kNN Regression

This notebook performs kNN regression on the Auto data in package ISLR. The ISLR book is a really good and free resource for statistical learning in R. Two of the authors, Hastie and Tibshirani, also have a series of videos available at this link

Linear Regression baseline

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.

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

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

Train test split

Also remove the name column.

```
set.seed(1234)
i <- sample(1:nrow(Auto), round(nrow(Auto)*0.8), replace=FALSE)
train <- Auto[i, -9]
test <- Auto[-i, -9]
```

Build a linear regression model with all predictors.

```
lm1 <- lm(mpg ~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1240 -1.9819 -0.0147  1.7986 13.5953
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  -1.828e+01  5.078e+00  -3.600 0.000371 ***
## cylinders    -6.165e-01  3.460e-01  -1.782 0.075789 .
## displacement  2.131e-02  8.123e-03   2.623 0.009147 **
## horsepower   -2.849e-02  1.430e-02  -1.992 0.047229 *
## weight       -5.768e-03  7.074e-04  -8.154 9.28e-15 ***
## acceleration -5.539e-02  1.056e-01  -0.524 0.600428
## year          8.010e-01  5.649e-02  14.180  < 2e-16 ***
## origin2       2.166e+00  6.102e-01   3.549 0.000448 ***
## origin3       2.852e+00  5.967e-01   4.779 2.74e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.195 on 305 degrees of freedom
## Multiple R-squared:  0.8309, Adjusted R-squared:  0.8265
## F-statistic: 187.4 on 8 and 305 DF,  p-value: < 2.2e-16
```

Evaluate

```
pred1 <- predict(lm1, newdata=test)
cor_lm1 <- cor(pred1, test$mpg)
mse_lm1 <- mean((pred1 - test$mpg)^2)
print(paste("cor=", cor_lm1))
```

```
## [1] "cor= 0.893368513146442"
```

```
print(paste("mse=", mse_lm1))
```

```
## [1] "mse= 14.5943814943833"
```

These results aren't bad. Let's see what happens with kNN

kNN for regression

Notice that origin needs to be an integer.

```
library(caret)
```

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

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

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

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

```
train$origin <- as.integer(train$origin)
test$origin <- as.integer(test$origin)

# fit the model
fit <- knnreg(train[,2:8],train[,1],k=3)

# evaluate
pred2 <- predict(fit, test[,2:8])
cor_knn1 <- cor(pred2, test$mpg)
mse_knn1 <- mean((pred2 - test$mpg)^2)
print(paste("cor=", cor_knn1))
```

```
## [1] "cor= 0.878344750927035"
```

```
print(paste("mse=", mse_knn1))
```

```
## [1] "mse= 17.029943019943"
```

The results for kNN weren't quite as good as the linear regression model.

One reason might be that we didn't scale the data. kNN will work better on scaled data.

Scale the data

Notice we are scaling both train and test on the means and standard deviations of the training set. This is considered a best practice so that information about the test data doesn't leak into the scaling.

```
train_scaled <- train[, 2:8]  # omit name and don't scale mpg
means <- sapply(train_scaled, mean)
stdvs <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center=means, scale=stdvs)
test_scaled <- scale(test[, 2:8], center=means, scale=stdvs)
```

kNN on scaled data

```
fit <- knnreg(train_scaled, train$mpg, k=3)
pred3 <- predict(fit, test_scaled)
cor_knn2 <- cor(pred3, test$mpg)
mse_knn2 <- mean((pred3 - test$mpg)^2)
print(paste("cor=", cor_knn2))
```

```
## [1] "cor= 0.905223016978903"
```

```
print(paste("mse=", mse_knn2))
```

```
## [1] "mse= 14.4425925925926"
```

Wow. Now kNN has a higher correlation than linear regression and a lower mse.

Find the best k

Try various values of k and plot the results.

