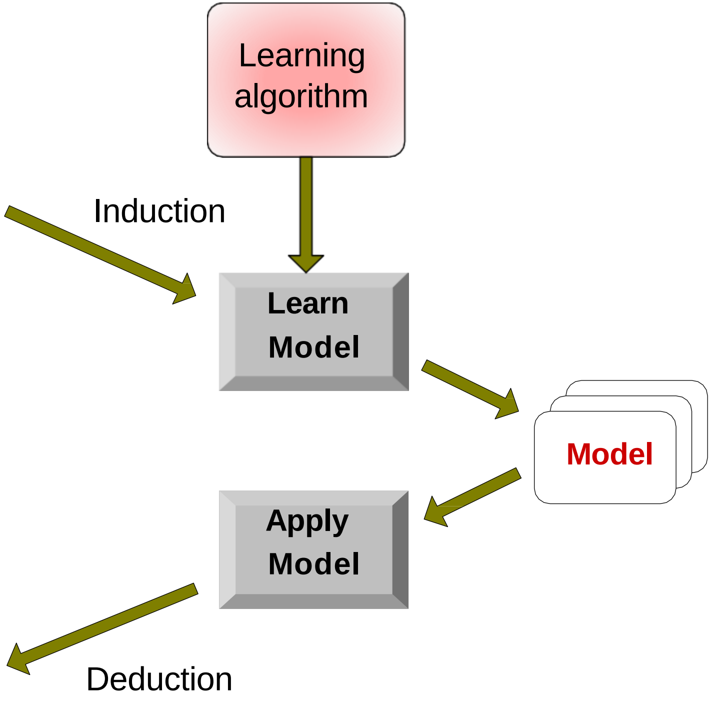
P(X

667

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

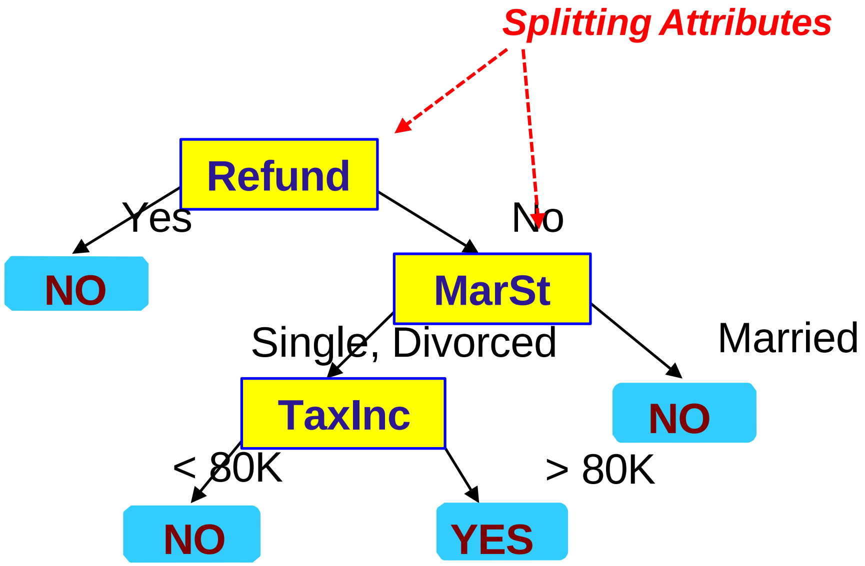
Test Set

Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

Naïve Bayes Classification

```
summary(iris)

classifier<-naiveBayes(iris[,1:4], iris[,5])

mode1=table(predict(classifier, iris[,-5]), iris[,5])

save(mode1,file="mymodel.rda")

pairs(iris[1:4], main = "Iris Data (red=setosa,green=versicolor,blue=virginica)",
    pch = 21, bg = c("red", "green", "blue")[unclass(iris$Species)])

table(predict(classifier, iris[,-5]), iris[,5])
```

```
summary(iris)

classifier<-naiveBayes(iris[,1:4], iris[,5])

mode1=table(predict(classifier, iris[,-5]), iris[,5])

save(mode1,file="mymodel.rda")

pairs(iris[1:4], main = "Iris Data
(red=setosa,green=versicolor,blue=virginica)",

pch = 21, bg = c("red", "green", "blue")
[unclass(iris$Species)])

table(predict(classifier, iris[,-5]), iris[,5])
```