Homework#6

Arinjay Jain

```r
library(plgp)
```

```
## Warning: package 'plgp' was built under R version 3.6.3

## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 3.6.3

## Loading required package: tgp

## Warning: package 'tgp' was built under R version 3.6.3
```

Part A

Relevant data quantities.

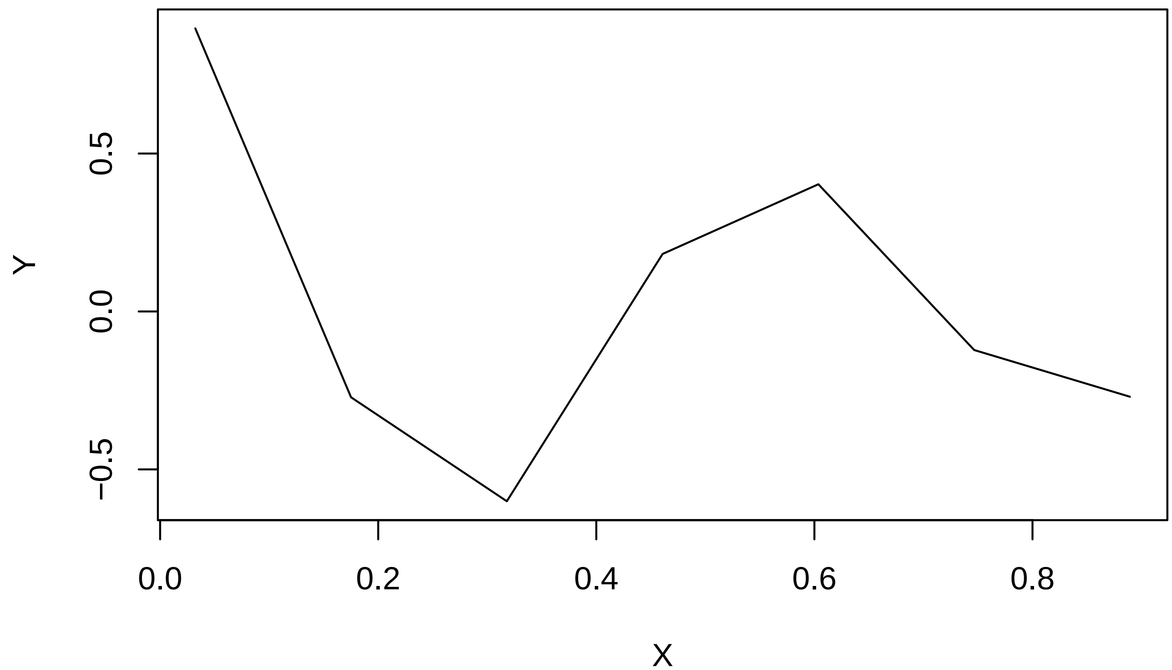
```r
x1=runif(1,min = 0, max = 1/7)
x2=x1+1/7
x3=x2+1/7
x4=x3+1/7
x5=x4+1/7
x6=x5+1/7
x7=x6+1/7

X=matrix(c(x1,x2,x3,x4,x5,x6,x7),ncol=1)

Y <- exp(-1.4*X)*cos(3.5*pi*X)
D <- distance(X)

eps <- sqrt(.Machine$double.eps)
Sigma <- exp(-50*D)

plot(X, Y, type="l")
```



Relevant predictive quantities.

