Experiment 4

Prof Meenakshi Garg

7/31/2020

Classification

Implementation and analysis of Classification algorithms like

1. Naive Bayesian,
2. K-Nearest Neighbor
3. ID3
4. C4.5

Naive Bayes • Based on the Bayes theorem
• Predicts based on probabilities from training data
P(B|A) = P(A|B) P(B)/P(A)
Gives posterior probability of 'B' given 'A' using
prior probability of 'B'
prior probability of 'A'
and conditional probability of 'A' given 'B'
• Takes two step approach
– Calculates the posterior probability of the Class given the input - for every class
– Assigns the class with higher posterior probability
• More suited when dimensionality of input is high the - widely used for document classification
• Also good for the multiclass classifications
• Works well with less datasets also, but the assumption that predictor variables are independent should hold ##Naive Bayes setwd("E:/R Orientation")

```
# loading library e1071
library(e1071)
library("klaR")

## Loading required package: MASS

library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

library(ggplot2)
# iris dataset
data(iris)
head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1 5.1 3.5 1.4 0.2 setosa
## 2 4.9 3.0 1.4 0.2 setosa
## 3 4.7 3.2 1.3 0.2 setosa
## 4 4.6 3.1 1.5 0.2 setosa
```



```
## 5 5.0 3.6 1.4 0.2 setosa ## 6 5.4 3.9 1.7 0.4 setosa
```

```r
unique(iris$Species)
```

```
## [1] setosa versicolor virginica
## Levels: setosa versicolor virginica
```

```r
#Plot graph

    pairs(iris[1:4], main="Iris Data (red=setosa,green=versicolor,blue=virginica)", pch=21,
                     bg=c("red","green3","blue")[unclass(iris$Species)])
```

Iris Data (red=setosa,green=versicolor,blue=virginica)

Sepal.Length

Petal.Width

7 5 3 1

0.
4
0.
3
0.
2

Petal.Length

5.
2
5.
1
5.
0

4.5 5.5 6.5 7.5

1 2 3 4 5 6 7
0.5 1.5 2.5

2.0 3.0 4.0

5.
7
0.
6
5.
4

Sepal.Width

```r
# training a naive Bayes model
index = sample(nrow(iris), floor(nrow(iris) * 0.7)) #70/30 split. train =
iris[index,]
test = iris[-index,]

xTrain = train[,-5] # removing y-outcome variable.
yTrain = train$Species # only y.

xTest = test[,-5]
yTest = test$Species

model = train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10)) model
```

Naive Bayes


```
##
## 105 samples
## 4 predictor
## 3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 94, 94, 96, 94, 95, 96, ...
## Resampling results across tuning parameters:
##
## usekernel Accuracy Kappa
## FALSE 0.9401515 0.9092949
## TRUE 0.9401515 0.9092949
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
## parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value. ## The final values
used for the model were fL = 0, usekernel = FALSE and adjust ## = 1.
```

```r
## table() gives frequency table, prop.table() gives freq% table.
prop.table(table(predict(model$finalModel,xTest)$class,yTest))
```

```
## yTest
## setosa versicolor virginica
## setosa 0.31111111 0.00000000 0.00000000
## versicolor 0.00000000 0.31111111 0.00000000
## virginica 0.00000000 0.04444444 0.33333333
```

K nearest Neighbour

```r
df <- data(iris) ##load data
```

```
head(iris) ## see the structure
```

Sepal.Length Sepal.Width Petal.Length Petal.Width Species ## 1 5.1 3.5 1.4 0.2 setosa ## 2 4.9 3.0 1.4 0.2 setosa ## 3 4.7 3.2 1.3 0.2 setosa ## 4 4.6 3.1 1.5 0.2 setosa ## 5 5.0 3.6 1.4 0.2 setosa ## 6 5.4 3.9 1.7 0.4 setosa

```
##Generate a random number that is 90% of the total number of rows in dataset. ran <-
sample(1:nrow(iris), 0.9 * nrow(iris))
##the normalization function is created
nor <-function(x) { (x -min(x))/(max(x)-min(x)) }

##Run nomalization on first 4 coulumns of dataset because they are the predictors iris_norm <-
as.data.frame(lapply(iris[,c(1,2,3,4)], nor))

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 :0.4406 Mean :0.4675 Mean :0.45806
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

```
##extract training set
iris_train <- iris_norm[ran,]
##extract testing set
iris_test <- iris_norm[-ran,]
##extract 5th column of train dataset because it will be used as 'cl' argument in knn function.
iris_target_category <- iris[ran,5]
##extract 5th column if test dataset to measure the accuracy
iris_test_category <- iris[-ran,5]
##load the package class
library(class)
##run knn function
pr <- knn(iris_train,iris_test,cl=iris_target_category,k=13)

##create confusion matrix
tab <- table(pr,iris_test_category)

##this function divides the correct predictions by total number of predictions that tell us how accura

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

[1] 86.66667