BUS 41204 Review Session 2

Examples using k-NN and Trees

Siying Cao
siyingc@uchicago.edu

01/14/2017

Plan

- Example using beer review data to
  - solve parts of Question 2 on hw1
  - run regression trees with `rpart` and `tree` package
- k-NN and regression tree when x is multivariate
- Classification using k-NN

Solve question 2 with beer review data

Load packages

```
library(data.table)
library(rpart)
library(rpart.plot)
library(tree)
library(kknn)
```

Read in data

```
setwd("C:/Users/siying/Box Sync/TA/TA BUS41204/my stuff")
dt <- fread("beer_review.csv", header=TRUE,
            stringsAsFactors=TRUE)
names(dt)[2:dim(dt)[2]]
```

```
##  [1] "beer/ABV"          "beer/beerId"       "beer/brewerId"
##  [4] "beer/name"         "beer/style"        "review/appearan
##  [7] "review/aroma"      "review/overall"    "review/palate"
## [10] "review/taste"      "review/timeUnix"   "user/ageInSecon
## [13] "user/birthdayRaw"  "user/birthdayUnix" "user/gender"
## [16] "user/profileName"
```

Prepare dataset with intended variables

- $y = score$
- Dataset with two input variables: *ABV* and *Age*.

```
dt2 <- dt[,.(`review/overall`,`beer/ABV`,`user/ageInSeconds`)]
# remove rows with missing value
dt2 <- dt2[complete.cases(dt2*0),]
n <- dim(dt2)[1]
colnames(dt2) <- c("score", "abv", "age")
setkey(dt2, abv, age)
```

- Dataset with only one input: *ABV*

```
dt1 <- dt2[,.(score,abv)]
```

Summarize the data

```r
summary(dt2)
```

```
##      score            abv             age
##  Min.   :1.000   Min.   : 0.500   Min.   :7.034e+08
##  1st Qu.:3.500   1st Qu.: 5.500   1st Qu.:9.783e+08
##  Median :4.000   Median : 7.000   Median :1.100e+09
##  Mean   :3.911   Mean   : 7.451   Mean   :1.175e+09
##  3rd Qu.:4.500   3rd Qu.: 9.400   3rd Qu.:1.276e+09
##  Max.   :5.000   Max.   :39.440   Max.   :3.627e+09
```

Split the data

...75% training and 25% test

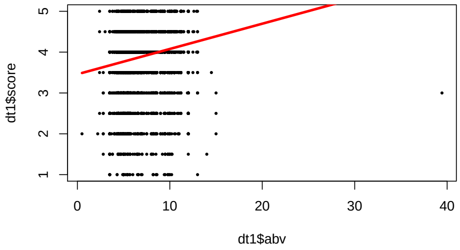
```
set.seed(123)

n_train <- 7000 # approximately 75% of data
tr_idx <- sample(1:n, n_train)
train <- dt1[tr_idx,]
test <- dt1[-tr_idx,]
```

Fit linear regression model

```
linear <- lm(score~abv, data=train)
```

Create scatter plot and best linear fit

```
plot(dt1$abv, dt1$score, cex=.5, pch=16)
yhat=predict(linear, dt1)
lines(dt1$abv, yhat, col="red", lwd=3)
```



k-NN using cross-validation

Download cv utilities

```
download.file("https://raw.githubusercontent.com/ChicagoBoothML/
source("docv.R") #this has docvknn used below
```