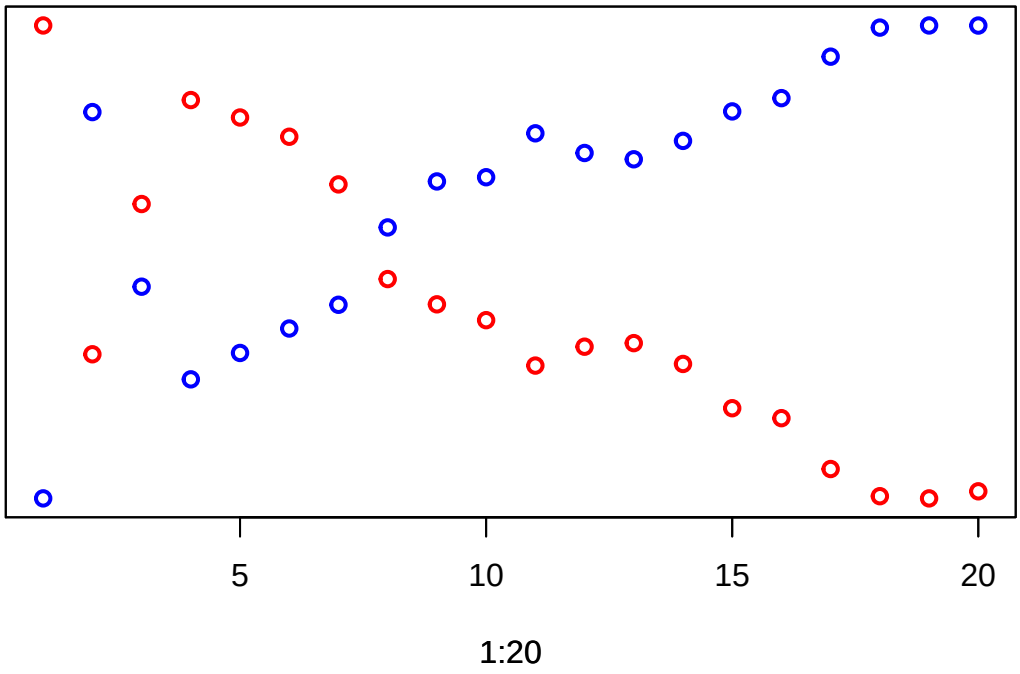
```
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for (k in seq(1, 39, 2)){
  fit_k <- knnreg(train_scaled,train$mpg, k=k)
  pred_k <- predict(fit_k, test_scaled)
  cor_k[i] <- cor(pred_k, test$mpg)
  mse_k[i] <- mean((pred_k - test$mpg)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}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

```
## [1] "k= 1 0.928963796663712 10.4517948717949"
## [1] "k= 3 0.905223016978903 14.4425925925926"
## [1] "k= 5 0.916071827066313 12.6385384615385"
## [1] "k= 7 0.923579467849894 11.6815096807954"
## [1] "k= 9 0.922319734386922 11.9541389680279"
## [1] "k= 11 0.92092994977216 12.2067524899343"
```

```
## [1] "k= 13 0.917489418697034 12.4518047337278"
## [1] "k= 15 0.91065328097819 13.2510102564103"
## [1] "k= 17 0.908829459455411 13.7260651228817"
## [1] "k= 19 0.90768560899354 13.7695333475389"
## [1] "k= 21 0.904407122181182 14.2233957528763"
## [1] "k= 23 0.905769718187576 14.0201846735495"
## [1] "k= 25 0.906027349563838 13.9550775384615"
## [1] "k= 27 0.904524090335153 14.14520347508"
## [1] "k= 29 0.901328845151209 14.4494716317571"
## [1] "k= 31 0.90060956863369 14.5867511139572"
## [1] "k= 33 0.896938247996035 15.0150978087511"
## [1] "k= 35 0.894975428658597 15.3155299843014"
## [1] "k= 37 0.894818579095498 15.3374458242026"
## [1] "k= 39 0.895327551594399 15.3366590805644"
```

```
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
```



We can visually see that at k=1 the red correlation is highest and the blue mse is lowest.

```
which.min(mse_k)
```

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

```
which.max(cor_k)
```

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

kNN with k=1

The results were better with k=1.

```
fit <- knnreg(train_scaled, train$mpg, k=1)
pred4 <- predict(fit, test_scaled)
cor_knn3 <- cor(pred4, test$mpg)
mse_knn3 <- mean((pred4 - test$mpg)^2)
print(paste("cor=", cor_knn3))
```

```
## [1] "cor= 0.928963796663712"
```

```
print(paste("mse=", mse_knn3))
```

```
## [1] "mse= 10.4517948717949"
```