Advanced Machine Learning, Lab 3

Rebin Hosini (rebho 150)

2017-09-27

Question 1

a

```
#Init
nSim <- 10
sigmaF <- 1
l <- 0.3
hyperParam <- c(sigmaF,l)

#data
x <- c(-1.0, -0.6,-0.2,0.4,0.8)
xStar <- seq(-1,1,0.01)
y <- c(0.768,-0.044,-0.940,0.719,-0.664)


#Define kernal
kernel <- function(x1,x2,sigmaF=1,l=3){
  n1 <- length(x1)
  n2 <- length(x2)
  K <- matrix(NA,n1,n2)
  for (i in 1:n2){
    K[,i] <- sigmaF^2*exp(-0.5*( (x1-x2[i])/l)^2 )
  }
  return(K)
}


posteriorGP <- function(x, y, xStar, hyperParam, sigmaNoise){

  k <- kernel(x1 = x, x2 = x,
              sigmaF = hyperParam[1],
              l = hyperParam[2])
  kStar <- kernel(x,xStar,
              sigmaF = hyperParam[1],
              l = hyperParam[2])
  kStars <- kernel(xStar,xStar,
              sigmaF = hyperParam[1],
              l = hyperParam[2])

  L <- chol(k + sigmaNoise*diag(length(x)))
  alpha <- solve(L,(solve(t(L),y)))
  fStar <- t(kStar)%*%alpha #Predictive mean
  v <- solve(t(L),kStar)
  vfstar <- kStars-t(v)%*%v #Predictive variance
```

```
  return(list("Variance" = vfstar, "Mean" = fStar))

}


CI <- function(res, band = 1.96){
  upper <- res$Mean + band*sqrt(diag(res$Variance))
  lower <- res$Mean - band*sqrt(diag(res$Variance))
  return(list("upper" = upper, "lower" = lower))
}
```

b

```
xb <- 0.4
yb <- 0.719

resb <- posteriorGP(x = xb,y = yb, xStar,hyperParam = c(1,0.3), sigmaNoise = 0.1)


plot(y = resb$Mean, x = xStar, type = "l", ylim = c(-5,5),
     ylab = "f(x)", xlab = "x")
lines(y = CI(resb)$upper, x = xStar, col = "red", type = "l")
lines(y = CI(resb)$lower, x = xStar, col = "red", type = "l")
points(y = yb, x = xb)
```