Do 5-fold cv twice, 10-fold cv onces

```
set.seed(123)
k_vec <- 2:100 # vector of k to loop over
cv1 <- docvknn(matrix(dt1$abv,ncol=1), dt1$score, k_vec,
               nfold=5)
```

```
## in docv: nset,n,nfold:  99 10479 5
## on fold:  1 , range:  1 : 2096
## on fold:  2 , range:  2097 : 4192
## on fold:  3 , range:  4193 : 6288
## on fold:  4 , range:  6289 : 8384
## on fold:  5 , range:  8385 : 10479
```

```r
cv2 <- docvknn(matrix(dt1$abv,ncol=1), dt1$score, k_vec,
               nfold=5)
```

```
## in docv: nset,n,nfold:  99 10479 5
## on fold:  1 , range:  1 : 2096
## on fold:  2 , range:  2097 : 4192
## on fold:  3 , range:  4193 : 6288
## on fold:  4 , range:  6289 : 8384
## on fold:  5 , range:  8385 : 10479
```

```r
cv3 <- docvknn(matrix(dt1$abv,ncol=1), dt1$score, k_vec,
               nfold=10)
```

```
## in docv: nset,n,nfold:  99 10479 10
## on fold:  1 , range:  1 : 1048
## on fold:  2 , range:  1049 : 2096
## on fold:  3 , range:  2097 : 3144
## on fold:  4 , range:  3145 : 4192
## on fold:  5 , range:  4193 : 5240
## on fold:  6 , range:  5241 : 6288
## on fold:  7 , range:  6289 : 7336
## on fold:  8 , range:  7337 : 8384
## on fold:  9 , range:  8385 : 9432
## on fold:  10 , range:  9433 : 10479
```

k-NN using cross-validation

Compute MSE for each cv (at the vector of k's)

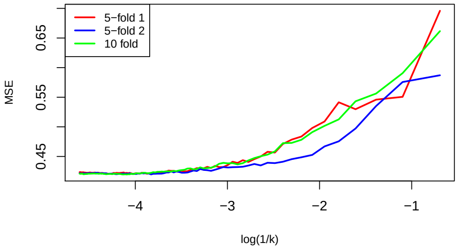
```
cv1=cv1/n
cv2=cv2/n
cv3=cv3/n
```

(Optional) Run cv multiple times using for loop

```
set.seed(123)
cvmean=rep(0,length(k_vec))
ndocv=50
cvmat=matrix(0, length(k_vec), ndoc)  # matrix to store results
for (i in 1:ndocv){
    cvtemp=docvknn(matrix(dt1$abv,ncol=1), dt1$score,
                   k_vec, nfold=10)
    cvmean=cvmean+cvtemp
    cvmat[,i]=cvtemp/n
}
cvmean=cvmean/ndocv
cvmean=cvmean/n
plot(k_vec, cvmean, type="n", ylim=range(cvmat), xlab="k")

# plot each cv
for (i in 1:ndocv) lines(k_vec, cvmat[,i], col=i, lty=3)
# plot average
lines(k_vec, cvmean, type="b", col="black", lwd=3)
```

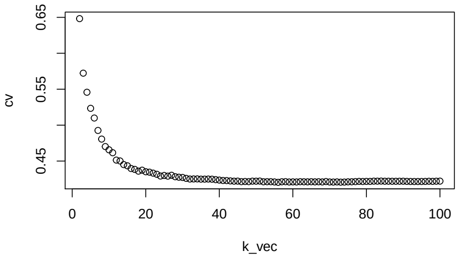
Plot three cv curves on one figure

```
rgy = range(c(cv1,cv2,cv3))
plot(log(1/k_vec),cv1,type="l",col="red",ylim=rgy,lwd=2,
     cex.lab=0.8, xlab="log(1/k)", ylab="MSE")
lines(log(1/k_vec),cv2,col="blue",lwd=2)
lines(log(1/k_vec),cv3,col="green",lwd=2)
legend("topleft",legend=c("5-fold 1","5-fold 2","10 fold"),
       col=c("red","blue","green"),lwd=2,cex=0.8)
```



Plot cross-validated MSE(k)

```
cv = (cv1+cv2+cv3)/3 #use average
plot(k_vec, cv)
```



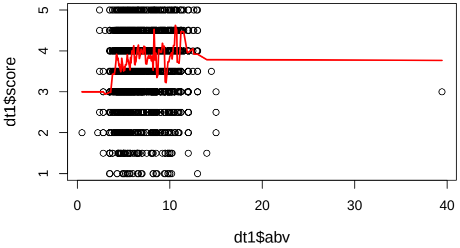
Get the best k

```
k_best = k_vec[which.min(cv)]
cat("the best k is: ",k_best,"\n")
```

```
## the best k is:  56
```

Fit with the best *k* and plot

```
near_best <- kknn(score~abv, train=dt1, test=dt1[,.(abv)],
                  k=k_best, kernel='rectangular')
plot(dt1$abv, dt1$score, cex.lab=1.2)
lines(dt1$abv, near_best$fitted, col="red", lwd=2, cex.lab=2)
```



