HW 5

Wenjuan Bian

08032023

Provide your answers and relevant outputs below each question in this markdown file and submit the knitted `html` or `pdf` file to Canvas. Put your answers/comments/conclusions in **bold**.

Scalar-on-function GLM

Suppose we simulate 250 independent 2D point clouds, where each point cloud consists of 50 points sampled uniformly from the *unit circle* (shape 1) with some Gaussian noise added to the x and y coordinates. In a similar fashion, we simulate another 250 independent point clouds sampled from *two concentric circles with radii 0.9 and 1.1* (shape 2). The response variable Y_i is binary: $Y_i = 0$ if the point cloud is sampled from shape 1 and $Y_i = 1$ if sampled from shape 2 ($i = 1, 2, \ldots, 500$). The shapes of the point clouds can be described by two functions $X_{1i}(t)$ and $X_{2i}(t)$, called *persistence silhouettes* (of homological dimensions H_0 and H_1). The goal is to build a *scalar-on-function* GLM to predict Y_i (shape 1 or 2) using persistence silhouette functions either $X_{1i}(t)$ or $X_{2i}(t)$ (but not both).

The files `P2Ytrain.csv` and `P2Ytest.csv` contain 500 observed values of Y_i of the train and test datasets respectively. `P2X1train.csv` and `P2X2train.csv` store the values of $X_{1i}(t)$ and $X_{2i}(t)$ observed at 100 equally spaced points $t_j, j = 1, 2, \ldots, 100$ for the training set. Similarly, `P2X1test.csv` and `P2X2test.csv` contain the test data.

a. Using `matplot()` function (with type='l' inside of it), plot the raw values of $X_{1i}(t)$ and $X_{2i}(t)$ for the training set. Color the curves based on the shape type (shape 1 or shape 2). Looking at the plots, which of the two sets of curves is more informative to distinguish the two shapes?

```
library(fda)
```

```
## Loading required package: splines
```

```
## Loading required package: fds
```

```
## Loading required package: rainbow
```

```
## Loading required package: MASS
```

```
## Loading required package: pcaPP
```

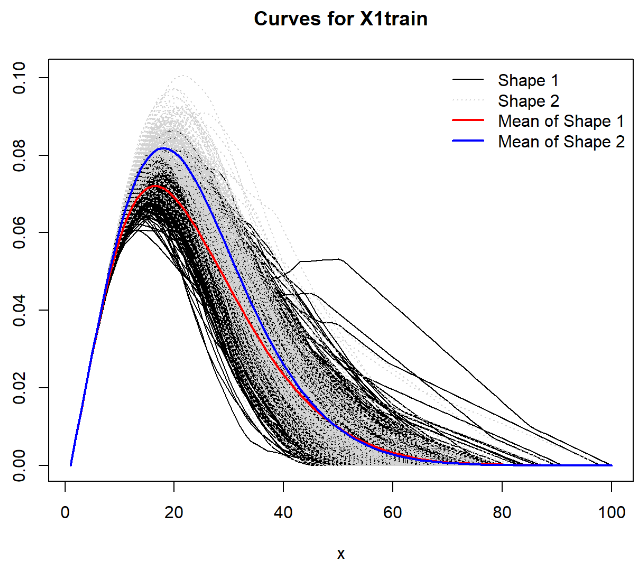
```
## Loading required package: RCurl
```