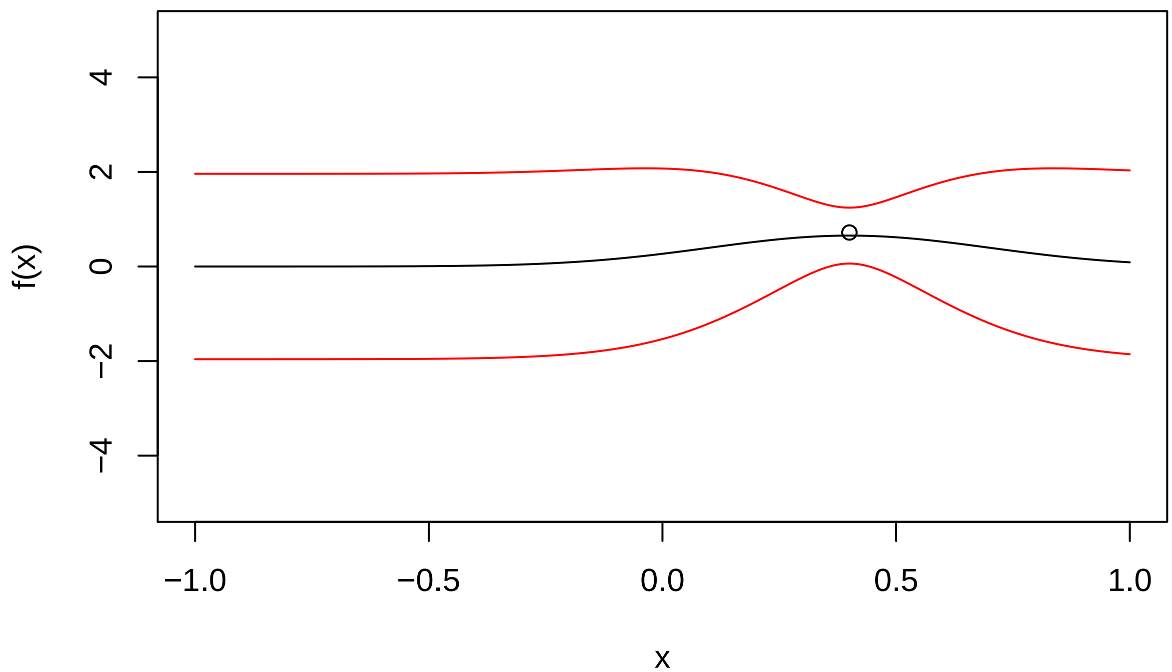


Figure 1: Posterior mean and variance with only one observation

c

```
xc <- c(xb, -0.6)
yc <- c(yb, -0.0444)

resc <- posteriorGP(x = xc,y = yc, xStar,hyperParam = c(1,0.3), sigmaNoise = 0.1)

plot(y = resc$Mean, x = xStar, type = "l", ylim = c(-5,5),
     ylab = "f(x)", xlab = "x")
lines(y = CI(resc)$upper, x = xStar, col = "red", type = "l")
lines(y = CI(resc)$lower, x = xStar, col = "red", type = "l")
points(y = yc, x = xc)
```

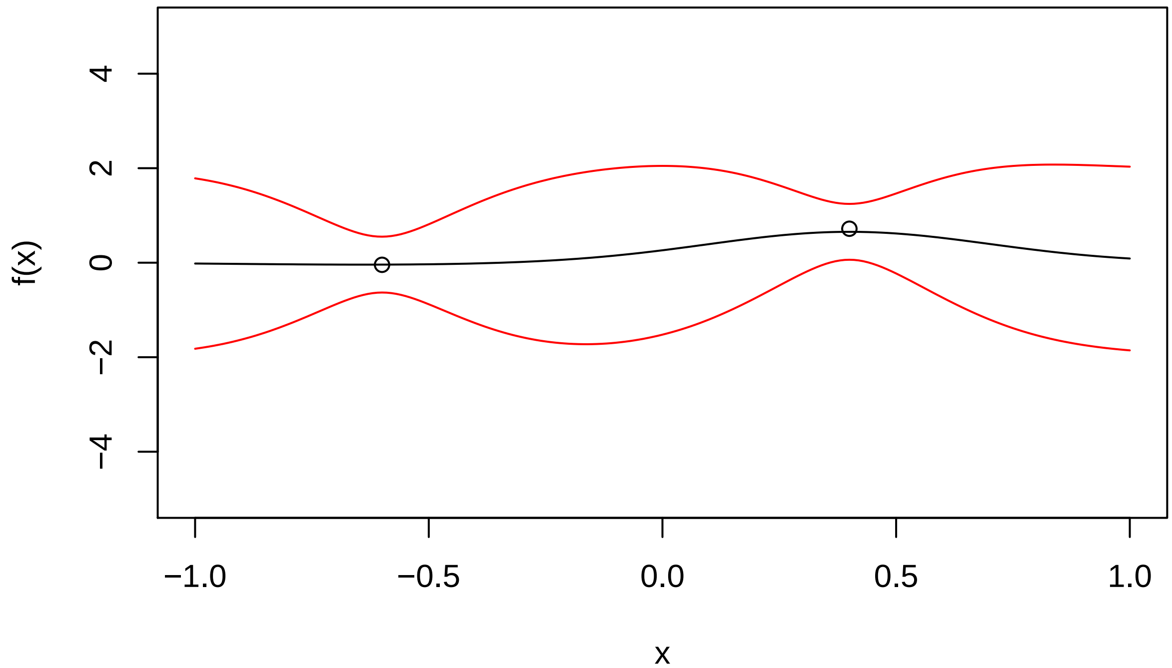


Figure 2: Posterior mean and variance when adding an observation

d

```
resd <- posteriorGP(x,y,xStar,hyperParam = hyperParam, sigmaNoise = 0.1)
plot(y = resd$Mean, x = xStar, type = "l", ylim = c(-5,5),
     ylab = "f(x)", xlab = "x")
lines(y = CI(resd)$upper, x = xStar, col = "red", type = "l")
lines(y = CI(resd)$lower, x = xStar, col = "red", type = "l")
points(y = y, x = x)
```

```
hyperParamD <- c(1,1)
rese <- posteriorGP(x,y,xStar,hyperParam = hyperParamD, sigmaNoise = 0.1)
plot(y = rese$Mean, x = xStar, type = "l", ylim = c(-5,5),
     ylab = "f(x)", xlab = "x")
lines(y = CI(rese)$upper, x = xStar, col = "red", type = "l")
lines(y = CI(rese)$lower, x = xStar, col = "red", type = "l")
```

