Supervised learning – Classification (Demo)

Method of k-Nearest Neighbors (k-NN)

STAT 32950-24620

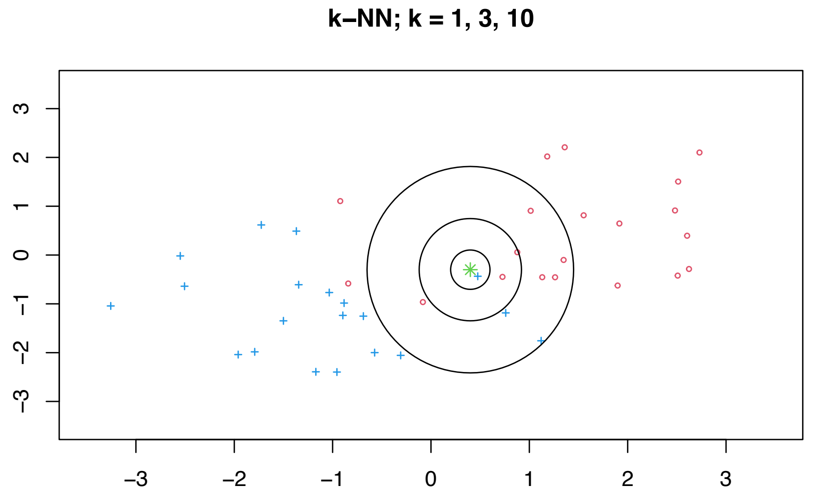
Spring 2023 (4/27)

K-NN Algorithm

- For each test point to be classified,
  find the K nearest samples in the training data
- Classify the point according to the majority vote
  of their class labels

Illustration: How k-NN classifies a new observation

```
# K-NN illustrate by 2-class 2-d plot
library(plotrix)
set.seed(2023)
x1=rnorm(20)+1; y1=rnorm(20)+1
x2=rnorm(20)-1; y2=rnorm(20)-1
plot(x1,y1,xlim=c(-3.5,3.5),ylim=c(-3.5,3.5),
     xlab="",ylab="", col=2,cex=.5)
points(x2,y2,pch=3,col=4,cex=.5)
points(0.4,-.3,pch=8,col=3)
draw.circle(.4,-.3,.2)
draw.circle(.4,-.3,.52)
draw.circle(.4,-.3,.95)
title("k-NN; k = 1, 3, 10")
```



Training data and testing data

Example:

Use the familiar iris data as an example.

Classes $g = 3$, variables $p = 4$, total observations $n = 150$.

Set 70% as training data, 30% as testing data.

```
library(class)   # for k-nn function
data(iris)
myDat = iris        # Plot titles need to match
trainRate = 0.7     # training vs testing data
set.seed(2023)
trainNo=sample(dim(myDat)[1],trainRate*dim(myDat)[1])
```

```
myDat[1:2,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Spec
## 1          5.1         3.5          1.4         0.2  set
## 2          4.9         3.0          1.4         0.2  set
```

```
trainNo[1:10]
```

```
##  [1] 117 105  26  44  98  29  49 125 133  72
```

```
table(myDat[trainNo,]$Species)  # training class label
```

```
##
##     setosa versicolor  virginica
##         36         37         32
```

```
table(myDat[-trainNo,]$Species) # testing class label
```

```
##
##     setosa versicolor  virginica
##         14         13         18
```

Fit k-NN model for k=1

```
knn1 = knn(train=myDat[trainNo,1:4],
           test=myDat[-trainNo,1:4],
    cl=myDat$Species[trainNo],k=1)
table(myDat$Species[-trainNo],knn1)
```

```
##             knn1
##              setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0         13         0
##   virginica       0          1        17
```

Fit k-NN model for k=10

```
knn10 = knn(train=myDat[trainNo,1:4],
           test=myDat[-trainNo,1:4],
    cl=myDat$Species[trainNo],k=10)
table(myDat$Species[-trainNo],knn10)
```

```
##             knn10
##              setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0         12         1
##   virginica       0          1        17
```

Fit k-NN model for k=7

```
knn7 = knn(train=myDat[trainNo,1:4],
           test=myDat[-trainNo,1:4],
    cl=myDat$Species[trainNo],k=7,prob=TRUE)
table(myDat$Species[-trainNo],knn7)
```

```
##               knn7
##                setosa versicolor virginica
##    setosa          14          0         0
##    versicolor       0         12         1
##    virginica        0          1        17
```

```
attributes(knn7)
```

```
## $levels
## [1] "setosa"     "versicolor" "virginica"
##
## $class
## [1] "factor"
##
## $prob
##  [1] 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.
## [11] 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.
## [21] 0.8571 0.5714 1.0000 0.8889 1.0000 1.0000 1.0000 1.
## [31] 1.0000 1.0000 1.0000 1.0000 0.8571 0.7143 1.0000 1.
## [41] 1.0000 1.0000 1.0000 1.0000 1.0000
```

