Knn

Chase Enzweiler

11/6/2017

K-Nearest-Neighbors

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.3.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(class)
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.3.2
```

```
# my_knn()
my_knn <- function(X_train, X_test, Y_train, k){

  # make a matrix of distances of columns test observations rows are train obs

  distances <- matrix(0, ncol = dim(X_test)[1], nrow = dim(X_train)[1])

  for (i in 1:dim(X_test)[1]){

    for (j in 1:dim(X_train)[1]){

    distances[j,i] <- sqrt(sum((X_train[j,] - X_test[i,])^2))

    }
  }

  min_indices <- matrix(0, ncol = dim(X_test)[1], nrow = k)



  # find the k smallest distances

  for (i in 1:k){

    for (j in 1:dim(X_test)[1]){

    min_indices[i,j] <- which.min(distances[,j])

    distances[which.min(distances[,j]),j] <- NA
```

```
    }
  }



  # get the classes of the k minimum indexed distances

  obs <- c()

  for (i in 1:dim(min_indices)[2]){

    obs[i] <- names(which.max(table(Y_train[min_indices[,i]])))

  }

  return(obs)
}
```

test my_knn

```
set.seed(1)
train_idx <- sample(nrow(iris), 90)
test_set <- iris[-train_idx,]
train_set <- iris[train_idx,]

as.factor(my_knn(train_set[,-5], test_set[,-5],train_set[,5], 5))
```

```
##  [1] setosa     setosa     setosa     setosa     setosa     setosa
##  [7] setosa     setosa     setosa     setosa     setosa     setosa
## [13] setosa     setosa     setosa     setosa     setosa     setosa
## [19] setosa     setosa     setosa     versicolor versicolor versicolor
## [25] versicolor versicolor versicolor versicolor versicolor versicolor
## [31] versicolor versicolor versicolor versicolor versicolor versicolor
## [37] versicolor virginica  virginica  virginica  virginica  virginica
## [43] virginica  virginica  virginica  virginica  virginica  virginica
## [49] virginica  virginica  versicolor virginica  virginica  virginica
## [55] virginica  versicolor virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
knn(train_set[,-5], test_set[,-5],train_set[,5],k= 5)
```

```
##  [1] setosa     setosa     setosa     setosa     setosa     setosa
##  [7] setosa     setosa     setosa     setosa     setosa     setosa
## [13] setosa     setosa     setosa     setosa     setosa     setosa
## [19] setosa     setosa     setosa     versicolor versicolor versicolor
## [25] versicolor versicolor versicolor versicolor versicolor versicolor
## [31] versicolor versicolor versicolor versicolor versicolor versicolor
## [37] versicolor virginica  virginica  virginica  virginica  virginica
## [43] virginica  virginica  virginica  virginica  virginica  virginica
## [49] virginica  virginica  versicolor virginica  virginica  virginica
## [55] virginica  versicolor virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

KNN cross validation

```r
find_k_CV <- function(X_train, Y_train, k = 1:10, nfold = 10){

  folds <- createFolds(X_train[,1], k = nfold)
  #columns are the number of neighbors, rows are the fold
  acc_mat <- matrix(0, nrow = length(folds), ncol = length(k))

  for (i in 1:length(folds)){

    train <- X_train[-folds[[i]],]

    train_y <- Y_train[-folds[[i]]]

    test <- X_train[folds[[i]],]

    test_y <- Y_train[folds[[i]]]

    for (j in k){

      pred <- my_knn(train, test, train_y, k = j)

      accuracy <- mean((test_y == pred)*1)

      acc_mat[i,j] <- accuracy

    }

    final <- colMeans(acc_mat)

  }

  final <- colMeans(acc_mat)

  return(which.max(final))

}
```

Output the cross validation function

```r
find_k_CV(train_set[,-5], train_set[,5], k = 1:5, nfold = 3)
```

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

Comparisons

```r
# code provided to generate datasets
set.seed(100)

expit <- function(x) {
  exp(x) / (1 + exp(x))
  }


gen_datasets <- function() {
```

```r
id <- diag(c(1, 1))

df1 <- data.frame(y=factor(rep(c(0, 1), each=50)), rbind(rmvnorm(50, mean=c(0, 0), sigma = id), rmvnorm

covmat <- matrix(c(1, -0.5, -0.5, 1), nrow=2)

df2 <- data.frame(y=factor(rep(c(0, 1), each=50)), rbind(rmvnorm(50, mean=c(0, 0), sigma = covmat), rmv

mu <- c(0, 0); sigma <- matrix(c(1, 1/2, 1/2, 1), 2); nu <- 4

n <- 50 # Number of draws

x_first <- t(t(mvrnorm(n, rep(0, length(mu)), sigma) * sqrt(nu / rchisq(n, nu))) + mu)

mu <- c(1, 1); sigma <- matrix(c(1, 1/2, 1/2, 1), 2); nu <- 4

n <- 50 # Number of draws

x_second <- t(t(mvrnorm(n, rep(0, length(mu)), sigma) * sqrt(nu / rchisq(n, nu))) + mu)

df3 <- data.frame(y=factor(rep(c(0, 1), each=50)), rbind(x_first, x_second))

covmat2 <- matrix(c(1, 0.5, 0.5, 1), nrow=2)

df4 <- data.frame(y=factor(rep(c(0, 1), each=50)), rbind(rmvnorm(50, mean=c(0, 0), sigma = covmat2), rm

x <- matrix(rnorm(200), ncol=2)


df5_temp <- data.frame(x ^ 2, x[, 1] * x[, 2])


beta <- c(0, 2, -1, -2)

y <- apply(df5_temp, 1, function(row) {

  p <- expit(sum(c(1, row) * beta))

sample(x=c(0, 1), size=1, prob=c(1-p, p)) })

df5 <- data.frame(y=factor(y), x)
x <- matrix(rnorm(200), ncol=2)
y <- 1 * (x[, 1]^2 + x[, 2]^2 > qchisq(p=0.5, df=2))
df6 <- data.frame(y=factor(y), x)

list(df1, df2, df3, df4, df5, df6)
}
```