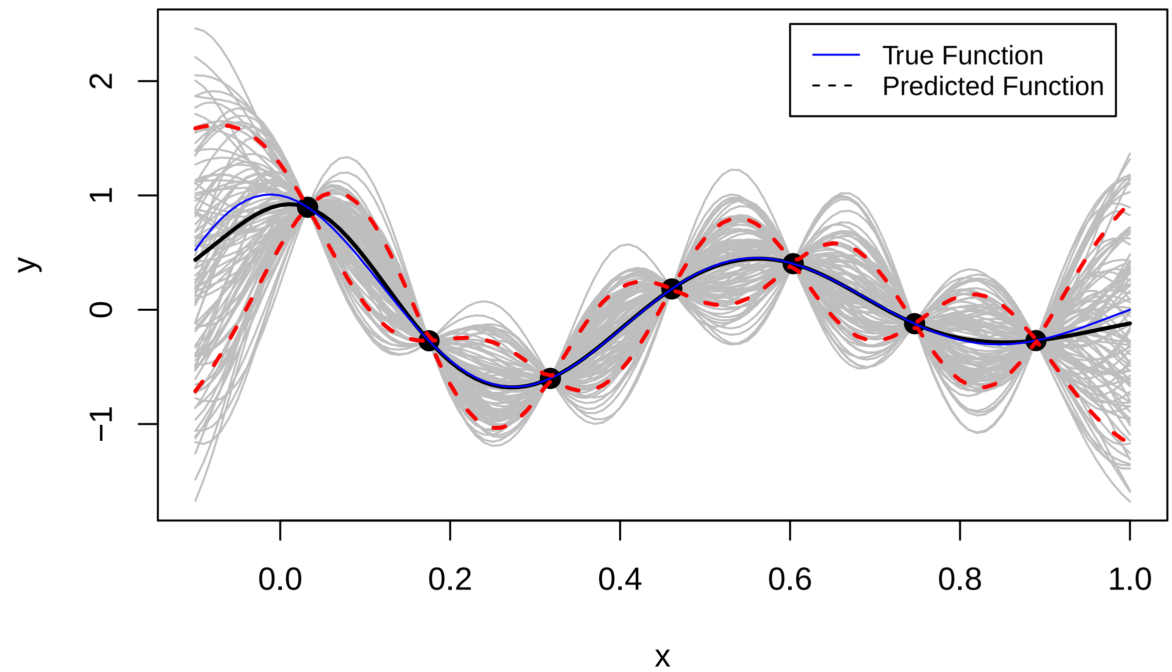
```
XX <- matrix(seq(-0.1,1,by = 1/100), ncol=1)
DXX <- distance(XX)
SXX <- exp(-50*DXX)+ diag(eps, ncol(DXX))
DX <- distance(XX, X)
SX <- exp(-50*DX)
```

Predictive equations in R

```
Si <- solve(Sigma)
mup <- SX %*% Si %*% Y
Sigmap <- SXX - SX %*% Si %*% t(SX)
```

```
YY <- rmvnorm(100, mup, Sigmap)
q1 <- mup + qnorm(0.10, 0, sqrt(diag(Sigmap)))
q2 <- mup + qnorm(0.90, 0, sqrt(diag(Sigmap)))
```

```
matplot(XX, t(YY), type="l", col="gray", lty=1, xlab="x", ylab="y")
points(X, Y, pch=20, cex=2)
lines(XX, mup, lwd=2);
lines(XX, exp(-1.4*XX)*cos(3.5*pi*XX), col="blue")
lines(XX, q1, lwd=2, lty=2, col=2);
lines(XX, q2, lwd=2, lty=2, col=2)
legend(0.6, 2.5, legend=c("True Function", "Predicted Function"),col=c("blue", "black"), lty=1:2, cex=0
```



Part B

```
counter <- 0
nlg <- function(g, D, Y)
  {
    n <- length(Y)
    K <- exp(-50*D) + diag(g, n)
    Ki <- solve(K)
    ldetK <- determinant(K, logarithm=TRUE)$modulus
    ll <- - (n/2) * log(t(Y) %*% Ki %*% Y) - (1/2) * ldetK
    counter <<- counter + 1
    return(-ll)
}


gnlg <- function(g, D, Y)
 {
    n <- length(Y)
    K <- exp(-50*D) + diag(g, n)
    Ki <- solve(K)
    KiY <- Ki %*% Y
    dll <- (n/2) * t(KiY) %*% KiY / (t(Y) %*% KiY) - (1/2) * sum(diag(Ki))
    return(-dll)
```

```
}
```

```
X <- rbind(X, X, X)
n <- nrow(X)
D <- distance(X)
y <- exp(-1.4*X)*cos(3.5*pi*X) + rnorm(n,mean = 0, sd = 1)
```

Optim function

```
g <- optimize(nlg, interval=c(eps, var(y)), D=D, Y=y)$minimum
g
```

```
## [1] 1.026447
```

```
K <- exp(-50*D) + diag(g, n)
Ki <- solve(K)
tau2hat <- drop(t(y) %*% Ki %*% y / n)
c(tau=sqrt(tau2hat), sigma=sqrt(tau2hat*g))
```

```
##       tau     sigma
## 0.8559679 0.8672129
```