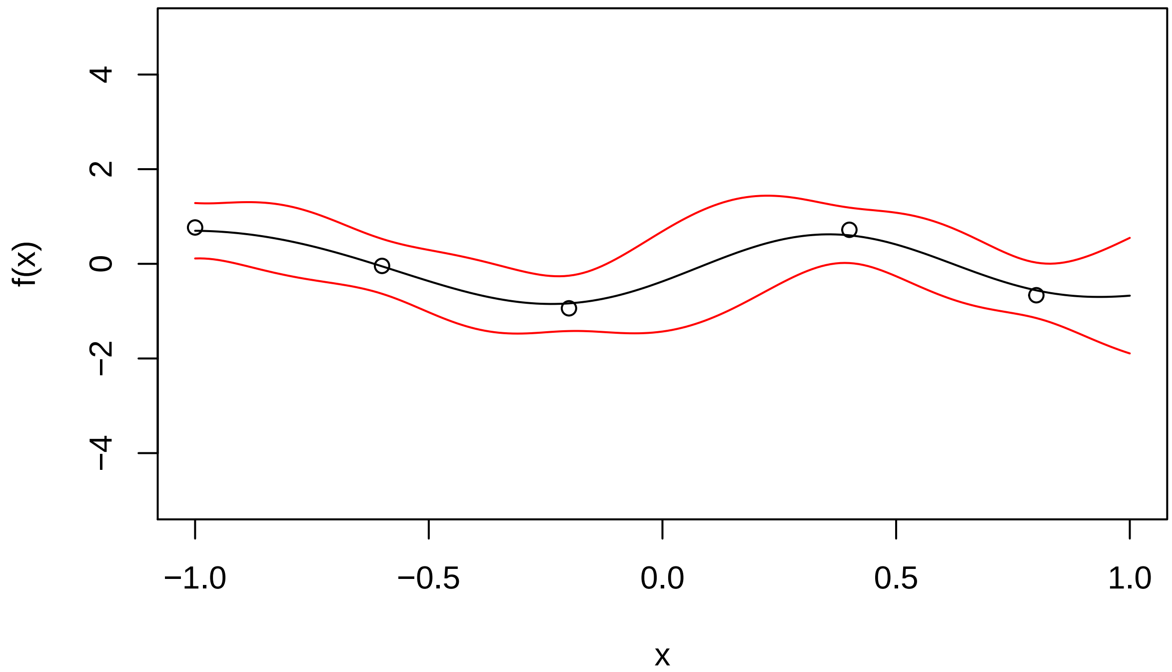


Figure 3: Posterior mean and variance

```
points(y = y, x = x)
```

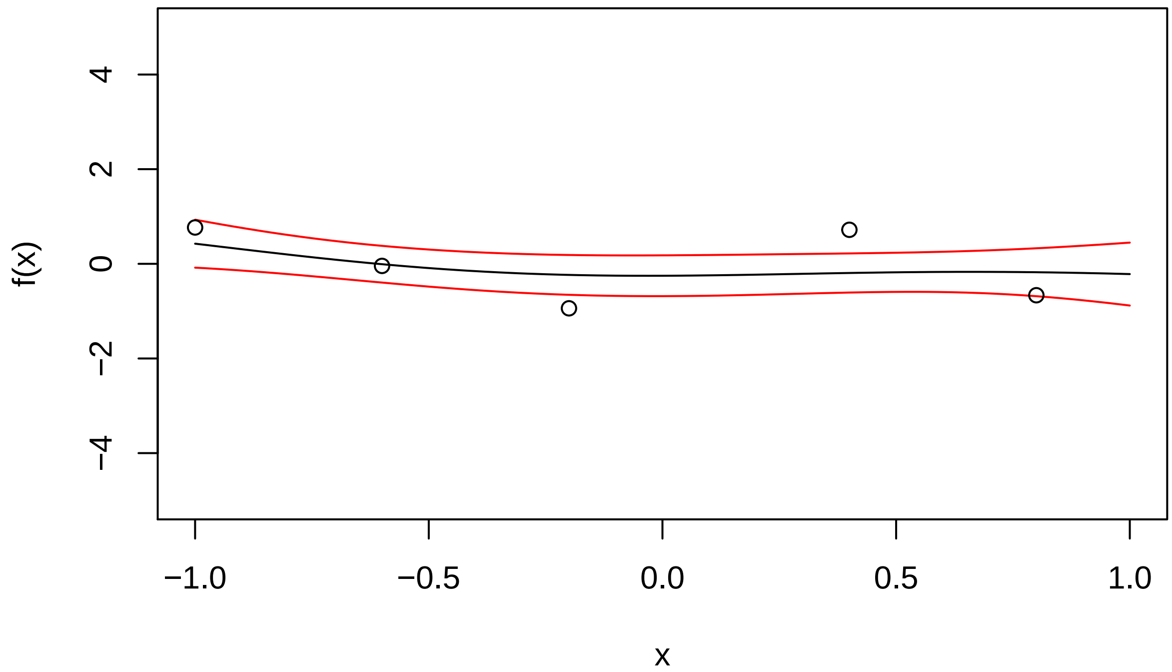


Figure 4: Posterior mean and variance with updated hyperparameters

Question 2

Preprocessing

```
#Libraries
library(kernlab)

#Data
temp <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullinge.c

temp$time <- c(1:nrow(temp))

count <- 1

#Create day variable
tempdf <- function(temp){
  count <- 1
  for (i in 1:length(temp[,"time"])){
    if (count == 366){
      count <- 1
      temp$day[i] <- count
      count <- count + 1
    } else {
      temp$day[i] <- count
      count <- count + 1
    }
  }
  return(temp)
```

```
}

#Updated dataframe
temp <- tempdf(temp)
index <- c()

for (i in 1:nrow(temp)){
  remainder <- i %% 5
  if (remainder == 1){
    index[i] <- i
  }
}

subTemp <- temp[na.omit(index),]
```

a

```
#Define Kernal
SEkernal <- function(l, sigmaf){

  SEK <- function(x,y = NULL){
    r <- x-y
    res <- sigmaf^2*exp(-r^2/(2*l^2))
    return(res)
  }
  class(SEK) <- "kernel"
  return(SEK)
}

#Init data
l <- 1
sigmaf <- 1

x <- c(1,3,4)
xStar <- c(2,3,4)

kernel1 <- SEkernal(l = l, sigmaf = sigmaf)
kernelMatrix(kernel = kernel1, x, xStar)
```

```
