StatR 201: Winter 2013

Homework 05

Rod Doe

March 10, 2013

Data Preparation, Exploration, and Sanity Check

```
setwd("C:/Users/Rod/SkyDrive/R/201/Week08")  # SkyDrive from home
#setwd("C:/Users/rodney.doe/SkyDrive/R/201/Week08")  # SkyDrive from work
train = read.csv("seedTrain.csv", header=TRUE)

library(class)

# Sanity check data
str(train)
plot(train$Area)
plot(train$Perim)
plot(train$Compact)
plot(train$Length)
plot(train$Width)
plot(train$Asym)
plot(train$Groove)
plot(train$Variety)
cor(train)
pairs(train)
```

The data appear to be in order. No pathological issues here…

KNN Cross-Validation

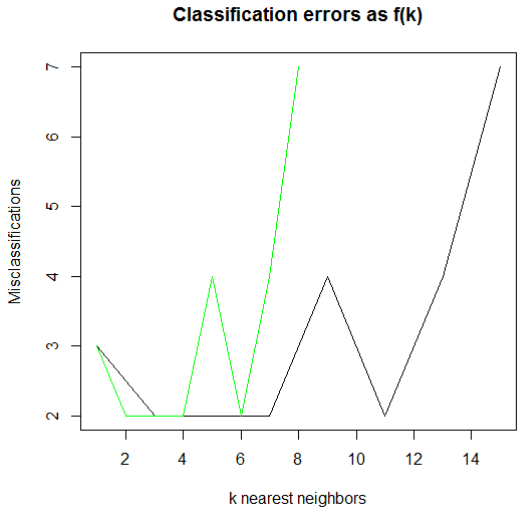
```
# knn.cv returns a vector of classifications of, in this case, Variety.
# Compare the difference between values of Variety estimated by
# knn classification to those specified in the data.
length(train$variety) - sum(knn.cv(train, cl=train$Variety, k=1) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=1) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=3) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=5) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=7) ==
train$Variety)
```

```
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=9) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=11) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=13) ==
train$Variety)
length(train$Variety) - sum(knn.cv(train, cl=train$Variety, k=15) ==
train$Variety)

# Store results of successive runs of knn.cv.
# Goal here is to compare the result of classification
# of Variety to actual Variety.

idx = seq(1, 15, 2)
errs = numeric(length(idx))
for (i in 1:length(idx)) {
  errs[i] = sum(knn.cv(train, cl=train$Variety, k=idx[i]) != train$Variety)
}

plot(idx, errs, type="l", main="Classification errors as f(k)", xlab="k
nearest neighbors", ylab="Misclassifications")
lines(errs, col="green")
```



We see the best performance for k = 3.

RPART Cross Validation

I spent about ten hours on Saturday trying to get this to work. This is the summary of my efforts:

```
=====================================================================
> setwd("C:/Users/Rod/SkyDrive/R/201/Week08")  # SkyDrive from home
> #setwd("C:/Users/rodney.doe/SkyDrive/R/201/Week08")  # SkyDrive from work
> source("rpartCV.r")
> data = read.csv("seedTrain.csv", header=TRUE)
>
> library(rpart)
>
> edges = c(min(data$Variety) - 1, sort(unique(data$Variety))) + 0.5
> strata = cut(data$Variety, edges)
> cvid = rep(NA, length(strata))
>
> size = 5
> for (a in levels(strata)) {
+    aid = which(strata == a)
+    cvid[aid] = sample(rep(1:5, ceiling(length(aid)/5)), size)
+ }
>
> cpvec = 0.1^seq(1, 4, 0.5)
>
> cout = rpartCV(data$Variety ~ ., data,
+ cvid=cvid, predtype='class', cpvec=cpvec,
+ parms=list(split="information"))
Sun Mar 10 11:49:48 2013
Error in model.frame.default(formula = formula, data = data[cvid != a,  :
  variable lengths differ (found for 'Area')
>
> #=====================================================================
```

I have exhausted all available profanity and even borrowed some from my neighbor in an effort to jump start this code. Sadly, even that failed. I'm going to take inspiration from Frank Sinatra and do it My Way.

