# Stats_790_Assignment_1

Amandeep Sidhu

2023-01-19

**I collaborated with Yiran Zhang on Question 2 and Cameron Roopnarine on Question 3**

## Question 1: Write a short description of any opinions, thoughts, or questions you are left with after reading Breiman, 2001, and one of the responses to it

**Short Description of the Breiman Paper:**

The first thought on the article by Breiman is that the Occam dilemma is fascinating because it suggests that accuracy requires more complex prediction and that the simple techniques do not produce the most accurate results in practice. In addition, another thought I had was that as the dimensionality increases then the likeliness that separation occurs increases as well. This opposes my original opinion because I would have assumed adding more input features would cause more noise for the overall plane. However, one explanation might be that when adding features you can seperate the input features more easily (less dependent on one another).

**Short Description of a Response to the Breiman Paper: Gelman, 2021**

I agree with Gelmans analysis of the Breiman article where the problem of taking inference from existing data and applying it to new data to predict new problems still exists in modern statistics. This feels like the crux of modern day statistics given the rise of machine learning algorithms to try and predict how models behaviour when trained on data sets. Moreover, Gelman argues that model checking is important as we must try to improve models when necessary as in certain applications a better model is crucial. For example, heart disease prediction given several predictors for the patient.

## Question 2: Pick a figure from ESL Chapter 2 and write, R, Python, or Julia code to replicate it

For the purpose of this assignment, I will be selecting Figure 2.1, which is the Linear Regression 0/1 Response. Using the description of the two datasets, it appears that the Blue class was generated using 10 means $m_k$ from a bivariate Gaussian Distribution $N((1,0)^T, I)$. Then, another 10 more means were drawn from a similar distribution and this was labelled as orange. Finally, to obtain the sample, 100 observations were obtained from each class, where $m_k$ was selected with a probability of 1/10. See pages 16-17 in ESL for more details.

Now, let us set up the orange and blue class.

```
#Import the necessary packages
library(MASS) #this will be used to create the required distribution
library(dplyr) #for data processing

set.seed(1234) #ensure reproducibility for the grader
```

```r
#let us create 10 means for orange and blue respectively using the definition in
#the textbook provided

#I use mvrnorm function as we are required to find a bivariate distribution
#Recall from the question, mu = (1,0)' and sigma = I

blue_class <- mvrnorm(n = 10, mu = c(1,0),
                      Sigma = matrix(c(1,0,0,1), ncol = 2))
orange_class <- mvrnorm(n = 10, mu = c(1,0),
                        Sigma = matrix(c(1,0,0,1), ncol = 2))

#Now, we must draw 100 samples from both classes with specifications listed in
#ESL book


#Remember that each sample is drawn with probability 1/10

#sample for blue
means_blue <- blue_class[sample(nrow(blue_class), 100,
                                replace = TRUE, prob = rep(0.1, 10)),]

#sample for orange
means_orange <- blue_class[sample(nrow(orange_class), 100,
                                  replace = TRUE, prob = rep(0.1, 10)),]

#Now, by Page 17 of ESL, we must obtain observations for orange and blue that
#follow N(m_k, I/5)

blue_observations <- matrix(ncol = 2, nrow = 100)

for (i in 1:100){
  blue_observations[i,] <- mvrnorm(n = 1, mu = means_blue[i,],
                                   Sigma = matrix(c(1/5,0,0,1/5), ncol = 2))
}

#repeat for orange
orange_observations <- matrix(ncol = 2, nrow = 100)

for (i in 1:100){
  orange_observations[i,] <- mvrnorm(n = 1, mu = means_orange[i,],
                                   Sigma = matrix(c(1/5,0,0,1/5), ncol = 2))
}


#Now, we have the observations and we must join them
df <- as.data.frame(rbind(blue_observations, orange_observations))

#Find the linear regression coefficients as this is the analysis performed

lin_mid <- lm(V2 ~ V1, data = df)

#Use lin_mid$Coefficients to find coefficients
B0 = -0.5135887
```