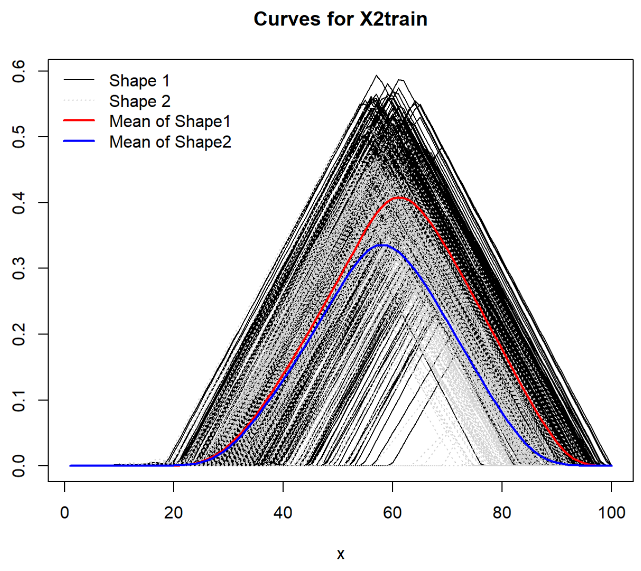
```
## Loading required package: deSolve
```

```
##
## Attaching package: 'fda'
```

```
## The following object is masked from 'package:graphics':
##
##     matplot
```

```r
library(refund)
# Read the training data
# Read the training data
path_Ytrain <- "C:/Users/fwjbi/OneDrive - Bowling Green State University/Summer 2023/6820-Functi
on/Chapter 6/data/P2Ytrain.csv"
path_X1train <- "C:/Users/fwjbi/OneDrive - Bowling Green State University/Summer 2023/6820-Funct
ion/Chapter 6/data/P2X1train.csv"
path_X2train <- "C:/Users/fwjbi/OneDrive - Bowling Green State University/Summer 2023/6820-Funct
ion/Chapter 6/data/P2X2train.csv"
path_X1test <- "C:/Users/fwjbi/OneDrive - Bowling Green State University/Summer 2023/6820-Functi
on/Chapter 6/data/P2X1test.csv"
path_Ytest <- "C:/Users/fwjbi/OneDrive - Bowling Green State University/Summer 2023/6820-Functio
n/Chapter 6/data/P2Ytest.csv"
path_X2test <- "C:/Users/fwjbi/OneDrive - Bowling Green State University/Summer 2023/6820-Functi
on/Chapter 6/data/P2X2test.csv"

Ytrain <- read.csv(path_Ytrain)
X1train <- read.csv(path_X1train)
X2train <- read.csv(path_X2train)
X1test <- read.csv(path_X1test)
X2test <- read.csv(path_X2test)
Ytest <- read.csv(path_Ytest)

colors <- ifelse(Ytrain$x == 0, "black", "lightgray")
line_types <- ifelse(Ytrain$x == 0, 1, 3)

means1 <- sapply(X1train[Ytrain$x == 0, ], mean)
means2 <- sapply(X1train[Ytrain$x == 1, ], mean)
means3 <- sapply(X2train[Ytrain$x == 0, ], mean)
means4 <- sapply(X2train[Ytrain$x == 1, ], mean)

par(mfrow = c(1, 2))
# First plot
matplot(t(X1train), type = 'l', col = colors, lty = line_types, main = "Curves for X1train")
lines(means1, col = "red", lwd = 2)
lines(means2, col = "blue", lwd = 2)
legend("topright", legend = c("Shape 1", "Shape 2", "Mean of Shape 1", "Mean of Shape 2"),
       col = c("black", "lightgray", "red", "blue"), lty = c(1, 3, 1, 1), lwd = c(1, 1, 2, 2), b
ty = "n")

# Second plot
matplot(t(X2train), type = 'l', col = colors, lty = line_types, main = "Curves for X2train")
lines(means3, col = "red", lwd = 2)
lines(means4, col = "blue", lwd = 2)
legend("topleft", legend = c("Shape 1", "Shape 2", "Mean of Shape1", "Mean of Shape2"),
       col = c("black", "lightgray", "red", "blue"), lty = c(1, 3, 1, 1), lwd = c(1, 1, 2, 2), b
ty = "n")
```





```
par(mfrow = c(1, 1))
```

The X2train curves, is more informative for distinguishing between the two shapes. In the left plot, which shows the X1train curves, the two shapes overlap significantly, making it more challenging to differentiate between them. In contrast, the right plot of X2train offers a better distinction, with a y-axis range from 0 to 0.6. Compared to the 0 to 0.1 range in the left plot, the wider range in the right plot might contribute to better separation and less overlap.

