STATS 790
Assignment #1

Dean Hansen - 400027416

20 January, 2023

Contents

Question #1

In addition to the reading from Breiman, I read Professor Gelman's response. One remark is Professor Gelman wrote his response 20 years after Breiman's article. In those 20 years, computing power has grown exponentially, and methods that were once infeasible, such as certain kinds of Bayesian analysis, have since become widespread due to the cheapness and availability of compute. It would be interesting if Breiman had written a follow up article discussing some of the new ways prediction machines like neural networks, whose utility has been demonstrated in tasks from protein structure prediction to speech recognition, have strengthened his original point that academic Statistics has been falling behind on many interesting problems. One further point, it seems now that prediction has slowly become as much of a focus as inference for those in academic Statistics, so maybe Breiman's initial point(s) are not as applicable nowadays.

Question #2

Here we replicate figure 2.5 from ESL (page 23).

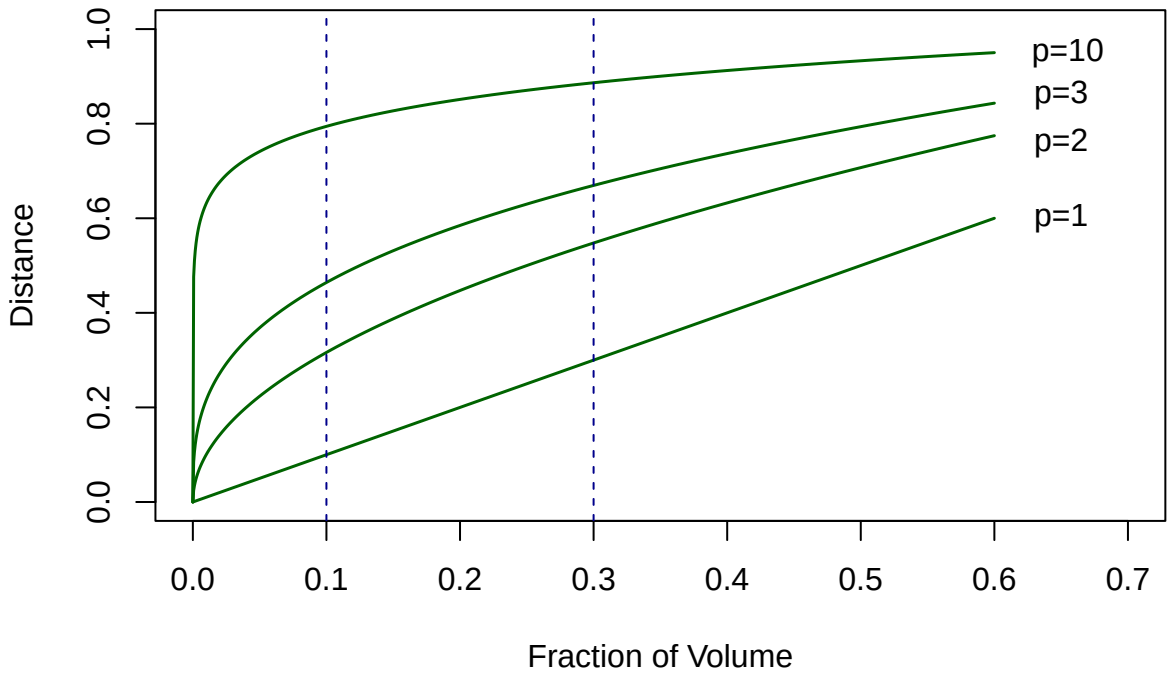
```
#functions from the book
fraction_volume <- function(p, r) {
  r^(1/p)
}

#dimensions and setting axis'
p <- c(1, 2, 3, 10)
r <- seq(0, 0.6, length.out = 1000)

x_1 <- fraction_volume(p=p[1], r=r)
x_2 <- fraction_volume(p=p[2], r=r)
x_3 <- fraction_volume(p=p[3], r=r)
x_10 <- fraction_volume(p=p[4], r=r)

#graph the distance
plot(r, seq(0, 0, length.out = 1000),
     xlim = c(0, 0.7),
     ylim = c(0,1),
     xlab = "Fraction of Volume",
     ylab = "Distance",
     col = "white",
     )
lines(r, x_1, col = "dark green", lwd = 1.5)
lines(r, x_2, col = "dark green", lwd = 1.5)
lines(r, x_3, col = "dark green", lwd = 1.5)
lines(r, x_10, col = "dark green", lwd = 1.5)
abline(v = 0.1, col="dark blue", lwd=1, lty=2)
abline(v = 0.3, col="dark blue", lwd=1, lty=2)
text(x = c(0.65, 0.65, 0.65, 0.655),
```

```
    y = c(0.6, 0.76, 0.86, 0.95),
    labels = c("p=1", "p=2", "p=3", "p=10")
    )
```