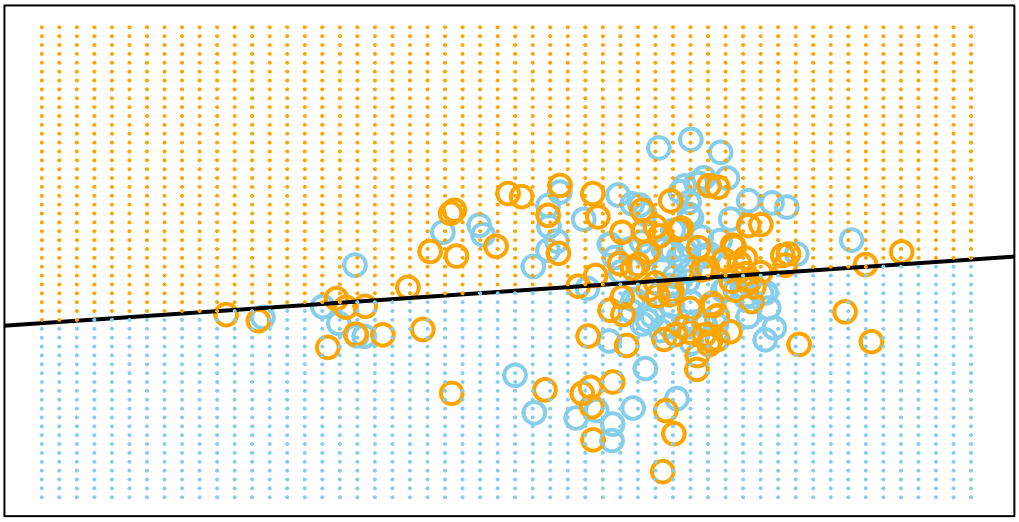
```r
B1 = 0.1353249

linear <- function(x) {
  B0 + B1*x
}

#background
x <- seq(from = -4, to = 4, by = 0.15)
y <- seq(from = -4, to = 4, by = 0.15)
background = expand.grid(x,y)
#filter out the line points
blue_bg_pt <- background %>% filter(Var2 < linear(x))
orange_bg_pt <- background %>% filter(Var2 >= linear(x))


#Let us Attempt to create the figure (will not be identical as the points are
#random)
plot(blue_observations, col = "skyblue", xlim = c(-4, 4), ylim = c(-4,4),
     lwd = 2, cex = 1.5, xlab = '', ylab = '', xaxt = 'n', yaxt= 'n')
#Note, xaxt and yaxt are 'n' since the book had no indicators
points(orange_observations, col='orange', lwd= 2, cex = 1.5)
abline(lin_mid, col = 'black', lwd = 2)

#Background dots trying to replicate the grid in the book
points(orange_bg_pt, col = "orange", pch = 20, cex = 0.2)
points(blue_bg_pt, col = "skyblue", pch = 20, cex = 0.2)
```



Since, the simulation is sampled randomly this is the best we can do to try and replicate it. The process however is consistent with the process listed in the book.

## Question 3: ADA Problem 1.2

In theory, the MAE is not minimized by m = E[Y}, but rather the **median**. The following steps will prove that the median minimizes the MAE.

Suppose m is some term in the interval and m' is the mean.

$$E[|X - m|] = \int_{-\infty}^{\infty} |x - m| f(x) dx$$

$$= \int_{-\infty}^{m} (m - x) f(x) dx + \int_{m}^{\infty} (m - x) f(x) dx$$

$$= \int_{-\infty}^{m} (m - x) f(x) dx + \int_{m}^{m'} (x - m) f(x) dx + \int_{m'}^{\infty} (x - m) f(x) dx$$

$$= \int_{-\infty}^{m'} (m - x) f(x) dx - \int_{m}^{m'} (m - x) dx + \int_{m}^{m'} (x - m) f(x) dx + \int_{m'}^{\infty} (x - m) f(x) dx$$