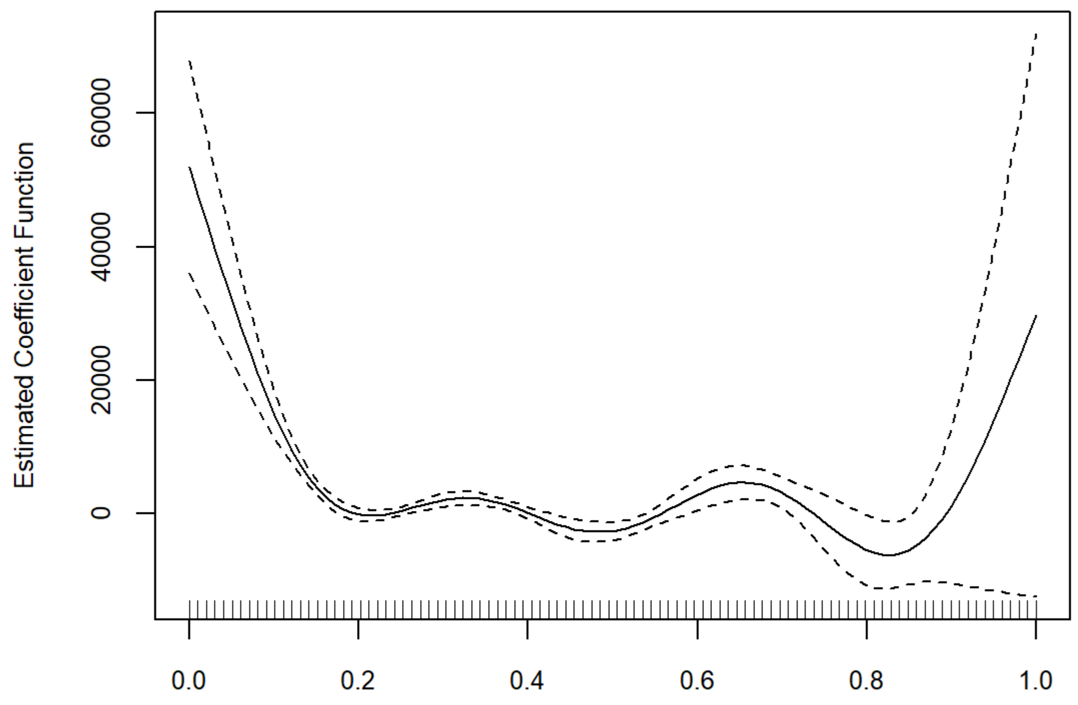
b. Using the `pfr()` (with `lf()` inside of it) function fit a regression model with $X_{1i}(t)$ as the input functional predictor. Use the *logit* as your link function. Let the number of basis functions, k, vary from 4 to 7 and choose its optimal value based on the AIC score. Set `fx=TRUE` inside `lf()` and use the default settings for the other arguments. Your final model is the model fit with the optimal value of k.

c. Plot the graph of the estimated coefficient function $\hat{\beta}(t)$.

```
X = as.matrix(X1train)
X2 = as.matrix(X2train)
Xtest = as.matrix(X1test)
Xtest2 = as.matrix(X2test)


aic_all1 = numeric(0)
k_all1 = 4:7
for(k in k_all1){
  fit1 <- pfr(Ytrain ~ lf(X, k = k, fx = TRUE), family = binomial(link = "logit"))
  tmp1 = extractAIC(fit1)[2]
  aic_all1 = c(aic_all1,tmp1)
}


k_opt1 <- k_all1[which.min(aic_all1)]
k_opt1
```

```
## [1] 7
```

```
fit.f1 <- pfr(Ytrain ~ lf(X, k = k_opt1, fx = TRUE), family = binomial(link = "logit"))
# summary(fit.f1)
# str(fit.f1)
plot(fit.f1, xlab="", ylab="Estimated Coefficient Function", cex.lab=0.8, cex.axis=0.8)
```



d. Use the `predict()` function (with type = 'response' inside of it) to compute $P(Y_i = 1)$ (probability that a given curve corresponds to shape 2) on the test set. Convert these probabilities into binary outcome of 0 (shape 1) and 1 (shape 2) using the 0.5 probability threshold. Finally, compute and report the *accuracy* of your final model on the test set.

```
predictions1 <- predict(fit.f1, newdata = list(X = Xtest), type = "response")
pred1 <- ifelse(predictions1 > 0.5, 1, 0)

dfp1 <- data.frame(Preds1 = pred1, Obs1 = Ytest)
dfp1$Correct <- dfp1$Preds1 == dfp1$x
Accuracy1 <-sum(dfp1$Correct)/length(dfp1$Correct)
Accuracy1
```