## An object of class "kernelMatrix"
##           [,1]      [,2]      [,3]
## [1,] 0.6065307 0.1353353 0.0111090
## [2,] 0.6065307 1.0000000 0.6065307
## [3,] 0.1353353 0.6065307 1.0000000
```

b

```
gaussPred <- function(sigmaf = 20, l = 0.2){
  sigma <- sd(lm(temp ~ poly(time, 2), data = subTemp)$residuals)
  gaussianFit <- gausspr(temp ~ time, kpar = list(sigmaf = sigmaf, l = l),
                         data = subTemp, kernel = SEkernal, var = sigma^2)
```

```
  gpred <- predict(gaussianFit, subTemp)
  return(gpred)
}

sigmaflist <- c(20, 30, 40)
llist <- c(0.2, 1, 5)
xGrid <- seq(1, length(subTemp$time))

plot(y = gaussPred(sigmaflist[1], llist[1]), x = xGrid, type = "l",
     col = "red", main = "Gaussian prediction",
     xlab = "Time", ylab = "Temp", ylim = c(-15,35))
lines(y = gaussPred(sigmaflist[2], llist[2]), x = xGrid, type = "l",
     col = "green", main = "Gaussian prediction",
     xlab = "Time", ylab = "Temp")
lines(y = gaussPred(sigmaflist[3], llist[3]), x = xGrid, type = "l",
     col = "blue", main = "Gaussian prediction",
     xlab = "Time", ylab = "Temp")
points(y = subTemp$temp, x = xGrid, lwd = 0.5)
legend("topright", legend = c("Sigma = 20, l = 0.2",
                              "Sigma = 30, l = 1",
                              "Sigma = 40, l = 5"),
       col = c("red", "green", "blue"), lty = c(1,1,1))
```