All that has been done is breaking up the absolute value into two portions. Now, with the help of some algebra properties, we can reduce the equation further as so.

$$= \int_{-\infty}^{m'} (m - m' + m' - x) f(x) dx + 2 \int_{m}^{m'} (x - m) f(x) dx + \int_{m'}^{\infty} (x - m' + m' - m) f(x) dx$$

$$= E[|x - m|] + 2 \int_{m}^{m'} (x - m) f(x) dx + \int_{-\infty}^{m'} (m - m') f(x) dx + \int_{m'}^{\infty} (m' - m) f(x) dx$$

Note, this cannot be reduced further without the specific values. But, if the mean were equal to the median we could reduce it even further.

Now, we must take the derivative and show that the median does indeed minimize the MAE.

Suppose m is the median.

$$\frac{\partial}{\partial m} E[|x - m|] = E[\frac{\partial}{\partial m} |x - m|]$$

$$= E[\frac{-1(x - m)}{|x - m|}]$$

$$= E[1_{x<m} - 1_{x>m}]$$

The 1's in the above are the indicator function. Breaking this into probabilities and setting it equal to zero, we get

$$P(x < m) - P(x > m) = 0$$

Now, the only way this is equal to zero is if both terms equal 1/2. But if both of those two terms are 1/2, that means that the median is the minimizing term by definition (i.e. m is the median here).

The last part of this question asks whether to use MSE or MAE to measure error. In theory, since the median is more robust to outliers, if the data set may contain outliers it might be best to use the MAE as the minimizing term is the median. However, if the data set contains not many outliers then it may be more optimal to use the MSE as its minimizing term is the mean (as shown in the textbook).

## Question 4: ADA Problem 1.7

Suppose that the global mean is our linear smoother in this case. Recall that our influence matrix, $w$ is a $nxn$ matrix with the weight, $w_{ij}$ saying how much each observation $y_{ij}$ contributes to the fitted values.

Observing the classic mean formula of

$$\mu = (\frac{1}{n}) * (\Sigma_i^n y_i)$$

We observe the weight is essentially $\frac{1}{n}$ for every entry of the influence matrix $w$.

To observe this property visually, we construct an influence matrix as so,

$$w = \begin{pmatrix} 1/n & 1/n & \cdots & 1/n \\ 1/n & 1/n & \cdots & 1/n \\ \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/n & \cdots & 1/n \end{pmatrix}$$

This makes sense as each component has equal influence on the observation.

By Equation (1.70) of the textbook that states

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

we observe that the trace of $w$ is essentially $tr(w) = \frac{1}{n} + \frac{1}{n} + ... + \frac{1}{n} = 1$.

Thus we conclude that the degrees of freedom for when the global mean is the linear smoother is $df = 1$.

## Question 5: ADA Problem 1.8

Let us consider the case when k-nearest neighbors regression acts as our linear smoother. Recall that our influence matrix, $w$ is a $nxn$ matrix with the weight, $w_{ij}$ saying how much each observation $y_{ij}$ contributes to the fitted values.

We observe in (1.55) of the textbook that $\hat{(}w(x_i, x))$ is equal to $1/k$ when $x_i$ is one of the k nearest neighbors of x and 0 otherwise.

As a result, we obtain a similar matrix to the one in Question (4), where the diagonal entries are strictly $1/k$, which makes sense since the distance would be 0 about the entries of x if they are not considered a k nearest neighbor.

Thus an approximate matrix $w$ for this question can be constructed as so,

$$w = \begin{pmatrix} 1/k & \cdots & \cdots & 0 \\ & 1/k & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1/k \end{pmatrix}$$

Since the matrix is once again $nxn$, we can take the trace of the matrix by Equation (1.70) to obtain the following,

$$tr(w) = \frac{1}{k} + \frac{1}{k} + ... + \frac{1}{k} = \frac{n}{k}$$

As a result, we see that the degrees of freedom for when k-nearest neighbor regression is a linear smoother is $df = \frac{n}{k}$.