Regression tree using cross-validation

```
big.tree=rpart(score~abv, data=dt1,
               control=rpart.control(minsplit=5,
                                     cp=0.0001,
                                     xval=10))
nbig = length(unique(big.tree$where))
cat('size of big tree: ',nbig,'\n')
```

```
## size of big tree:  60
```

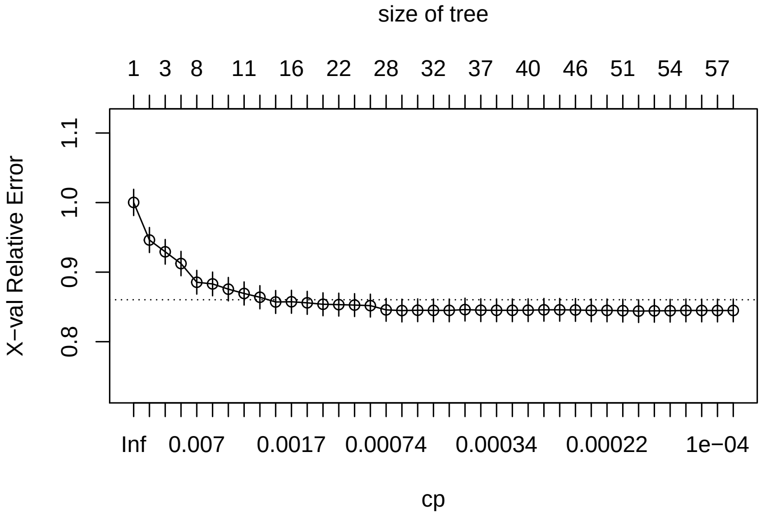
Find the best size of tree cp

```
cptable = printcp(big.tree)
```

```
##
## Regression tree:
## rpart(formula = score ~ abv, data = dt1, control = rpart.cont
##     cp = 1e-04, xval = 10))
##
## Variables actually used in tree construction:
## [1] abv
##
```

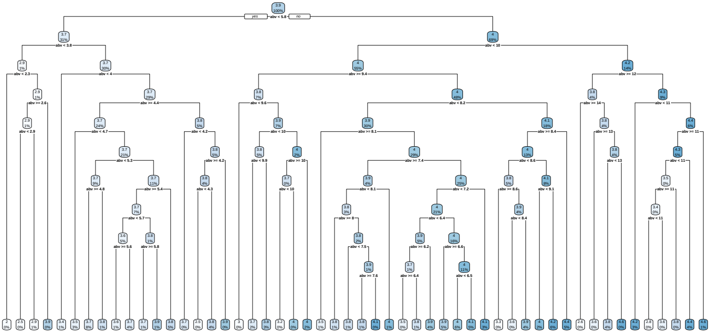
Optimal cp parameter

```
bestcp = cptable[ which.min(big.tree$cptable[,"xerror"]), "CP" ]
plotcp(big.tree) # plot results
```



Prune down the optimal tree

```
par(mfrow=c(1,1))
best.tree=prune(big.tree, cp=bestcp)
rpart.plot(best.tree)
```



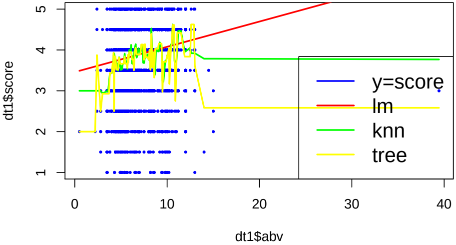
Put everything together

```
dt1[,`:=`(lm_fit=yhat, knn_fit=near_best$fitted,
          tree_fit=predict(best.tree))]
```

```
##          score   abv   lm_fit  knn_fit tree_fit
##      1:    2.0  0.50 3.489640 3.000000 2.000000
##      2:    2.0  2.20 3.594817 3.000000 2.000000
##      3:    2.5  2.40 3.607191 3.000000 3.875000
##      4:    5.0  2.40 3.607191 3.000000 3.875000
##      5:    3.5  2.40 3.607191 3.000000 3.875000
##    ---
## 10475:    3.5 14.50 4.355802 3.785714 2.583333
## 10476:    2.0 15.00 4.386737 3.785714 2.583333
## 10477:    3.0 15.00 4.386737 3.785714 2.583333
## 10478:    2.5 15.00 4.386737 3.785714 2.583333
## 10479:    3.0 39.44 5.898808 3.767857 2.583333
```