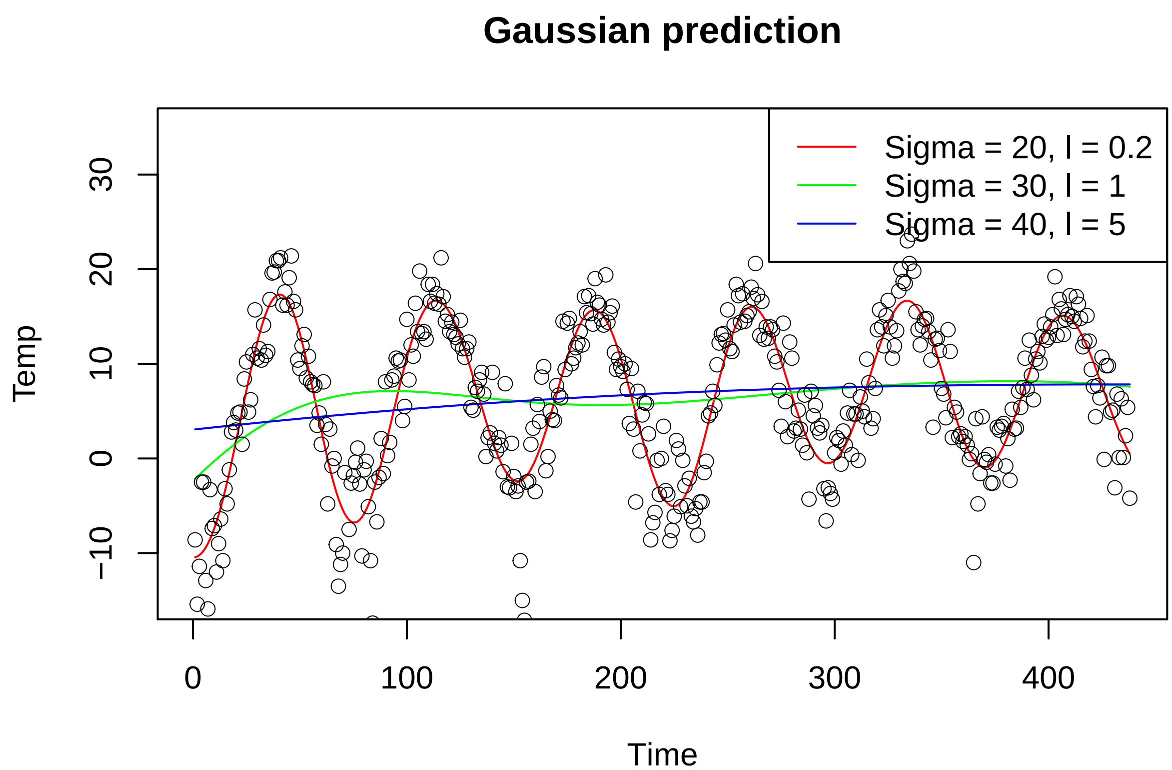


Figure 5: Gaussian prediction with different initial parameters

c

```r
#xGrid <- seq(range(subTemp$time)[1], range(subTemp$time)[2], 1)
xGrid <- seq(1, length(subTemp$time))
postVar <- posteriorGP(x = subTemp$time, y = subTemp$temp,
                       hyperParam = c(1,0.2),
                       sigmaNoise = 8.176288,
                       xStar = xGrid)

lower <- gaussPred(sigmaflist[1], llist[1]) - 1.96*sqrt(diag(postVar$Variance))
upper <- gaussPred(sigmaflist[1], llist[1]) + 1.96*sqrt(diag(postVar$Variance))


plot(y = gaussPred(sigmaflist[1], llist[1]), x = xGrid, type = "l",
     col = "red", main = "Gaussian prediction",
     xlab = "Time", ylab = "Temp", ylim = c(-15,35))
lines(y = gaussPred(sigmaflist[2], llist[2]), x = xGrid, type = "l",
     col = "green", main = "Gaussian prediction",
     xlab = "Time", ylab = "Temp")
lines(y = gaussPred(sigmaflist[3], llist[3]), x = xGrid, type = "l",
     col = "blue", main = "Gaussian prediction",
     xlab = "Time", ylab = "Temp")
lines(y = upper, x = xGrid[1:438], col = "orange", type = "l", lty = 3)
lines(y = lower, x = xGrid[1:438], col = "orange", type = "l", lty = 3)
legend("topright", legend = c("Sigma = 20, l = 0.2",
                              "Sigma = 30, l = 1",
                              "Sigma = 40, l = 5",
                              "Confidence Bands"),
       col = c("red", "green", "blue", "orange"), lty = c(1,1,1,3))
```