## Question 6: ESL Problem 2.8

The zipcode data from the book is used for this question as outlined in ESL Chapter 2.

```
library(class) #this will be used for knn regression

#set working directory - this was done in rstudio as outlined by Ben
# setwd('C:/Users/User/Documents/stat790/Stats_790_Assignment_1')

#load in the datasets available online
#I used as.matrix to get the appropriate data type later during the class
zip.train <- as.matrix(read.table(gzfile('zip.train.gz')))
zip.test <- as.matrix(read.table(gzfile('zip.test.gz')))

#Now, as outlined in the question, we must only consider the 2's and 3's
#Create the training data sets
X_train <- zip.train[which(zip.train[, 1] == 2 | zip.train[, 1] == 3), -1]
y_train <- zip.train[which(zip.train[, 1] == 2 | zip.train[, 1] == 3), 1] == 3
```

```r
#Create the test data sets

X_test <- zip.test[which(zip.test[, 1] == 2 | zip.test[, 1] == 3), -1]
y_test <- zip.test[which(zip.test[, 1] == 2| zip.test[, 1] == 3), 1] == 3

#Notice that i set the y's as equal to 3 to return a true or false value, this
#will be important later as we need to differentiate between a 2 or 3


#Now, let us compute the linear classification using R

linear_classification <- lm(y_train ~ X_train) #train using the datasets

#set cut-off for linear classification as 0.5

#Find train error rate
y_hat_train <- (cbind(1, X_train) %*% linear_classification$coefficients) >= 0.5
train_error <- mean(y_hat_train != y_train)

#Find test error rate
y_hat_test <- (cbind(1, X_test) %*% linear_classification$coefficients) >= 0.5
test_error <- mean(y_hat_test != y_test)


#Now, compute the k-nn classification for the values given i.e. 1,3,5,7,15

k <- c(1,3,5,7,15)

#set up an empty vector for both the training and test error rates
k_train_error <- rep(NA, 5)
k_test_error <- rep(NA, 5)

#create a for loop to iterate through all the k's

for (i in 1:length(k)){

  y_hat_train <- knn(X_train, X_train, y_train, k[i])
  y_hat_test <- knn(X_train, X_test, y_train, k[i])
  k_train_error[i] <- mean(y_hat_train != y_train)
  k_test_error[i] <- mean(y_hat_test != y_test)
}

#throw it all in a table for easy visibility
classification_type <- c("Linear Classification", "k-NN with K = 1",
                         "k-NN with K = 3", "k-NN with K = 5",
                         "k-NN with K = 7", "k-NN with K = 15")
training_error_rates <- c(0.005759539, 0.000000000, 0.005039597, 0.005759539,
                          0.006479482, 0.009359251)
test_error_rates <- c(0.04120879, 0.02472527, 0.03021978, 0.03021978,
                      0.03296703, 0.03846154)
tab <- data.frame(classification_type, training_error_rates,
                    test_error_rates)
tab
```

```
##     classification_type training_error_rates test_error_rates
## 1 Linear Classification          0.005759539       0.04120879
## 2       k-NN with K = 1          0.000000000       0.02472527
## 3       k-NN with K = 3          0.005039597       0.03021978
## 4       k-NN with K = 5          0.005759539       0.03021978
## 5       k-NN with K = 7          0.006479482       0.03296703
## 6      k-NN with K = 15          0.009359251       0.03846154
```

As you can see in the table above, the error rate performs best for both the train and test data sets with K = 1 for the k-NN classification technique. As a result, you could argue the smaller the k, the better the model performance. One possible hypothesis that I propose is the argument involving the curse of dimensionality. With the data points far apart, they could actually not be considered near one another.