Plot the fit

```
plot(dt1$abv, dt1$score, pch=16, col='blue', cex=0.5)
lines(dt1$abv,dt1$lm_fit, col="red", lwd=2)
lines(dt1$abv, dt1$knn_fit, col="green", lwd=2)
lines(dt1$abv, dt1$tree_fit, col="yellow", lwd=2)
legend("bottomright",legend=c("y=score","lm","knn","tree"),
       col=c("blue","red","green","yellow"),lwd=2,cex=1.5)
```



Pick the best performing model

- using test MSE

```
MSE <- dt1[-tr_idx,
           lapply(.SD,function(x) (x-score)^2),
           .SDcols=paste0(c("lm", "knn", "tree"),"_fit")]
MSE[,lapply(.SD, mean),
    .SDcols=paste0(c("lm", "knn", "tree"),"_fit")]
```

```
##        lm_fit   knn_fit  tree_fit
## 1: 0.4830103 0.4243779 0.4158278
```

- Tree does the best!

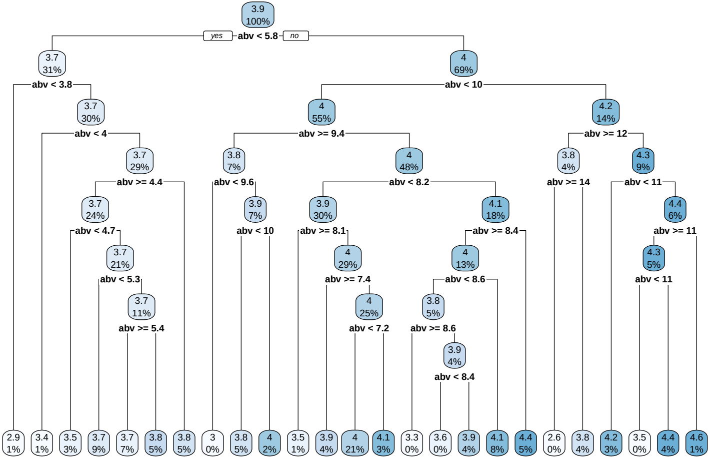
Run regression trees with *rpart* package

Start with the big tree

```
temp = rpart(score~abv, data=dt1,
              control=rpart.control(minsplit=5,
                                    cp=0.001,
                                    xval=0)
              )
```

Run regression trees with *rpart* package

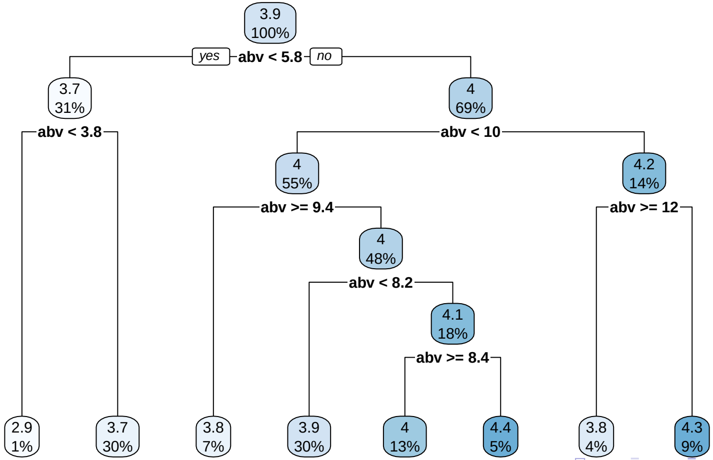
```
rpart.plot(temp)
```



Run regression trees with *rpart* package

Prune the tree to have 8 leafs

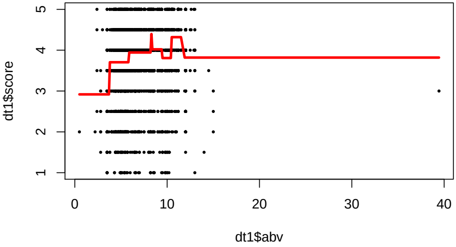
```
beer.tree = prune(temp, cp=0.007)
rpart.plot(beer.tree)
```