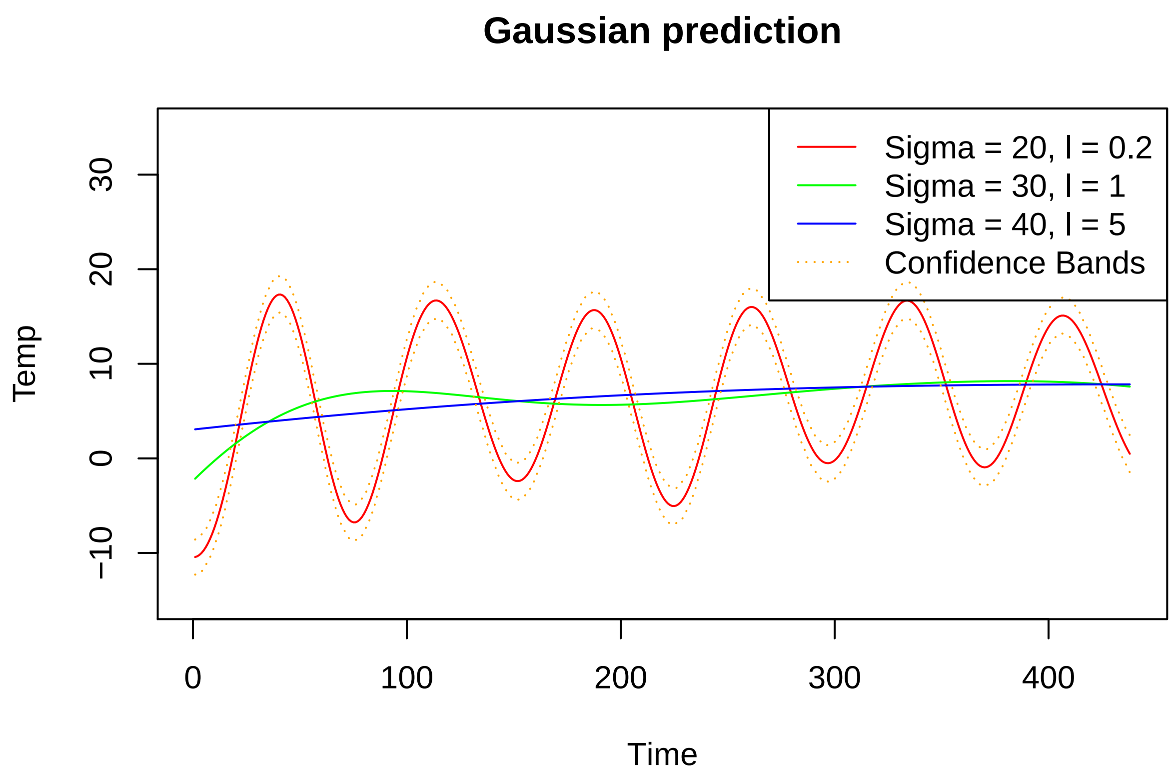


Figure 6: Gaussian prediction from 2b with confidence bands

d

```
sigma <- sd(lm(temp ~ poly(time, 2), data = subTemp)$residuals)

g0.2 <- gausspr(temp ~ time, kpar = list(sigmaf = 20, l = 0.2),
                       data = subTemp, kernel = SEkernal, var = sigma^2)

g1.2 <- gausspr(temp ~ day, kpar = list(sigmaf = 20, l = 1.2),
                       data = subTemp, kernel = SEkernal, var = sigma^2)

pg0.2 <- predict(g0.2, subTemp)
pg1.2 <- predict(g1.2, subTemp)

plot(pg0.2, x = subTemp$time, type = "l", col = "blue",
     ylim = c(-11, 25), ylab = "Temp", xlab  = "Time")
lines(pg1.2, x = subTemp$time, type = "l", col = "red")
legend("topright", col = c("blue", "red"), legend = c("Sigmaf = 30, l = 0.2",
                                                       "Sigmaf = 30, l = 1.2"),
       lty = c(1,1))
```

```
#points(y = temp$temp, x = temp$time, lwd = 0.1)
```

