621 Assignment 3

Brett Burk

Sunday, March 01, 2015

Contents

Assumptions: I am assuming this assignment is more about working through creating the algorithms than the results.

1

New weights are created

```r
# Reading in the data
folder <- "C:\\Users\\Brett\\Dropbox\\CUNY\\621\\Week3\\"
data.train <- read.csv(paste0(folder, "sample-training-data-naive-bayes.csv"))
data.test <-  read.csv(paste0(folder, "sample-testing-data-naive-bayes.csv"))

# Seperating the variable of interest
var.interest <- data.train[,4]
training.vars <- data.train[,1:3]

# Creates the weights for Naive Bayes as a list of lists
create.weights <- function(training.vars, var.interest){
  split.data <- split(training.vars, var.interest)
  weights <- list()
  for(i in names(split.data)){
    temp.total <- nrow(split.data[[i]])
    weights[[i]] <- list()
    for(j in colnames(split.data[[i]])){
      weights[[i]][[j]] <- table(split.data[[i]][[j]])/temp.total
    }
  }
  return(weights)
}

new.weights <- create.weights(training.vars, var.interest)
new.weights
```

```
## $`-1`
## $`-1`$Cat1
##
##         A         B
## 0.3717949 0.6282051
##
## $`-1`$Cat2
##
##         G         H
## 0.4871795 0.5128205
##
## $`-1`$Cat3
##
##         X         Y         Z
## 0.5384615 0.2692308 0.1923077
##
##
## $`1`
## $`1`$Cat1
##
##         A         B
## 0.6463415 0.3536585
##
## $`1`$Cat2
##
##         G         H
## 0.5243902 0.4756098
##
## $`1`$Cat3
##
##         X         Y         Z
## 0.1829268 0.3414634 0.4756098
```

The weights are first created

```
test.weights <- function(test.data, weights){
  results <- test.data[ncol(test.data)]
  test.data <- test.data[,1:(ncol(test.data)-1)]
  weighted.df <- data.frame(matrix(data = 0, ncol = length(weights) + 1, nrow = nrow(test.data)))
  colnames(weighted.df) <- c(names(weights), 'prediction')
  for(result in unique(unlist(results))){
    for(row in 1:nrow(test.data)){
      temp.total <- 1
      for(i in names(test.data)){
        temp.total <- temp.total * weights[[as.character(result)]][[i]][test.data[i][row,]]
      }
    weighted.df[as.character(result)][row,] <- temp.total
    }
  }
  for(row in 1:nrow(weighted.df)){
    weighted.df['prediction'][row,] <- names(which.max(weighted.df[row,1:(ncol(weighted.df)-1)]))
  }
  predictions <- weighted.df['prediction']
  accuracy <- sum(predictions == results)/nrow(results)
  return(accuracy)
}
```

The weights are tested against the training and test data sets

```
test.weights(data.train, new.weights)
```

```
## [1] 0.65625
```

```
test.weights(data.test, new.weights)
```

```
## [1] 0.65
```

Both are predicted with about 65% accuracy.

My Naive Bayes algorithm

```r
naive.bayes <- function(train.data, test.data){
  weights <- create.weights(train.data[,1:(ncol(train.data)-1)], train.data[ncol(train.data)])
  weighted.df <- data.frame(matrix(data = 0, ncol = length(weights) + 1, nrow = nrow(test.data)))
  colnames(weighted.df) <- c(names(weights), 'prediction')
  for(result in names(weights)){
    for(row in 1:nrow(test.data)){
      temp.total <- 1
      for(i in names(test.data)){
        temp.total <- temp.total * weights[[as.character(result)]][[i]][test.data[i][row,]]
      }
      weighted.df[as.character(result)][row,] <- temp.total
    }
  }
  for(row in 1:nrow(weighted.df)){
    weighted.df['prediction'][row,] <- names(which.max(weighted.df[row,1:(ncol(weighted.df)-1)]))
  }
  return(cbind(test.data, weighted.df['prediction']))
}
```

The code does not perform perfectly, but still a 65% accuracy is definitely better than just guessing. (it predicts both on the training data set and the testing data set)