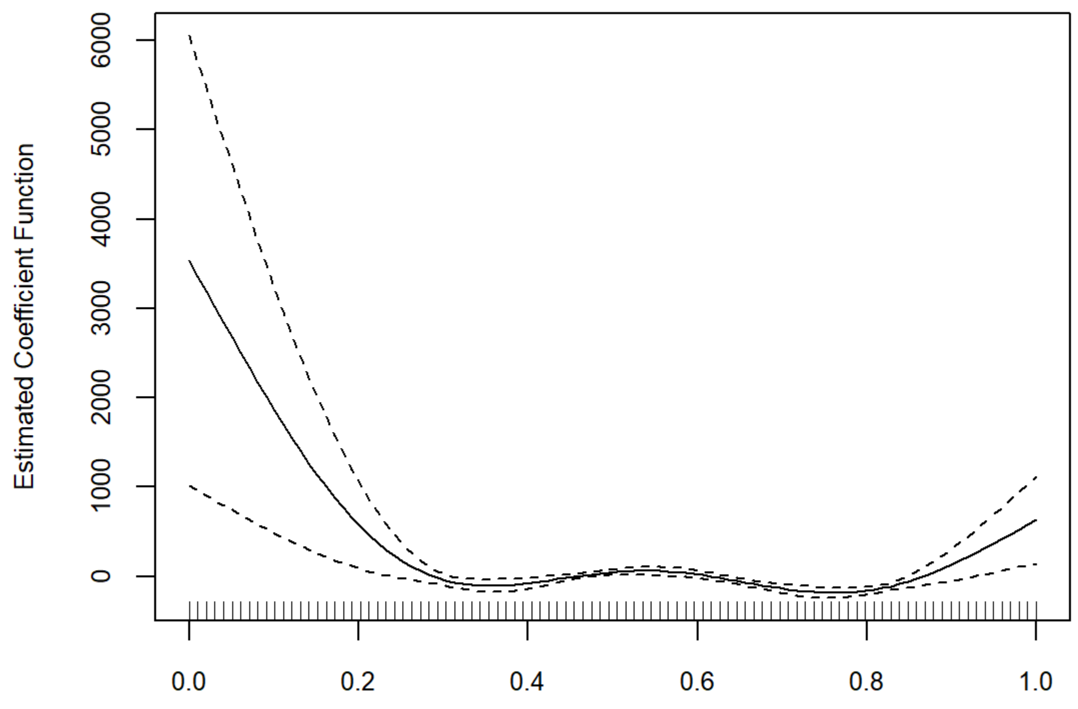
```
## [1] 0.546
```

e. Repeat (b)-(d) for $X_{2i}(t)$.

```
aic_all2 = numeric(0)
k_all2 = 4:7
for(k in k_all1){
  fit2 <- pfr(Ytrain ~ lf(X2, k = k, fx = TRUE), family = binomial(link = "logit"))
  tmp2 = extractAIC(fit2)[2]
  aic_all2 = c(aic_all2,tmp2)
}


k_opt2 <- k_all2[which.min(aic_all2)]
k_opt2
```

```
## [1] 5
```

```
fit.f2 <- pfr(Ytrain ~ lf(X2, k = k_opt2, fx = TRUE), family = binomial(link = "logit"))
# summary(fit.f2)
plot(fit.f2, xlab="", ylab="Estimated Coefficient Function",cex.lab=0.8, cex.axis=0.8)
```



```
predictions2 <- predict(fit.f2, newdata = list(X2 = Xtest2), type = "response")
pred2 <- ifelse(predictions2 > 0.5, 1, 0)

dfp2 <- data.frame(Preds2 = pred2, Obs2 = Ytest)
dfp2$Correct <- dfp2$Preds2 == dfp2$x
Accuracy2 <-sum(dfp2$Correct)/length(dfp2$Correct)
Accuracy2
```

```
## [1] 0.826
```

f. Which of the two models provides the lowest test accuracy?

Model 1 has an accuracy rate of 0.546, whereas Model 2 has a higher accuracy rate of 0.826. Therefore, Model 1 provides the lower test accuracy of the two.