Query points

```
XX <- matrix(seq(-0.1,1,by = 1/100), ncol=1)
```

prediction using the estimated hyperparameters

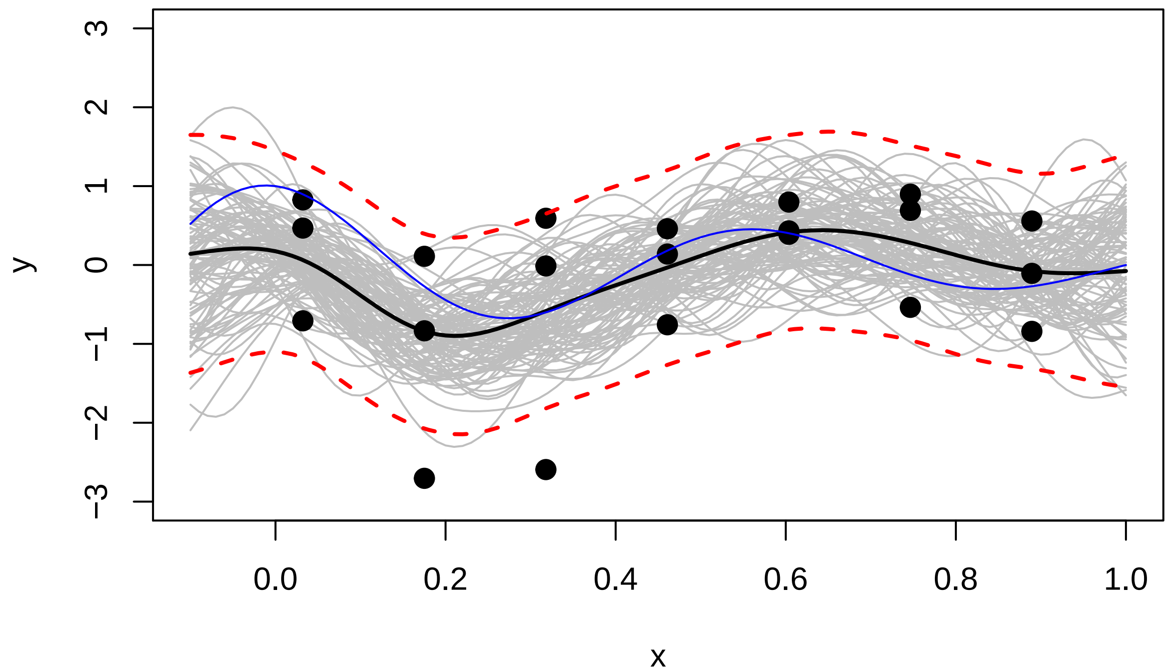
```
DX <- distance(XX, X);
KX <- exp(-50*DX)
KXX <- exp(-50*DXX) + diag(g, nrow(DXX))
```

Derive the predictive mean vector and covariance matrix.

```
mup <- KX %*% Ki %*% y
Sigmap <- tau2hat * (KXX - KX %*% Ki %*% t(KX))
q1 <- mup + qnorm(0.10, 0, sqrt(diag(Sigmap)))
q2 <- mup + qnorm(0.90, 0, sqrt(diag(Sigmap)))
```

```
Sigma.int <- tau2hat * (exp(-50*DXX) + diag(eps, nrow(DXX)) - KX %*% Ki %*% t(KX))
YY <- rmvnorm(100, mup, Sigma.int)

matplot(XX, t(YY), type="l", lty=1, col="gray", xlab="x", ylab="y",ylim=c(-3,3))
points(X, y, pch=20, cex=2)
lines(XX, mup, lwd=2);
lines(XX, exp(-1.4*XX)*cos(3.5*pi*XX), col="blue")
lines(XX, q1, lwd=2, lty=2, col=2);
lines(XX, q2, lwd=2, lty=2, col=2)
```




```r
cat("Blue line is True Function, Black Points are Interpolation, and Black Line is Predicted Function")
```

```
## Blue line is True Function, Black Points are Interpolation, and Black Line is Predicted Function
```