STAT790 HW1

2023-01-18

Q1

When we have a large dataset and the data shows that it could fit a statistical model well, the data modeling culture works; however, lots of dataset do not have a statistical model pattern or it is not large enough; therefore, I believe that the algorithmic modeling culture would work better in most cases.

Q2

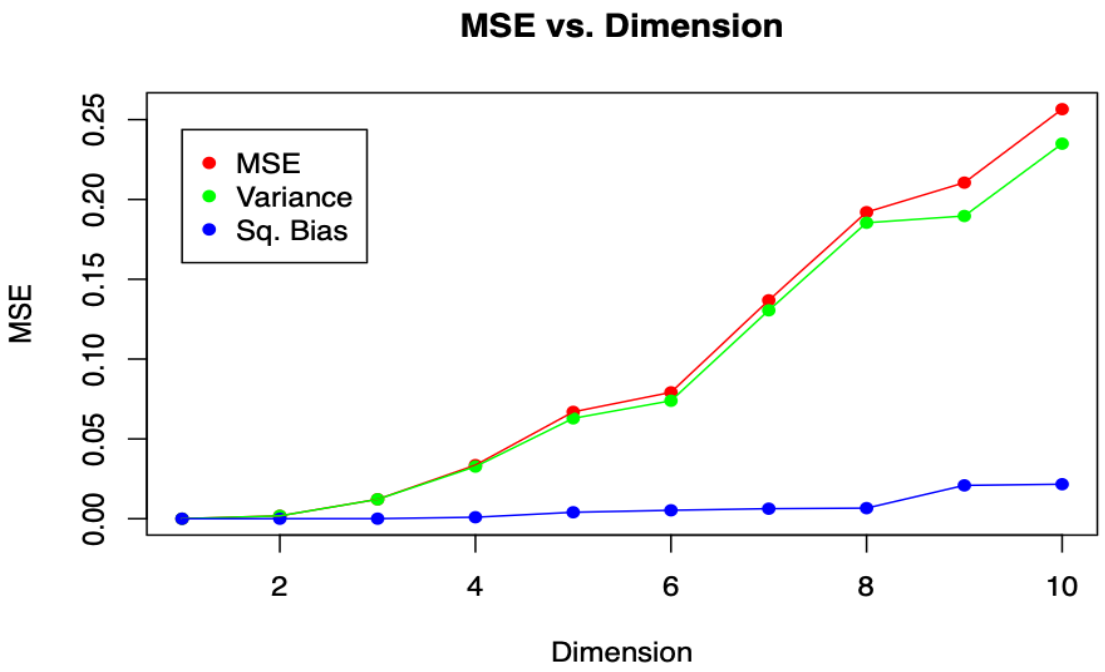
```
library(class)
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
```