Run regression trees with *rpart* package

Plot the data and fit

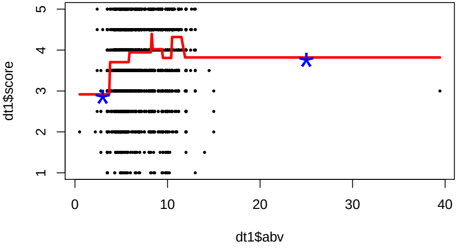
```
beer.fit = predict(beer.tree)
plot(dt1$abv, dt1$score, cex=.5, pch=16)
lines(dt1$abv,beer.fit,col="red",lwd=3) #step function fit
```



Run regression trees with *rpart* package

Make prediction on two points

```
preddf = data.frame(abv=c(3,25))
yhat = predict(beer.tree,preddf)
plot(dt1$abv, dt1$score, cex=.5, pch=16) #plot data
lines(dt1$abv,beer.fit,col="red",lwd=3)
points(preddf$abv,yhat,col="blue",pch="*",cex=3)
```



Run regression trees with *tree* package

```
temp2 <- tree(score~abv, data=dt1, mindev=0.0001)
cat("first big tree size: \n")
```

```
## first big tree size:
```

```
print(length(unique(temp2$where)))
```

```
## [1] 50
```

```
beer.tree2=prune.tree(temp2,best=8)
cat("pruned tree size: \n")
```

```
## pruned tree size:
```

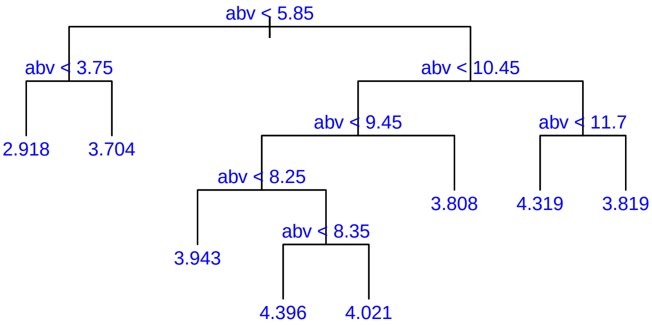
```
print(length(unique(beer.tree2$where)))
```

```
## [1] 8
```

Run regression trees with *tree* package

Plot the tree

```
plot(beer.tree2,type="uniform")
text(beer.tree2,col="blue",label=c("yval"),cex=.8)
```



k-NN and regression tree when x is multivariate

- Regression tree

```
temp2=rpart(score~., data=dt2,
           control=rpart.control(minsplit=5,
                                cp=0.001,
                                    xval=0))
rpart.plot(temp2)

# prune
beer.tree2=prune(temp2, cp=0.008)
rpart.plot(beer.tree2)
```

k-NN and regression tree when x is multivariate

- Regression tree

```
# create a perspective plot
pv=seq(from=0.01,to=0.99,by=0.1)
x1q=quantile(dt2$abv, probs=pv)
x2q=quantile(dt2$age, probs=pv)
grids=expand.grid(x1q, x2q)
dt_pred <- data.table(age=grids[2], abv=grids[1])
colnames(dt_pred) <- c("age","abv")
beer.fit.plt=predict(beer.tree2,dt_pred)

persp(x1q, x2q, matrix(beer.fit.plt, ncol=length(x2q), byrow=T),
      theta=150, xlab='age', ylab='abv', zlab='score',
      zlim=c(min(dt2$score), 1.1*max(dt2$score)))
```

k-NN and regression tree when x is multivariate

- k-NN

```
# scale input variables
scf = function(x) {return((x-min(x))/(max(x)-min(x)))}
dt2[,`:=`(abv=scf(abv), age=scf(age))]
```

```
##            score         abv        age
##        1:    2.0 0.00000000 0.08702472
##        2:    2.0 0.04365691 0.20658398
##        3:    2.5 0.04879301 0.06580783
##        4:    5.0 0.04879301 0.12898571
##        5:    3.5 0.04879301 0.20658398
##      ---
## 10475:    3.5 0.35952748 0.03217994
## 10476:    2.0 0.37236775 0.11320478
## 10477:    3.0 0.37236775 0.13974191
## 10478:    2.5 0.37236775 0.18066741
## 10479:    3.0 1.00000000 0.16997031
```