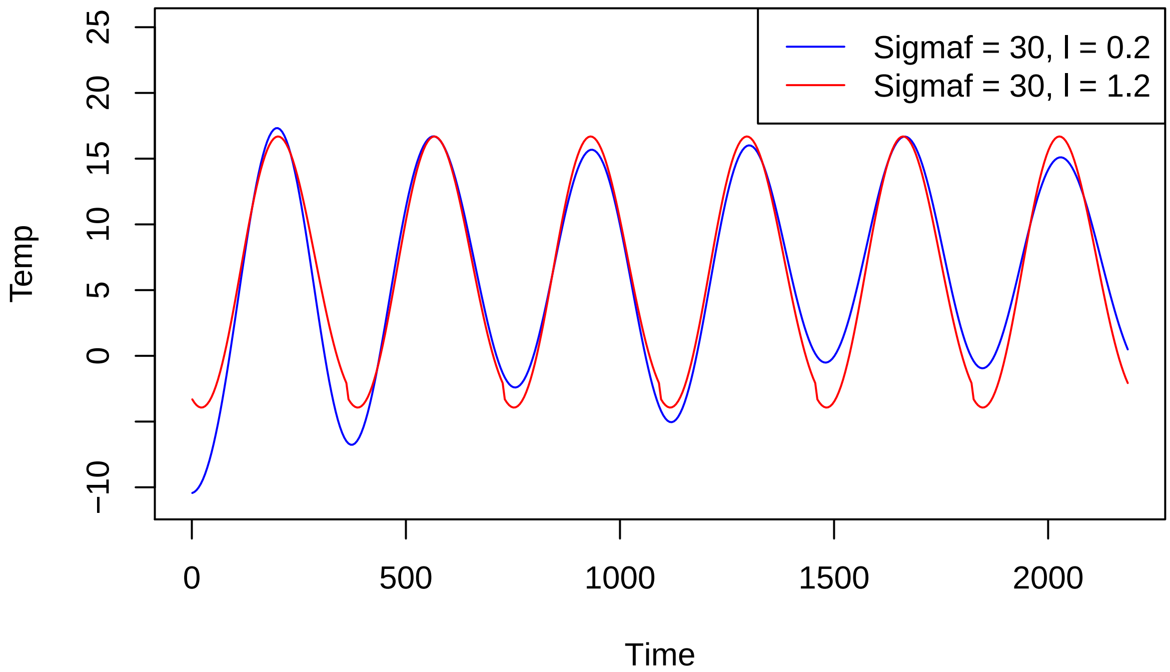
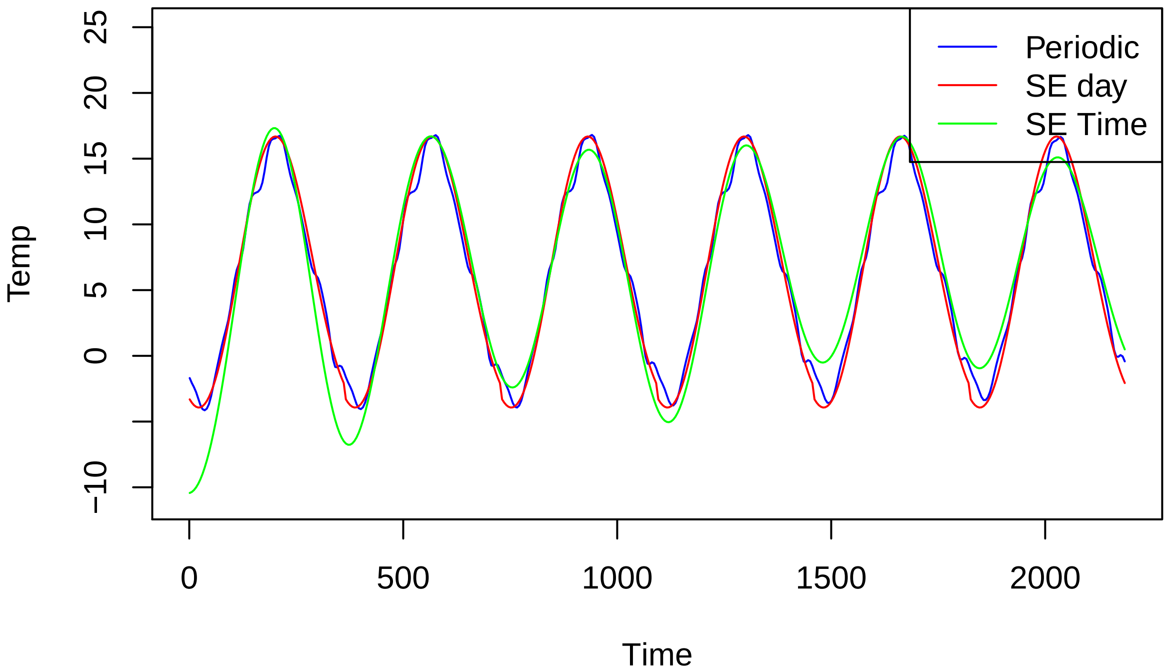


Figure 7: The blue prediction: l = 1.2 with day, Red prediction: l = 0.2 with time

e

```
#Define Kernal
pKernel <- function(l1,l2,d,sigmaf){
  out <- function(x1,x2){
    sigmaf*exp((-2*sin(pi*abs(x1-x2)/d)^2)/l1^2)*
      exp(-0.5*abs(x1-x2)^2/l2^2)
  }
  class(out) <- "kernel"
  return(out)
}

#Run prediction
sigma <- sd(lm(temp ~ poly(time, 2), data = subTemp)$residuals)

gd <- gausspr(temp ~ time, kpar = list(sigmaf = 20, l1 = 0.2, l2 = 10, d = 365/sd(subTemp$time)),
                       data = subTemp, kernel = pKernel, var = sigma^2)

pgd <- predict(gd, subTemp)

plot(pgd, x = subTemp$time, type = "l", col = "blue",
     ylim = c(-11, 25), ylab = "Temp", xlab  = "Time")
lines(pg1.2, x = subTemp$time, type = "l", col = "red")
lines(pg0.2, x = subTemp$time, type = "l", col = "green")
#points(y = temp$temp, x = temp$time, lwd = 0.1)
legend("topright", col = c("blue", "red", "green"), legend = c("Periodic",
                                                          "SE day",
                                                          "SE Time"),
       lty = c(1,1,1))
```