2

```r
knn.train <- read.csv(paste0(folder, "sample-training-data-nearest-neighbor.csv"))
knn.test <- read.csv(paste0(folder, "sample-testing-data-nearest-neighbor.csv"))

knn.train[,1:3] <- scale(knn.train[,1:3])
knn.test[,1:3] <- scale(knn.test[,1:3])

k.near.neighbors <- function(unclassified, points){
  point.classification <- points[,ncol(points)]
  points <- points[,1:(ncol(points)-1)]
  classifications <- array(data = 0, dim = nrow(unclassified))
  k <- 3
  for(row in 1:nrow(unclassified)){
    distances <- data.frame(matrix(data=0, ncol=1, nrow=nrow(points)))
    for(point in 1:nrow(points)){
      distances[point,1] <- sum((points[point,] - unclassified[row,])^2)
    }
    knns <- order(distances, decreasing=F)[1:k]
    classifications[row] <- names(which.max(table(point.classification[knns])))
  }
  return(cbind(unclassified, classifications))
}

classif <- k.near.neighbors(knn.test[,1:3], knn.train)
sum(as.numeric(classif[,4]) == knn.test[,4])/length(classif[,4])
```

```
## [1] 0.8
```

80% prediction accuracy is pretty good.

3

Submitted as brettmburk@gmail.com

4

e1071

```r
library(e1071)
folder <- "C:\\Users\\Brett\\Dropbox\\CUNY\\621\\Week3\\"
data.train <- read.csv(paste0(folder, "sample-training-data-naive-bayes.csv"))
data.test <-  read.csv(paste0(folder, "sample-testing-data-naive-bayes.csv"))

model <- naiveBayes(as.factor(Class) ~ ., data = data.train)
pred <- predict(model, data.test[,1:3])

accuracy <- sum(pred == data.test[,4])/length(pred)
accuracy
```

```
## [1] 0.65
```

Exactly the same accuracy as my version

pred.knn

```r
library(class)
folder <- "C:\\Users\\Brett\\Dropbox\\CUNY\\621\\Week3\\"
knn.train <- read.csv(paste0(folder, "sample-training-data-nearest-neighbor.csv"))
knn.test <- read.csv(paste0(folder, "sample-testing-data-nearest-neighbor.csv"))

pred.knn <- knn(knn.train[,1:3], knn.test[,1:3], cl=as.factor(knn.train[,4]), k=3)

accuracy <- sum(pred.knn == knn.test[,4])/length(pred.knn)
accuracy
```

```
## [1] 0.725
```

My result was better, but as this is a n=1 trial, no real assumptions can/should be made, this may also have to do with feature scaling

5

Jury data set

```r
jury.train <- read.csv(paste0(folder, "jury-training-data.csv"))
jury.public <- read.csv(paste0(folder, "jury-learning-data-public.csv"))
jury.private <- read.csv(paste0(folder, "jury-learning-data-private.csv"))

jury.public <- jury.public[complete.cases(jury.public),]
jury.predictions <- naive.bayes(jury.train, jury.public[,-5])
accuracy <- sum(jury.predictions[,5] == jury.public[,5])/length(jury.predictions[,5])
accuracy
```

```
## [1] 0.6302521
```

```r
model <- naiveBayes(as.factor(tendency) ~ ., data = jury.train)
jury.pred <- predict(model, jury.public[,-5])
accuracy <- sum(jury.pred == jury.public[,5])/length(jury.pred)
accuracy
```

```
## [1] 0.6302521
```

It came up with exactly the same accuracy as my naive bayes implementation (although it runs significantly faster).

6

Pima data set

```r
pima.train <- read.csv(paste0(folder, "pima-training-data.csv"))
pima.test <- read.csv(paste0(folder, "pima-learning-data-public.csv"))

pima.train[,1:8] <- scale(pima.train[,1:8])
pima.test[,1:8] <- scale(pima.test[,1:8])

my.preds <- k.near.neighbors(pima.test[,-9], pima.train)
accuracy <- sum(my.preds[,9] == pima.test[,9])/length(my.preds[,9])
accuracy
```

```
## [1] 0.7037037
```

```r
pred.knn <- knn(pima.train[,1:8], pima.test[,1:8], cl=as.factor(pima.train[,9]), k=3)
accuracy <- sum(pred.knn == pima.test[,9])/length(pred.knn)
accuracy
```

```
## [1] 0.7037037
```

Both models came up with the same accuracy–that of ~ 70%