STATS 790 - Statistical Learning
Assignment 1

Tommy Flynn

20 January, 2023

Contents

Question 1:

After reading Breiman (2001) I was quite shocked at how aggressively algorithmic models were trumpeted over data models. It seems that at the time algorithmic models were unappreciated but now they are used excessively. In my previous industry positions, tree ensembles and deep neural networks were applied to almost everything. I enjoyed reading the response Miller (2021) which proposed that the problem and data at hand should inform where to select a model in the data, algorithmic, and mechanistic model space. I am left with the thought that the nuances of selecting a model from this space must be communicated to practitioners so that they don't just throw large algorithmic models at every dataset and problem that arises.

Question 2:

For this question, I recreated Figure 2.2 from page 15 of ESL. Below one can find the original figure and my version followed by the r code used to create it.

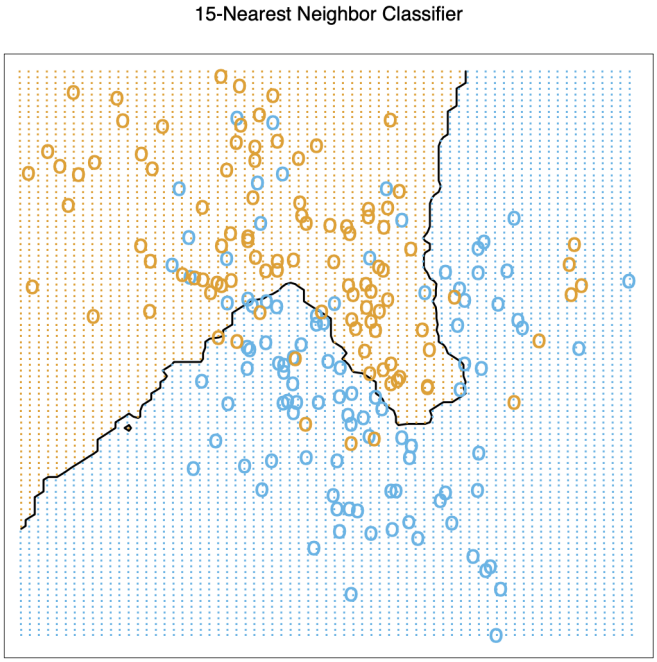

Figure 1: Original Figure 2.2 from ESL.

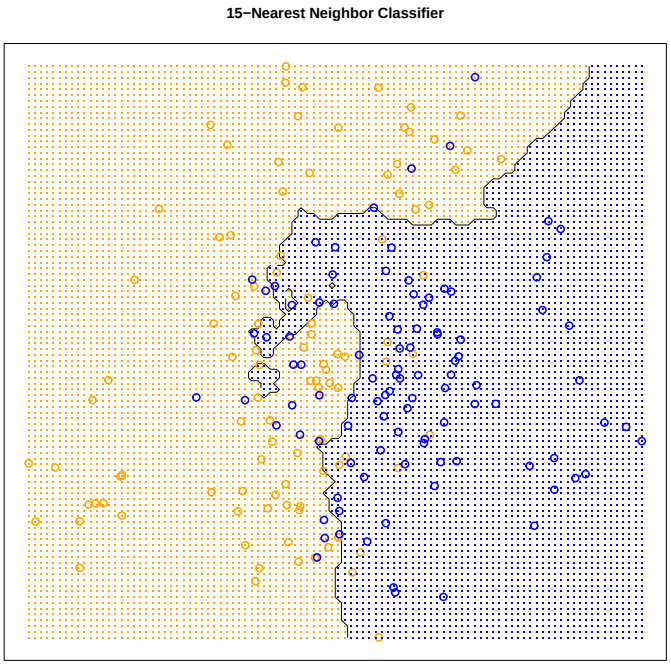

Figure 2: Recreation of Figure 2.2.

```r
# set-up
library(MASS)
library(class)
set.seed(13)

# function to generate Gaussian clusters
GaussianCluster <- function(n, mu, Sigma, sims) {

  means <- mvrnorm(n=n, mu=mu, Sigma=Sigma) # generate n means from N(mu, sigma)
  xs <- c()                                 # x coordinates for output
  ys <- c()                                 # y coordinates for output

  # generate sims observations
  for (i in 1:sims) {

    # select index of mean and generate observation
    row.sample <- sample(x=1:n, size=1, replace=TRUE)
    observation <- mvrnorm(n=1, mu=means[row.sample, ], Sigma=diag(2)/5)

    # populate the x,y coordinates
    xs[i] <- observation[1]
    ys[i] <- observation[2]
  }

  # return coordinates in a data frame
  return(data.frame(xs, ys))
}

# generate blue and orange samples
obs <- 100
blue <- GaussianCluster(n=10, mu=c(1, 0), Sigma=diag(2), sims=obs)
orange <- GaussianCluster(n=10, mu=c(0, 1), Sigma=diag(2), sims=obs)
```

```r
# code the colors and group into one data frame
blue$class <- 0
orange$class <- 1
cluster.df <- rbind(blue, orange)

# domain for plot
x_min <- min(cluster.df$xs)
x_max <- max(cluster.df$xs)
y_min <- min(cluster.df$ys)
y_max <- max(cluster.df$ys)
grid.x <- seq(from=x_min, to=x_max, length.out=obs)
grid.y <- seq(from=y_min, to=y_max, length.out=obs)
domain <- expand.grid(x1=grid.x, x2=grid.y)

# knn
model.knn <- knn(train=cluster.df[,1:2], test=domain, cl=cluster.df$class, k=15)
matrix.knn <- matrix(model.knn, nrow=obs, ncol=obs)

# decision boundary
contour(x=grid.x, y=grid.y, z=matrix.knn, levels=0.5, xaxt='n', col='black',
        yaxt='n', ann=FALSE, labels='', main='15-Nearest Neighbor Classifier')

# orange data points and background
points(x=orange$xs, y=orange$ys, col='orange', lwd=2, cex=1.25)
points(domain[which(matrix.knn>0.5),], col='orange', pch='.')

# blue data points and background
points(x=blue$xs, y=blue$ys, col='blue', lwd=2, cex=1.25)
points(domain[which(matrix.knn<0.5),], col='blue', pch='.')
```