Periodic
SE day
SE Time
Temp
25
20
15
10
5
0
−10
0
500
1000
1500
2000
Time

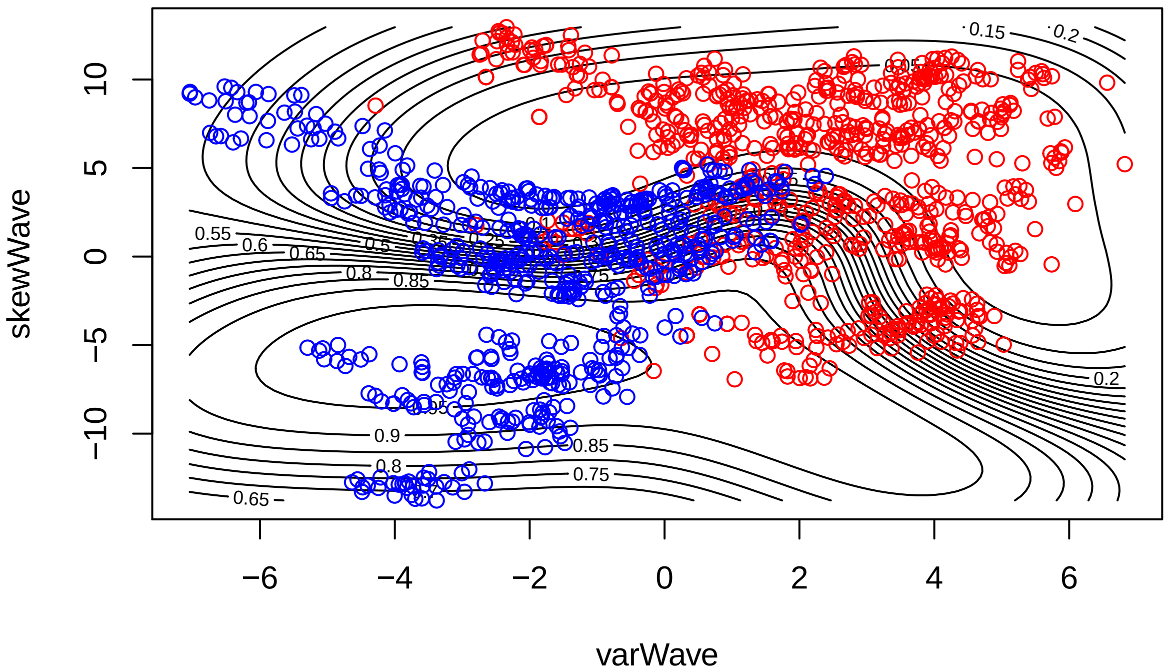
Question 3

a

```
#Libraries
library(AtmRay)

#Import data
data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/banknoteFraud
   header=FALSE, sep=",")
names(data) <- c("varWave","skewWave","kurtWave","entropyWave","fraud")
data[,5] <- as.factor(data[,5])

#Training data
set.seed(111)
SelectTraining <- sample(1:dim(data)[1], size = 1000, replace = FALSE)
train <- data[SelectTraining,]
test <- data[-SelectTraining,]

fit <- gausspr(fraud ~ varWave + skewWave, data = train)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
trainPred <- predict(fit, train[,1:2])

#Grids
x1 <- seq(min(data$varWave),max(data$varWave),length=100)
x2 <- seq(min(data$skewWave),max(data$skewWave),length=100)
gridPoints <- meshgrid(x1, x2)
gridPoints <- cbind(c(gridPoints$x), c(gridPoints$y))
gridPoints <- data.frame(gridPoints)
names(gridPoints) <- c("varWave", "skewWave")
predProbs <- predict(fit, gridPoints, type = "probabilities")


#Plotting for Prob(setosa)
contour(x1,x2,matrix(predProbs[,2],100), 20, xlab = "varWave", ylab = "skewWave")
points(data$varWave[data$fraud==0],data$skewWave[data$fraud==0], col = "red")
points(data$varWave[data$fraud==1],data$skewWave[data$fraud==1], col = "blue")
```



b

```r
#Prediction accuracy with test data
testPred <- predict(fit, test[,1:2])

sum(diag(table(testPred, test$fraud))/sum(table(testPred, test$fraud)))
```

```
## [1] 0.9354839
```

c

```r
fitAll <- gausspr(fraud ~., data = train)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```r
testPredAll <- predict(fitAll, test[1:4])

sum(diag(table(testPredAll, test$fraud))/sum(table(testPredAll, test$fraud)))
```

```
## [1] 0.9973118
```