Question 3 (ADA 1.2)

We can break the expectation into the sum of two pieces, one where $Y > m$ and $Y <= m$. Then, assuming we can differentiate under the expectation, we can prove the result as follows.

$$\frac{d}{dm}MAE(m) = \frac{d}{dm}E[|Y - m|] \tag{1}$$

$$= \frac{d}{dm}[\int_{-\infty}^{m}(m - Y) * f(y)dy + \int_{m}^{\infty}(Y - m) * f(y)dy] \tag{2}$$

$$= \int_{-\infty}^{m} f(y)dy - 0 + 0 - \int_{m}^{\infty} f(y)dy] \tag{3}$$

Setting this to 0 and using the CDF $F(y)$ of $f(y)$ we have the following.

$$0 = F(m) - (1 - F(m)) \tag{4}$$

$$2 * F(m) = 1 \tag{5}$$

$$F(m) = 1/2 \tag{6}$$

This is exactly the definition of the median (middle value) of the distribution. Thus, the median minimizes the MAE.

In cases where the underlying data may be very skewed (ex. income data), it may make more sense to minimize the median than the mean, as the mean will often over-estimate the value. In the case of income, minimizing the mean will lead to higher predicted salaries, whereas the median would provide more sensible predictions.

Question 4 (ADA 1.7)

Using the global mean as the linear smoother, we are ignoring the distance between a predictor $\mathbf{x_i}$ and $\mathbf{x}$ and instead weighing them by the distance from the global mean value. In this case, we have the n by n influence matrix $\mathbf{w}$, where all entries in the matrix equal $\frac{1}{n}$.

$$\mathbf{w} = \hat{\mathbf{w}}(\mathbf{x_i}, \mathbf{x}) \tag{7}$$

$$= \begin{bmatrix} \frac{1}{n} & ... & \frac{1}{n} \\ ... & ... & ... \\ \frac{1}{n} & ... & \frac{1}{n} \end{bmatrix} \tag{8}$$

By equation 1.70 in ADA, the rank of this matrix is n as shown below.

$$df(\hat{\mu}) = tr(\mathbf{w}) \tag{9}$$

$$= tr(\begin{bmatrix} \frac{1}{n} & ... & \frac{1}{n} \\ ... & ... & ... \\ \frac{1}{n} & ... & \frac{1}{n} \end{bmatrix}) \tag{10}$$

$$= \sum_{j=1}^{n} \frac{1}{n} \tag{11}$$

$$= 1 \tag{12}$$

Question 5 (ADA 1.8)

Using k-nearest-neighbors regression as a linear smoother, we weigh the distance between points $\mathbf{x}$ and $\mathbf{x_i}$ as $\frac{1}{k}$ if $\mathbf{x_i}$ is one of the k-nearest neighbors of $\mathbf{x}$ and 0 otherwise. So, in this sense, it is a local averaging that only considers the k-nearest points from $\mathbf{x}$. In this case, we have the n by n influence matrix $\mathbf{w}$, where the off diagonal entries are 0 and on the diagonal is $\frac{1}{k}$.

