kNN Clustering - Regression

Karen Mazidi

This example uses the Auto data set in package ISLR. First we try linear regression as a baseline and then see if knn can beat the linear model.

```r
library(ISLR)
attach(Auto)
Auto$origin <- as.factor(origin)
```

Build a linear model with all predictors.

```r
lm1 <- lm(mpg ~.-name, data=Auto)
summary(lm1)
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders    -4.897e-01  3.212e-01  -1.524 0.128215
## displacement  2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower   -1.818e-02  1.371e-02  -1.326 0.185488
## weight       -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
## acceleration  7.910e-02  9.822e-02   0.805 0.421101
## year          7.770e-01  5.178e-02  15.005  < 2e-16 ***
## origin2       2.630e+00  5.664e-01   4.643 4.72e-06 ***
## origin3       2.853e+00  5.527e-01   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.307 on 383 degrees of freedom
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

Train and test on a linear model

The results from lm1 indicated that weight, year, and origin appear to be significant predictors so let's build a model just from those.

First, randomly sample 80% from the data set, and let those indices be for training while the others are for the test set.

```r
set.seed(1958)  # for reproducible results
i <- sample(1:nrow(Auto), round(nrow(Auto)*0.8), replace=FALSE)
train <- Auto[i,]
test <- Auto[-i,]
lm2 <- lm(mpg~weight+year+origin, data=train)
pred <- predict(lm2, newdata=test)
cor(pred, test$mpg)
```

```
## [1] 0.9233468
```

```r
mse <- mean((pred - test$mpg)^2)
```

The correlation for the linear model was 91% which is good. Can kNN do better?

kNN for regression

We will use the same train and test set as we used for the linear model. We will train on weight, year and origin as for the linear model.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'Auto':
##
##     mpg
```

```r
library(DMwR)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
train$origin <- as.integer(train$origin)
test$origin <- as.integer(test$origin)
fit <- knnreg(train[,2:8],train[,1],k=3)
predictions <- predict(fit, test[,2:8])
cor(predictions, test$mpg)
```

```
## [1] 0.8699488
```

```r
mse <- mean((predictions - test$mpg)^2)
```

So the results were not as good for knn, the correlation was 81%.

As discussed in class, we know that clustering algorithms work best if the data is scaled, so let's scale the data and try again.

```r
Auto$origin <- as.integer(Auto$origin)
scaled_data <- scale(Auto[,c(1, 5,7,8)])
df <- data.frame(scale(Auto[,c(1, 5,7,8)])) # just mpg, weight, year, origin
train <- df[i,]
test <- df[-i,]
fit <- knnreg(train[,2:4],train[,1],k=3)
predictions <- predict(fit, test[,2:4])
cor <- cor(predictions, test$mpg)
mse <- mean((predictions - test$mpg)^2)
```

Wow, scaling improved the correlation to 92% which is a little better than the linear model.

Finding the best k

Try various values of k and plot the results.

```
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for (k in seq(1, 39, 2)){
  fit_k <- knnreg(train[,2:4],train[,1], k=k)
  pred_k <- predict(fit_k, test[,2:4])
  cor_k[i] <- cor(pred_k, test$mpg)
  mse_k[i] <- mean((pred_k - test$mpg)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}
```

```
## [1] "k= 1 0.898707858706253 0.20539753042819"
## [1] "k= 3 0.927107392998045 0.147055554639021"
## [1] "k= 5 0.936131993895116 0.122079270311939"
## [1] "k= 7 0.928225559570256 0.128061833150553"
## [1] "k= 9 0.930217856770169 0.123459880287717"
## [1] "k= 11 0.921957076237405 0.137230801091831"
## [1] "k= 13 0.924375397759162 0.131448318743602"
## [1] "k= 15 0.918907279870604 0.139858109252749"
## [1] "k= 17 0.917329161322684 0.140554805303069"
## [1] "k= 19 0.911897888670477 0.149280633486754"
## [1] "k= 21 0.907687109043115 0.1554077394442"
## [1] "k= 23 0.903566379070446 0.162107812995699"
## [1] "k= 25 0.905747799092626 0.158027814469075"
## [1] "k= 27 0.906107490345713 0.156906133001286"
## [1] "k= 29 0.904582549235259 0.159241624553884"
## [1] "k= 31 0.907545469776619 0.154346733520627"
## [1] "k= 33 0.906755319965085 0.155365229807224"
## [1] "k= 35 0.904330811615834 0.159127494979743"
## [1] "k= 37 0.905625619798105 0.157221680222979"
## [1] "k= 39 0.905144166235114 0.158072881643417"
```

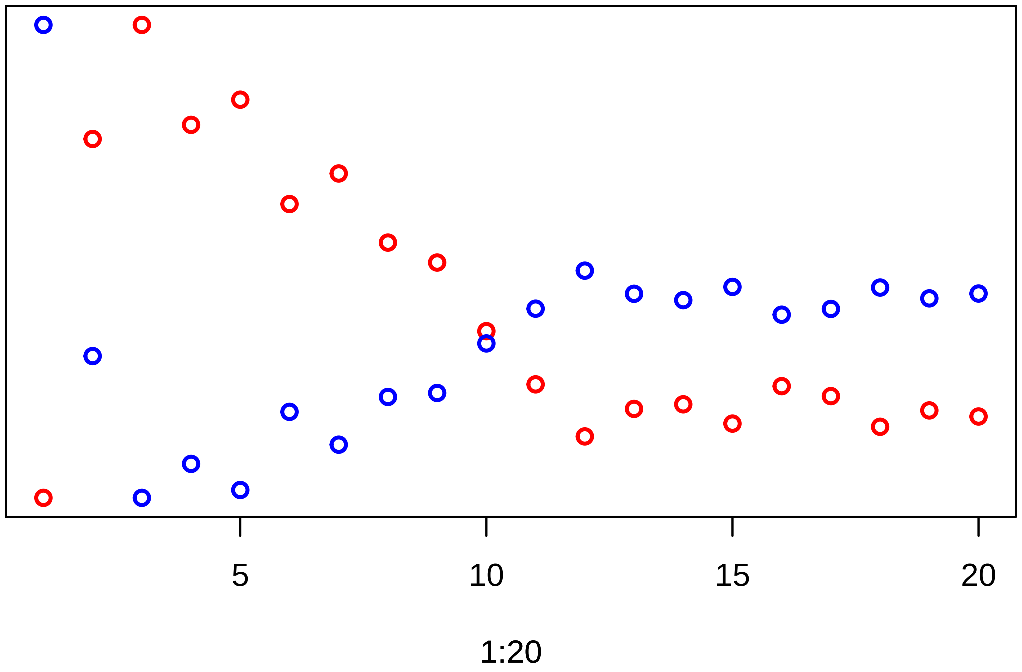
```
plot(1:20, cor_k, lwd=2, col='red', ylab="", yaxt='n')
par(new=TRUE)
plot(1:20, mse_k, lwd=2, col='blue', labels=FALSE, ylab="", yaxt='n')
```

```
## Warning in plot.window(...): "labels" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "labels" is not a graphical parameter

## Warning in box(...): "labels" is not a graphical parameter

## Warning in title(...): "labels" is not a graphical parameter
```



### k=15

The results above indicate that k=15 is best. We run the model above. It is slightly better than k=3.

```
fit_15 <- knnreg(train[,2:4],train[,1],k=3)
predictions_15 <- predict(fit_15, test[,2:4])
cor_15 <- cor(predictions_15, test$mpg)
mse_15 <- mean((predictions_15 - test$mpg)^2)
```