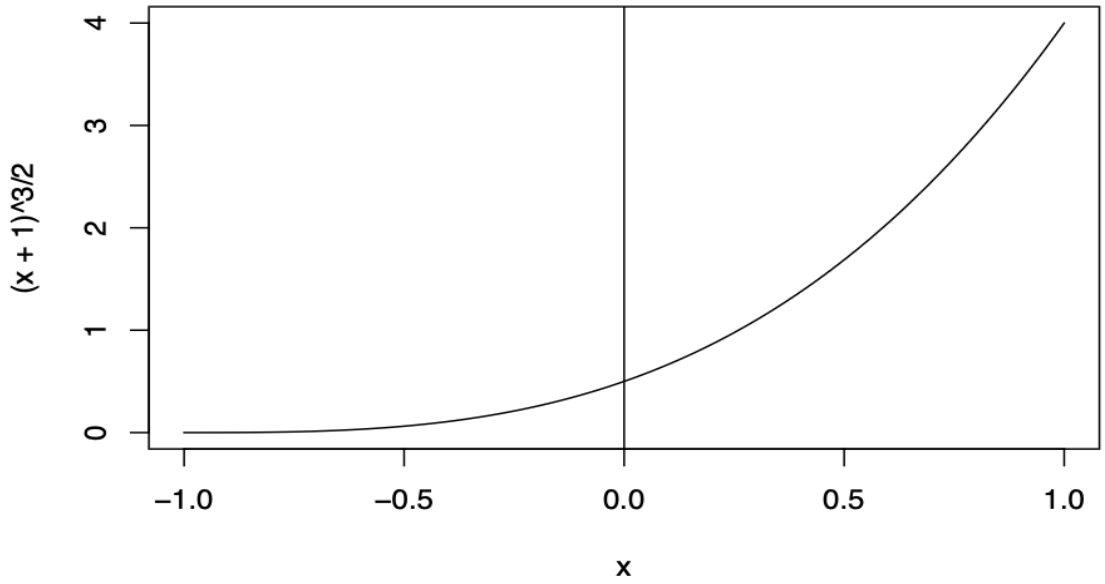
```
## The following object is masked from 'package:base':
##
##     Recall
```

```
f_x <- function(row) 0.5*((row[1] + 1)^3)
f_x0 <- rep(0.5,100)


training_data1 <- function(dim){
  dim = 1:dim
  tradata = matrix(nrow = 1000, ncol = length(dim))
  for (i in dim){
    column <- runif(n = 1000, min = -1, max = 1)
    tradata[,i] <- column
  }
  training_labels = apply(tradata, 1, f_x)
  return(list(tradata, training_labels))
}


output <- list()
for(dim in 1:10){
  dimension <- list()
  for(repetition in 1:100){
    output1 <- training_data1(dim)
    training_data <- output1[1][[1]]
    training_labels <- output1[2][[1]]
    test_observation <- rep(0, dim)
    prediction <- knn(training_data, test_observation, training_labels, k = 1)
    dimension[[repetition]] <- prediction
  }
  output[[dim]] <- dimension
}
```

```
mse_list <- list()
variance_list <- list()
bias_list <- list()
count <- 1
for(dim in output){
  f = unlist(dim)
  prediction <- as.numeric(levels(f))[f]
  mse_list[[count]] <- MSE(prediction, f_x0)
  y_hat <- mean(prediction)
  variance_list[[count]] <- MSE(prediction, rep(y_hat,100))
  bias_list[[count]] <- MSE(rep(y_hat,100), f_x0)
  count = count + 1
}

plot(unlist(mse_list), xlab = "Dimension", ylab = "MSE", col = 'red', pch = 16, main = "MSE vs. Dimensi
points(unlist(variance_list), col = 'green', pch = 16)
points(unlist(bias_list), col = 'blue', pch = 16)
lines(unlist(mse_list), col = 'red')
lines(unlist(variance_list), col = 'green')
lines(unlist(bias_list), col = 'blue')
y_legend <- max(unlist(mse_list))*0.95
legend(x = 1, y = y_legend, legend = c("MSE", "Variance", "Sq. Bias"), col = c('red', 'green', 'blue'),
```




```
x=seq(-1,1,0.01)
plot(x,(x+1)^3/2,type="l")
abline(v=0)
```



Q3

MAE(m)=E(|Y-m|)=$E(\sqrt{(Y-m)^2})$

We have $\frac{dMAE(\hat{m})}{d\hat{m}} = \frac{d(\int_{-\infty}^{\infty}|Y-\hat{m}|f(y)dy = \int_{-\infty}^{\hat{m}} -(Y-\hat{m})f(y)dy + \int_{\hat{m}}^{\infty}(Y-\hat{m})f(y)dy)}{d\hat{m}} = 0$

So we have $\int_{-\infty}^{\hat{m}} f(y)dy + \int_{\hat{m}}^{\infty} -f(y)dy = 0$, therefore, $f(\hat{m}) = 0.5$.