Question 3: (ADA 1.2)

Let Y be a random variable with density $f_Y(y)$ and let $m \in \mathbb{R}$. Then the mean absolute error is given by the following.

$$
\begin{aligned}
&\text{MAE}(m) \\
&= \text{E}[|Y - m|] \\
&= \int_{-\infty}^{\infty} |y - m| f_Y(y)\, dy \\
&= \int_{-\infty}^{m} (m - y) f_Y(y)\, dy + \int_{m}^{\infty} (y - m) f_Y(y)\, dy
\end{aligned}
$$

Setting the derivative to 0 we have,

$$
\begin{aligned}
&\frac{d\text{MAE}}{dm} = 0 \\
&\iff \int_{-\infty}^{m} f_Y(y)\, dy - \int_{m}^{\infty} f_Y(y)\, dy = 0 \\
&\iff F_Y(m) - (1 - F_Y(m)) = 0 \\
&\iff F_Y(m) = \frac{1}{2}
\end{aligned}
$$

This means that the MAE is minimized by taking m to be the median. The MSE is preferred to the MAE because it has a much more computationally friendly form. In particular, it is differentiable and well suited for optimization.

Question 4: (ADA 1.7)

The global mean as a linear smoother results in every observation y_i contributing equally to $\hat{\mu}_i$, in other words $w_{ij} = \frac{1}{n}$. Thus, the influence matrix will be a scaled matrix of ones $w = \frac{1}{n} J_n$. Then the degrees of freedom is given by $\mathrm{df}(\hat{\mu}) = \mathrm{tr}(w) = \mathrm{tr}(\frac{1}{n} J_n) = \sum_{i=1}^{n} \frac{1}{n} = 1$.

Question 5: (ADA 1.8)

k-nearest-neighbors regression as a linear smoother results in the closest k observations to y_i contributing equally to $\hat{\mu}_i$. Thus, each row of the influence matrix will contain exactly k nonzero entries with a value of $\frac{1}{k}$ at indices corresponding to y_i's nearest neighbors. The remaining $n-k$ entries will contain zeros for non-neighbor indices. Note that the diagonal elements will always be assigned a value of $\frac{1}{k}$ since y_i is always a nearest neighbor of itself. Therefore, the degrees of freedom is given by $\mathrm{df}(\hat{\mu}) = \mathrm{tr}(w) = \sum_{i=1}^{n} \frac{1}{k} = \frac{n}{k}$. We also note that when $k = n$ we achieve the same results as the global mean linear smoother from question 4 as every row will be populated with n values of $\frac{1}{n}$.

Question 6: (ESL 2.8)

Below is r code to perform linear regression and knn on the zipcode data. The error values from the output are summarized in the subsequent table. From the table, we see that the test error for linear regression is inferior to all knn models. Notably, as k increases the test error increases. Thus, the minimal test error occurs at $k = 1$ with an error of 2.47%.

Table 1: Train and test error for the zipcode models.

Model	Train Error	Test Error
Linear Regression	0.58%	4.12%
KNN (1)	0%	2.47%
KNN (3)	0.43%	3.02%
KNN (5)	0.58%	3.02%
KNN (7)	0.58%	3.02%
KNN (15)	0.94%	3.85%

```
# set-up
library(class)
set.seed(13)

# load data
zip.train <- read.table(gzfile('zip.train'))
zip.test <- read.table(gzfile('zip.test'))

# only include rows of class 2 or 3
zip.train <- zip.train[zip.train$V1 == 2 | zip.train$V1 == 3, ]
zip.test <- zip.test[zip.test$V1 == 2 | zip.test$V1 == 3, ]

# performance function
performance <- function(actual, predicted, name){
  confusion.matrix <- table(actual, predicted)
  error <- 1 - sum(diag(confusion.matrix))/sum(confusion.matrix)
  # print(confusion.matrix)
  cat(name, 'Error:', error, '\n')
}
```

```r
# linear model
zip.lm <- lm(V1 ~., data=zip.train)

# train and test error
performance(zip.train$V1, round(zip.lm$fitted.values), 'LM Train')
```

```
## LM Train Error: 0.005759539
```

```r
performance(zip.test$V1, round(predict(zip.lm, zip.test)), 'LM Test')
```

```
## LM Test Error: 0.04120879
```

```r
# knn
for (i in c(1, 3, 5, 7, 15)){

  # predictions
  zip.knn.train <- knn(train=zip.train, test=zip.train, cl=zip.train$V1, k=i)
  zip.knn.test <- knn(train=zip.train, test=zip.test, cl=zip.train$V1, k=i)

  # errors
  performance(zip.train$V1, zip.knn.train, cat('KNN Train', i))
  performance(zip.test$V1, zip.knn.test, cat(' KNN Test', i))
  cat('\n')
}
```

```
## KNN Train 1 Error: 0
##  KNN Test 1 Error: 0.02472527
##
## KNN Train 3 Error: 0.004319654
##  KNN Test 3 Error: 0.03021978
##
## KNN Train 5 Error: 0.005759539
##  KNN Test 5 Error: 0.03021978
##
```

```
## KNN Train 7 Error: 0.005759539
##  KNN Test 7 Error: 0.03021978
##
## KNN Train 15 Error: 0.009359251
##  KNN Test 15 Error: 0.03846154
```