Which k? Use training and validation data

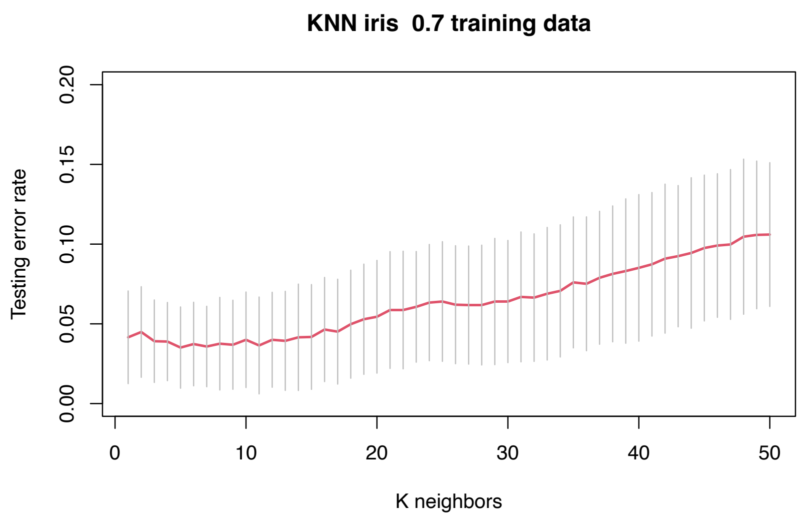
The data set is small enough to tryout various k's on k-NN model. For each k, randomly pick training/validation data, 100 times.

```
Rep = 100  # no. random training set
K = 50  # Fit KNN models for k=1:K
err=rep(0,K); errMat=matrix(0,Rep,K); # trainRate = 0.7
set.seed=329246
for (r in 1:Rep) {
  trainNo = sample(dim(myDat)[1],trainRate*dim(myDat)[1])
  for (i in 1:K){
    knnFit <- knn(train=myDat[trainNo,1:4],
                  test=myDat[-trainNo,1:4],
                  cl=myDat$Species[trainNo],k=i)
    err[i] = 1- mean(knnFit == myDat[-trainNo,]$Species)
    }
  errMat[r,]=err
}
```

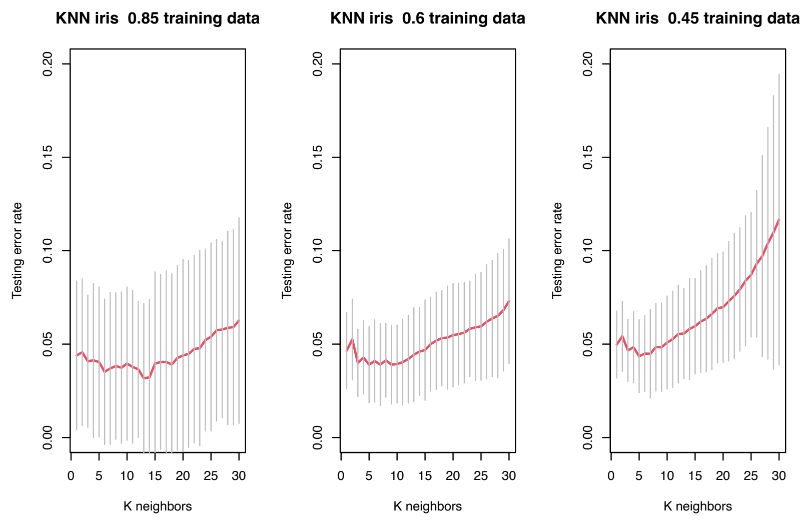
Plot classification error rates for each k (code)

```
plot(1:K,ylim=c(0,.2),type="n",
     ylab="Testing error rate",xlab="K neighbors",
    main=paste("KNN iris ", trainRate, "training data"))
points(1:K,colMeans(errMat),type="l",col=2,lwd=2)
segments(1:K, colMeans(errMat)-sqrt(diag(cov(errMat))),
    1:K, colMeans(errMat)+sqrt(diag(cov(errMat))),
    col='gray')
```

Plot classification error rates for each k (plot)



Classification testing errors vary by training data proportion

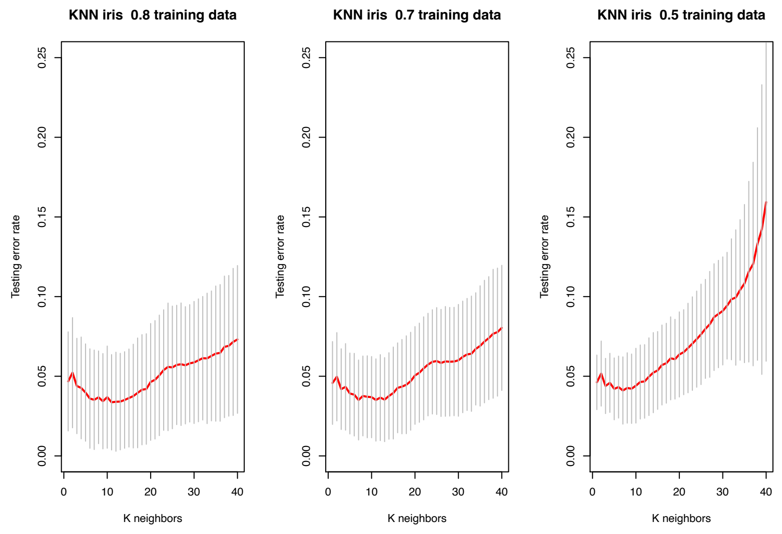


Comparisons

Discussions (based on the error rates plots)

From the error rate plots,

- What is a good choice of k, the number of nearest neighbors?
- How should we choose the size of training data vs testing data?
- Observations: variance trade-off.
- Scaling or normalizing variables are often necessary.


KNN iris 0.8 training data
KNN iris 0.7 training data
KNN iris 0.5 training data
Testing error rate
K neighbors