$$\mathbf{w} = \hat{\mathbf{w}}(\mathbf{x_i}, \mathbf{x}) \tag{13}$$

$$= \begin{bmatrix} \frac{1}{k} & 0 & 0 \\ 0 & ... & 0 \\ 0 & 0 & \frac{1}{k} \end{bmatrix} \tag{14}$$

By equation 1.70 in ADA, the rank of this matrix is $\frac{n}{k}$ as shown below.

$$df(\hat{\mu}) = tr(\mathbf{w}) \tag{15}$$

$$= tr(\begin{bmatrix} \frac{1}{k} & 0 & 0 \\ 0 & ... & 0 \\ 0 & 0 & \frac{1}{k} \end{bmatrix}) \tag{16}$$

$$= \sum_{j=1}^{n} \frac{1}{k} \tag{17}$$

$$= \frac{n}{k} \tag{18}$$

Note: when $k = n$, it reduces to the previous case.

Question 6 (ESL 2.8)

For the classification with linear regression, we choose any value that is greater than 2.5 to be 3 and 2 otherwise. We see from the results below that a knn classifier with $k = 1$ performs best on the train and test sets with error rates of 0% and 2.47% respectively.

```
#install.packages("class")
#install.packages("tidyverse")

#function for computing error rate
error_rate <- function(x,y) {
  1 - mean(x == y)
}

#load packages
library(class) #knn
library(tidyverse)

#load the zipcode dataset
train_raw <- read.table("zip.train", header = FALSE)
test_raw <- read.table("zip.test", header = FALSE)

#need to filter out the data for 2's and 3's
train <- train_raw %>%
  filter(V1 ==  2 | V1 ==  3)

test <- test_raw %>%
  filter(V1 == 2 | V1 == 3)

train_x <- select(train, -V1)
```

```r
test_x <- select(test, -V1)

train_y <- pull(train, V1)
test_y <- pull(test, V1)

#linear regression
lm_train <- lm(V1 ~., data = train)

lm_pred_train <- ifelse(predict(lm_train, train) > 2.5, 3, 2)
lm_pred_test <- ifelse(predict(lm_train, test) > 2.5, 3, 2)

#k-nearest neighbors
test_knn_1 <- knn(train_x,
                  test_x,
                  train_y,
                  k = 1)

test_knn_3 <- knn(train_x,
                  test_x,
                  train_y,
                  k = 3)

test_knn_5 <- knn(train_x,
                  test_x,
                  train_y,
                  k = 5)

test_knn_7 <- knn(train_x,
                  test_x,
                  train_y,
                  k = 7)
```

```
test_knn_15 <- knn(train_x,
                   test_x,
                   train_y,
                   k = 15)

train_knn_1 <- knn(train_x,
                   train_x,
                   train_y,
                   k = 1)

train_knn_3 <- knn(train_x,
                   train_x,
                   train_y,
                   k = 3)

train_knn_5 <- knn(train_x,
                   train_x,
                   train_y,
                   k = 5)

train_knn_7 <- knn(train_x,
                   train_x,
                   train_y,
                   k = 7)

train_knn_15 <- knn(train_x,
                   train_x,
                   train_y,
                   k = 15)

#train and test error rates
```

```r
error_rates <- data.frame(Train = c(error_rate(lm_pred_train, train_y),
                                    error_rate(train_knn_1, train_y),
                                    error_rate(train_knn_3, train_y),
                                    error_rate(train_knn_5, train_y),
                                    error_rate(train_knn_7, train_y),
                                    error_rate(train_knn_15, train_y)),
                          Test = c(error_rate(lm_pred_test, test_y),
                                   error_rate(test_knn_1, test_y),
                                   error_rate(test_knn_3, test_y),
                                   error_rate(test_knn_5, test_y),
                                   error_rate(test_knn_7, test_y),
                                   error_rate(test_knn_15, test_y)))

error_rates
```

```
##         Train       Test
## 1 0.005759539 0.04120879
## 2 0.000000000 0.02472527
## 3 0.005039597 0.03021978
## 4 0.005759539 0.03021978
## 5 0.006479482 0.03296703
## 6 0.009359251 0.03846154
```