STAT562-L3-KNN

Loading data

Iris dataset consists of 50 samples from each of 3 species of Iris (Iris setosa, Iris virginica, Iris versicolor) and a multivariate dataset introduced by British statistician and biologist Ronald Fisher in his 1936 paper. Four features were measured from each sample i.e length and width of the sepals and petals and based on the combination of these four features, we'd like to develop a KNN model to distinguish the species from each other. The data is built-in with R base package.

```
data(iris)
```

Structure of the data

```
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
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...
```

Normalization

The normalization step is optional for this data set because the features (width and length) are measured with the same scale. But you may apply normalize anyway.

```
normalize = function(x) {
return ((x - min(x)) / (max(x) - min(x))) }
iris.normalized=apply(iris[,1:4],2,normalize)
```

Splitting data into train and test data

I did a 70-30 split here.

```
n=nrow(iris)
set.seed(11)
train.index=sample(1:n,size=n*0.7,replace = FALSE)
train.x=iris.normalized[train.index,]
```

```
test.x=iris.normalized[-train.index,]
dim(train.x)
```

```
## [1] 105   4
```

```
dim(test.x)
```

```
## [1] 45  4
```

```
train.y=iris[train.index,5]
test.y=iris[-train.index,5]
```

Apply KNN

We will use the KNN function in class package. Load the library first.

```
library(class)
knn_3  = knn(train = train.x,
                      test = test.x,
                      cl = train.y,
                      k = 3)
```

##Calculate the errors with confusion matrix

Confusiin Matrix

```
cm <- table(test.y, knn_3)
cm
```

```
##             knn_3
## test.y       setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0         15         2
##   virginica       0          1        13
```

Calculate out of test error rates

```
test.error=mean(knn_3 != test.y)
test.error
```

```
## [1] 0.06666667
```

Repeat with different choice of K

```
knn_5  = knn(train = train.x,
                      test = test.x,
                      cl = train.y,
                      k = 5)
test.error=mean(knn_5 != test.y)
test.error
```

```
## [1] 0.04444444
```

```
knn_8  = knn(train = train.x,
                      test = test.x,
                      cl = train.y,
                      k = 8)
test.error=mean(knn_8 != test.y)
test.error
```

```
## [1] 0.02222222
```

```
knn_15  = knn(train = train.x,
                      test = test.x,
                      cl = train.y,
                      k = 15)
test.error=mean(knn_15 != test.y)
test.error
```

```
## [1] 0.04444444
```

```
knn_25  = knn(train = train.x,
                      test = test.x,
                      cl = train.y,
                      k = 25)
test.error=mean(knn_25 != test.y)
test.error
```

```
## [1] 0.08888889
```