$\tilde{\mu} = median(y)$ minimize MAE instead of E(Y).

I would suggest to use MSE to measure errors, since MSE considers both Expected value and Variance of the random variable, and MAE only consider the median value of random variable. Also, the MSE emphasized on the large errors, therefore, MSE could have a better performance when there is large error occurred.

Q4

Linear smoothers: $\hat{\mu}(x) = \sum_i y_i \hat{w}(x_i, x)$

In our question, we have $\hat{\mu}(x) = \frac{\sum_i^n y_i}{n}$, therefore, $\hat{w}(x_i, x) = \frac{1}{n}$ at this time.

we know that the degree of freedom is the trace of influence matrix w, where $w_{i,j} = \hat{w}(x_i, x_j) = 1/n$ and w is a n*n matrix.

therefore, the degree of freedom $df(\hat{\mu}) = tr(w) = \sum_{i=1}^{n} \frac{1}{n} = 1$.

Q5

From equation 1.55 we know that $\hat{w}(x_i, x) = 1/k$, when x_i is one of the k nearest neighbours of x, and 0 otherwise.

$df(\hat{\mu}) = tr(w) = \sum_{i=1}^{n} w_{i,i}$, since $x_i = x_i$, which means x_i must be one of the k nearest neighbours of x_i

$df(\hat{\mu}) = tr(w) = \sum_{i=1}^{n} w_{i,i} = \sum_{i=1}^{n} \frac{1}{k} = \frac{n}{k}$

Q6

```
library(class)

train <- read.table("zip.train")
```

```
test <- read.table("zip.test")

train <- train[(train$V1==2) | (train$V1==3),]
test <- test[(test$V1==2) | (test$V1==3),]

mod <- glm(V1 ~ . , data = train)

round_to_class_values <- function(value){
  if (value > 2.5) return(3)
  else (return(2))
}

pred_train <- predict(mod,train[,-1])
pred_test <- predict(mod, test[,-1])

pred_train_rounded <- sapply(pred_train, round_to_class_values)
pred_test_rounded <- sapply(pred_test, round_to_class_values)

mse_train_glm <- mean((train$V1 - pred_train_rounded)^2)
mse_test_glm <- mean((test$V1 - pred_test_rounded)^2)

K <- c(1,3,5,7,15)
train_mse <- list()
test_mse <- list()

for (k in K){
  train_pred_knn <- knn(train[,-1], train[,-1], train$V1, k = k)
  test_pred_knn <- knn(train[,-1], test[,-1], train$V1, k = k)

  train_pred_knn <- as.numeric(levels(train_pred_knn))[train_pred_knn]
  test_pred_knn <- as.numeric(levels(test_pred_knn))[test_pred_knn]

  mse_train_knn <- mean((train$V1 - train_pred_knn)^2)
  mse_test_knn <- mean((test$V1 - test_pred_knn)^2)

  train_mse <- append(train_mse, mse_train_knn)
  test_mse <- append(test_mse, mse_test_knn)
}


plot(y = unlist(train_mse), x = K, type='b',ylim=c(0,0.06),
     col = 'red', xaxt = 'n', ylab = 'MSE', main = 'Training vs Test Error')
axis(side=1,at=K)
points(y = unlist(test_mse), x = K, type='b', col = 'blue', pch = 19)
abline(h = mse_test_glm, col = 'green')
abline(h = mse_train_glm, col = 'purple')
legend(x=11.5,y=0.06,c("KNN train MSE", "KNN test MSE", "GLM train MSE", 'GLM test MSE'),
       col = c('red', 'blue', 'purple', 'green'),pch=19)
```


```
table(pred_train_rounded, train$V1)
```

```
##
## pred_train_rounded   2   3
##                  2 728   5
##                  3   3 653
```

```
table(pred_test_rounded, test$V1)
```

```
##
## pred_test_rounded   2   3
##                 2 191   8
##                 3   7 158
```