K Nearest Neighbors (KNN) Example Using R

Data Used: http://archive.ics.uci.edu/ml/datasets/Abalone

Abalone: http://en.wikipedia.org/wiki/Abalone

The dataset records whether an abalone is a female, infant, or male, three classifications that we will predict using KNN. It also records a number of attributes about an abalone, which can be seen at the UCI website.

Output from R Code (Using RStudio):

```
> table(abalone.actual$V1)  ## These are the outcomes we are going predict:
Female, Infant, Male

  F   I   M 
424 437 531 
> 
> ##  Impliment R's K Nearest Neighbors with 5 nearest observations
> ##  We are using two characteristics so this is similar to the hypothetical
2-dimensional example in class
> ##  The two characteristics are length and diameter
> abalone.kknn <- kknn(V1~V2+V3, abalone.train, abalone.actual, distance = 1,
k = 5)
> 
> ##  Fit the actual data based on the training data and show the results
> fit <- fitted(abalone.kknn)
> "TABLE 1: Five NN, Two Characteristics"
[1] "TABLE 1: Five NN, Two Characteristics"
> table(abalone.actual$V1, fit)
   fit
      F   I   M
  F 157  69 198
  I  87 280  70
  M 193 130 208
> 
> ##  Now do it with the 10 nearest neighbors
> set.seed(12345)  ## Set the random number generator to ensure I get the sam
e sample
> abalone.kknn <- kknn(V1~V2+V3, abalone.train, abalone.actual, distance = 1,
k = 10)
> fit <- fitted(abalone.kknn)
> "TABLE 2: 10 NN, Two Characteristics"
[1] "TABLE 2: 10 NN, Two Characteristics"
> table(abalone.actual$V1, fit)
   fit
      F   I   M
  F 152  75 197
  I  67 312  58
  M 195 125 211
> 
> ##  Now go back to 5 NN, but use all characteristics
> set.seed(12345)  ## Set the random number generator to ensure I get the sam
e sample
> abalone.kknn <- kknn(V1~., abalone.train, abalone.actual, distance = 1, k =
5)
> fit <- fitted(abalone.kknn)
> "TABLE 3: Five NN, All Characteristics"
[1] "TABLE 3: Five NN, All Characteristics"
> table(abalone.actual$V1, fit)
```

```
   fit
      F   I   M
  F 162  60 202
  I  51 324  62
  M 183 111 237
```

Discussion of Results:

For all tables, the predicted classification runs horizontally, while the correct classification runs vertically. As a result, the on-diagonal elements represent correct predictions, while the off-diagonal elements represent erroneous predictions. **We know they are correct because we know the truth**. For example, in TABLE 1, we correctly predict 157 of 424 females (or 37%). In contrast, we incorrectly predict 69 females as infants and 198 females as males.

In TABLE 1, with five nearest neighbors, we do rather poorly for females and males **because they look similar to each other in terms of length and diameter**. In contrast, we do much better for infants **because they look similar to themselves, but different than adults**.

In TABLE 2, we increase the number of nearest neighbors to 10 (that is, we double the number of nearest neighbors). We do a bit worse for females, a bit better for males, and substantially better for infants. This is not surprising, as there is **little to distinguish females from males**. These small changes in accuracy are largely the result of chance. We do better for infants, **but that is no surprise: they look like themselves but different than adults**.

In TABLE 3, we return to five nearest neighbors, but increase to all recorded characteristics (rather than two characteristics). You cannot picture this in your mind because it is greater than 3-dimensional. Compare this to the results in TABLE 1. In each case, we do better at fitting the truth. For adult females and males, however, this is not a marked improvement.

The lesson here is clear. When it is easy to do something in applied data analysis, such as predicting whether or not a particular abalone is an infant, most techniques will capture that ease (in this case, of classification). In contrast, when it is difficult, non-parametric machine learning algorithms such as NN **are not “cure-alls”**. The example I presented in class using the iris data is another example: **those data are easy to classify**. I would submit that you ought to consider this example when you approach any applied problem.

END NOTE: It has been suggested that a classification technique called “Random Forests” may be superior to this naïve non-parametric approach. Below are the results using the Random Forest classifier, results from which are directly comparable to TABLE 3. Setting aside format, I leave it to you to decide whether this is a marked improvement (setting aside computing time).

```
        predicted
observed   F   I   M
       F 188  51 185
       I  33 355  49
       M 171 103 257
```

R Code

```
library(kknn)  ## Activate R's KNN algorithm
abalone <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data",
header=FALSE) ## read in the dataset "Abalone" from UCI Machine Learning Website
attach(abalone)  ## I like to attach files
m <- dim(abalone)[1]  ## Measure the length of the file to split into separate training and actual datasets

## Randomly sample one-third of the dataset for training purposes
set.seed(12345)  ## Set the random number generator to ensure I get the same sample
val <- sample(1:m, size = round(m/3), replace = FALSE, prob = rep(1/m, m))
abalone.train <- abalone[-val,]  ## Assign training dataset
abalone.actual <- abalone[val,]  ## Assign actual dataset

table(abalone.actual$V1)  ## These are the outcomes we are going predict: Female, Infant, Male

##  Impliment R's K Nearest Neighbors with 5 nearest observations
##  We are using two characteristics so this is similar to the hypothetical 2-dimensional example in class
##  The two characteristics are length and diameter
abalone.kknn <- kknn(V1~V2+V3, abalone.train, abalone.actual, distance = 1, k = 5)

##  Fit the actual data based on the training data and show the results
fit <- fitted(abalone.kknn)
"TABLE 1: Five NN, Two Characteristics"
table(abalone.actual$V1, fit)

##  Now do it with the 10 nearest neighbors
abalone.kknn <- kknn(V1~V2+V3, abalone.train, abalone.actual, distance = 1, k = 10)
fit <- fitted(abalone.kknn)
"TABLE 2: 10 NN, Two Characteristics"
table(abalone.actual$V1, fit)

##  Now go back to 5 NN, but use all characteristics
abalone.kknn <- kknn(V1~., abalone.train, abalone.actual, distance = 1, k = 5)
fit <- fitted(abalone.kknn)
"TABLE 3: Five NN, All Characteristics"
table(abalone.actual$V1, fit)
```