RPART Cross Validation – My Way

Here is the cross-validation code:

```
#==My Way=============================================
setwd("C:/Users/Rod/SkyDrive/R/201/Week08")  # SkyDrive from home
data = read.csv("seedTrain.csv", header=TRUE)

library(rpart)

# Convert Variety to a factor.  Classification NEEDS this!
```

```
data$Variety = factor(data$Variety)  # MUY IMPORTANTE!!
levels(data$Variety)

idxVariety.1 = which(data[,idxVariety] == 1)
idxVariety.2 = which(data[,idxVariety] == 2)
idxVariety.3 = which(data[,idxVariety] == 3)

idxAll = 1:length(data[,1])
nFolds = 10
size = 5
# vcp = 0.1^seq(1, 4, 0.5)  # errs = 14 11 11 11 11 11 11
# vcp = 10^seq(-3.5, -0.5, 0.5)  # errs =  11 11 11 11 11 14 14
# vcp = 0.1^seq(1.0, 2.5, 0.25)  #errs = 14 14 11 11 11 11 11
# vcp = 10^seq(-3.0, -0.5, 0.25) # errs = 11 11 11 11 11 11 11 14 14 14 14
# vcp = 10^seq(-5.0, -0.5, 0.25)  # errs  =  11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 14 14 14 14
vcp = c(0.005, 0.01, 0.015)  # errs =  11 11 11

cvResult=matrix(NA,nrow=length(idxAll), ncol=length(vcp))

for (idxCp in 1:length(vcp)) {
      #cat('c')
      for (nFold in 1:nFolds)
      {
            # Determine where to start to read fold data.
            idxStart = ((nFold - 1) * size) + 1

            # Select a small, uniform set of validation indices.
            idxValidation = c(idxVariety.1[idxStart:(idxStart + (size - 1))],
                              idxVariety.2[idxStart:(idxStart + (size - 1))],
                              idxVariety.3[idxStart:(idxStart + (size - 1))])

            # Select all remaining indices as training indices.
            idxTrain = setdiff(idxAll, idxTest)

            # Select validation data.
            dataValidation = data[idxValidation, ]

            # Select training data.
            dataTrain = data[idxTrain, ]

            # Generate a model using training data.
            model = rpart(dataTrain$Variety ~ .,
                  data=dataTrain,
                  control=rpart.control(cp=vcp[idxCp]),
                  parms=list(split="information"))

            # Predict values using validation data.
```

```
            cvResult[idxValidation, idxCp] = predict(model,
newdata=dataValidation, type="class")
            #cat('f')
      }
}

# Compare cvResults to observed results.
errs = numeric(length(vcp))
for (i in 1:length(vcp)) {
  errs[i] = sum(data$Variety != cvResult[,i])
}
(errs)
```

Here are the results for various values of cp:

```
# vcp = 0.1^seq(1, 4, 0.5)  # errs = 14 11 11 11 11 11 11
# vcp = 10^seq(-3.5, -0.5, 0.5)  # errs =  11 11 11 11 11 14 14
# vcp = 0.1^seq(1.0, 2.5, 0.25)  #errs = 14 14 11 11 11 11 11
# vcp = 10^seq(-3.0, -0.5, 0.25) # errs = 11 11 11 11 11 11 11 14 14 14 14
# vcp = 10^seq(-5.0, -0.5, 0.25)  # errs  =  11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 14 14 14 14
vcp = c(0.005, 0.01, 0.015)  # errs =  11 11 11
```

It seems this model produces either 11 or 14 errors. As such, cp = 0.01 seems as good as any other.

Classification on Test Data

In the previous training, we found that:

- KNN worked best with k=3.
- rpart worked best with cp= 0.01

Let's run the test data.

```
#==Classification================================================
setwd("C:/Users/Rod/SkyDrive/R/201/Week08")  # SkyDrive from home
library(class)
library(rpart)

train = read.csv("seedTrain.csv", header=TRUE)
test = read.csv("seedTest.csv", header=TRUE)

train$Variety = factor(train$Variety)
test$Variety = factor(test$Variety)

# Run KNN using K=3, which was optimum in earlier step.
sum(knn.cv(train, cl=train$Variety, k=3) != train$Variety)
```

```
# This produced 2 errors out of 60.  Not that bad.
#> test = read.csv("seedTest.csv", header=TRUE)
#> sum(knn.cv(train, cl=train$Variety, k=3) != train$Variety)
#[1] 2

# Create model from all training data using cp found to be optimum.
vcp = c(0.01)

model = rpart(train$Variety ~ .,
                  data=train,
                  control=rpart.control(vcp[1]),
                  parms=list(split="information"))

idxAll = 1:length(test[,1])
result=matrix(NA,nrow=length(idxAll), ncol=length(vcp))

# Run model on test data
result = predict(model, newdata=test, type="class")
sum(result != test$Variety)

# This produced 5 prediction errors.
# Congratulations Evelyn Fix!  You posthumously kicked butt!
> sum(result != test$Variety)
> [1] 5
```

In summary, Evelyn Fix' simple method classified better that rpart. I would guess that the authors of rpart aren't good cooks.

And now, I'm going to salvage some weekend.