k-NN and regression tree when x is multivariate

- k-NN

```
beer.knn <- kknn(score~., train=dt2, test=dt2[,.(abv, age)],
                 k=5, kernel='rectangular')
dt1[,knn_fit:=beer.knn$fitted.values]
```

```
##          score   abv   lm_fit knn_fit tree_fit
##      1:    2.0  0.50 3.489640     2.8 2.000000
##      2:    2.0  2.20 3.594817     2.4 2.000000
##      3:    2.5  2.40 3.607191     2.2 3.875000
##      4:    5.0  2.40 3.607191     3.0 3.875000
##      5:    3.5  2.40 3.607191     2.4 3.875000
##    ---
## 10475:    3.5 14.50 4.355802     3.4 2.583333
## 10476:    2.0 15.00 4.386737     2.5 2.583333
## 10477:    3.0 15.00 4.386737     2.4 2.583333
## 10478:    2.5 15.00 4.386737     2.8 2.583333
## 10479:    3.0 39.44 5.898808     2.8 2.583333
```

Classification using k-NN

- Use `fgl` dataset from R library

Load the libraries and dataset

```
library("MASS")
library("kknn")
data(fgl)
```

View dataset codebook

```
help(fgl)
```

Quick look at the dataset

```
head(fgl,n=3)
```

```
##      RI    Na   Mg   Al    Si    K   Ca Ba Fe type
## 1  3.01 13.64 4.49 1.10 71.78 0.06 8.75  0  0 WinF
## 2 -0.39 13.89 3.60 1.36 72.73 0.48 7.83  0  0 WinF
## 3 -1.82 13.53 3.55 1.54 72.99 0.39 7.78  0  0 WinF
```

Classification using k-NN

Code the 7 types into 3 main categories

```
n=nrow(fgl)
y = rep(3,n)
y[fgl$type=="WinF"]=1
y[fgl$type=="WinNF"]=2
y = as.factor(y)
levels(y) = c("WinF","WinNF","Other")
print(table(y,fgl$type))
```

```
##
## y       WinF WinNF Veh Con Tabl Head
##   WinF    70     0   0   0    0    0
##   WinNF    0    76   0   0    0    0
##   Other    0     0  17  13    9   29
```

```
x = cbind(fgl$RI,fgl$Na,fgl$Al)
```

Classification using k-NN

Scale the input variables, construct ready-to-use dataset

```
scf = function(x) {return((x-min(x))/(max(x)-min(x)))}
x = apply(x,2,scf)     # scale all x

colnames(x) = c("RI","Na","Al")
ddf = data.frame(type=y,x)
names(ddf) =c("type","RI","Na","Al")
```

Check the dataset

```
tail(ddf, n=3)
```

```
##      type        RI        Na        Al
## 212 Other 0.4170325 0.5458647 0.5389408
## 213 Other 0.2352941 0.5488722 0.5140187
## 214 Other 0.2616330 0.5263158 0.5576324
```

Classification using k-NN

- Run k-NN

```
near = kknn(type~.,ddf,ddf,k=10,kernel = "rectangular")
print(table(near$fitted,ddf$type))
```

```
##
##          WinF WinNF Other
##   WinF     58    13    14
##   WinNF    11    57    12
##   Other     1     6    42
```

Predicted probabilities

```
fitdf = data.frame(type=ddf$type,near$prob)
names(fitdf)[2:4] = c("ProbWinF","ProbWinNF","ProbOther")
head(fitdf,n=3)
```

```
##   type ProbWinF ProbWinNF ProbOther
## 1 WinF      0.6       0.3       0.1
## 2 WinF      0.4       0.4       0.2
## 3 WinF      0.1       0.9       0.0
```

Classification using k-NN

```
par(mfrow=c(1,3))
plot(ProbWinF~type,fitdf,col=c(grey(.5),2:3),cex.lab=1.4)
plot(ProbWinNF~type,fitdf,col=c(grey(.5),2:3),cex.lab=1.4)
plot(ProbOther~type,fitdf,col=c(grey(.5),2:3),cex.lab=1.4)
```