Advanced Machine Learning Lab4

Kangbo Chen, Hugo Axandersson, Wuhao Wang

2022/10/7

Contribution

All exercises were completed individually and approaches/results were discussed within the group.

Question 1

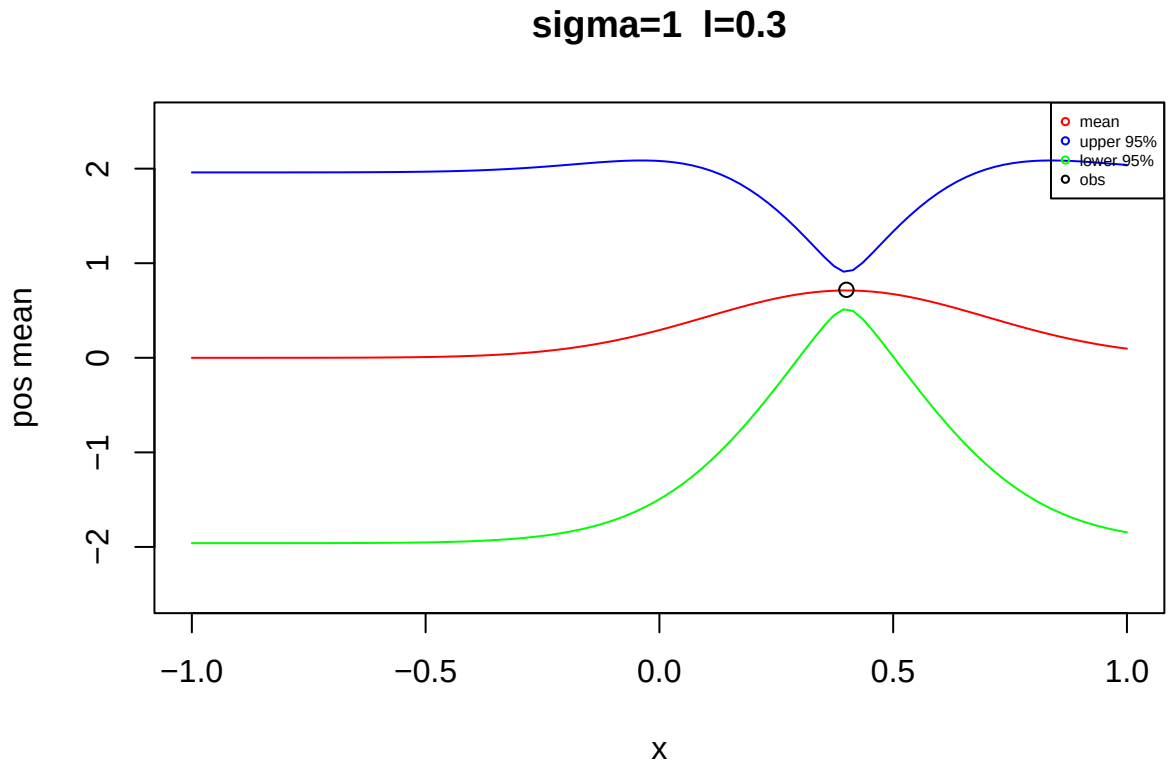
(1) the algorithm was implemented with the following code:

```
# Kernel function from course code
SquaredExpKernel <- function(x1,x2,sigmaF=1,l=0.3){
  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,sigmaNoise,k,...)
{
  n = length(X)
  K = k(X,X,...)
  Kstar  =k(X,Xstar,...)
  # GP_L2 P11 LINE:2
  L = t(chol(K+(sigmaNoise**2)*diag(n)))
  alpha = solve(t(L),solve(L,y))
  # GP_L2 P11 LINE:4
  f = t(Kstar) %*% alpha
  v = solve(L,Kstar,...)
  # GP_L2 P11 LINE:6
  vf = k(Xstar,Xstar)-t(v)%*%v
  res = list('f_mean'=f,'f_variance'=vf)
}
```

(2) Update prior with one observation

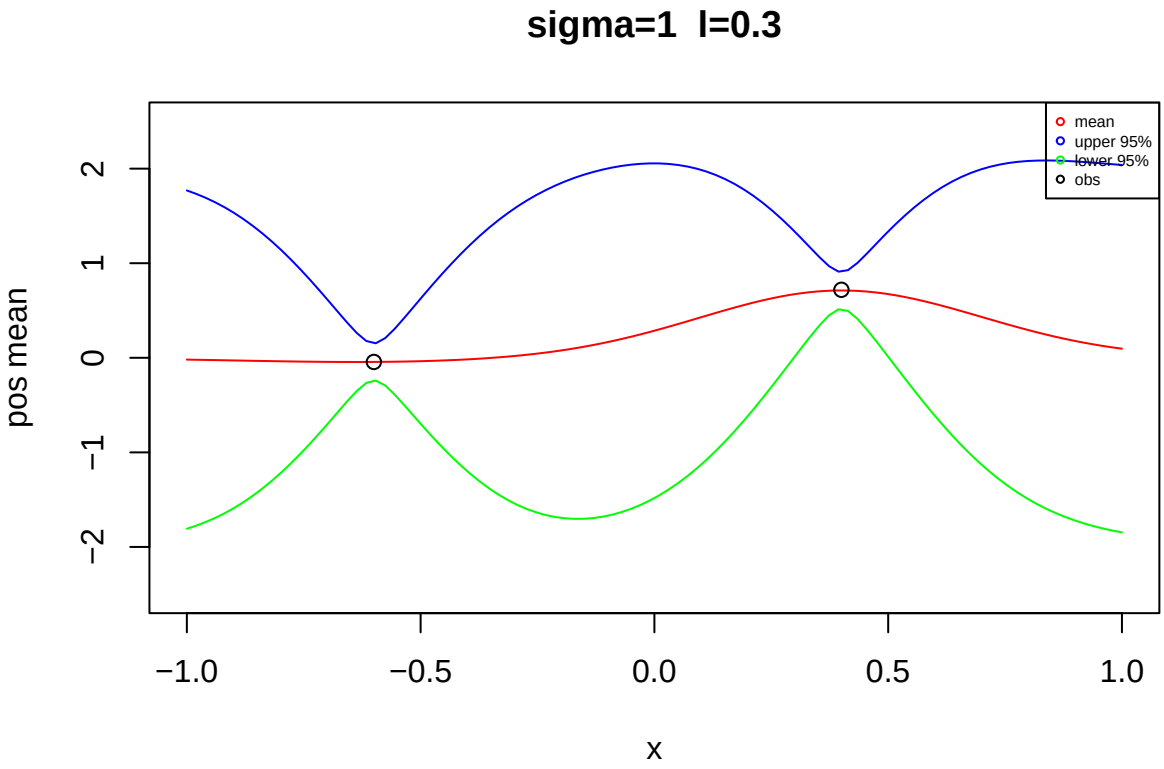
The following figure shows the mean (red) and 95% probability interval (blue and green) for a single observation (black).



As we can see from the plot the 95% probability interval shrunk close to the point and the mean strayed from 0 towards the point. This makes sense since we are more certain of how the function looks around our datapoint.

(3) Update prior with two observations

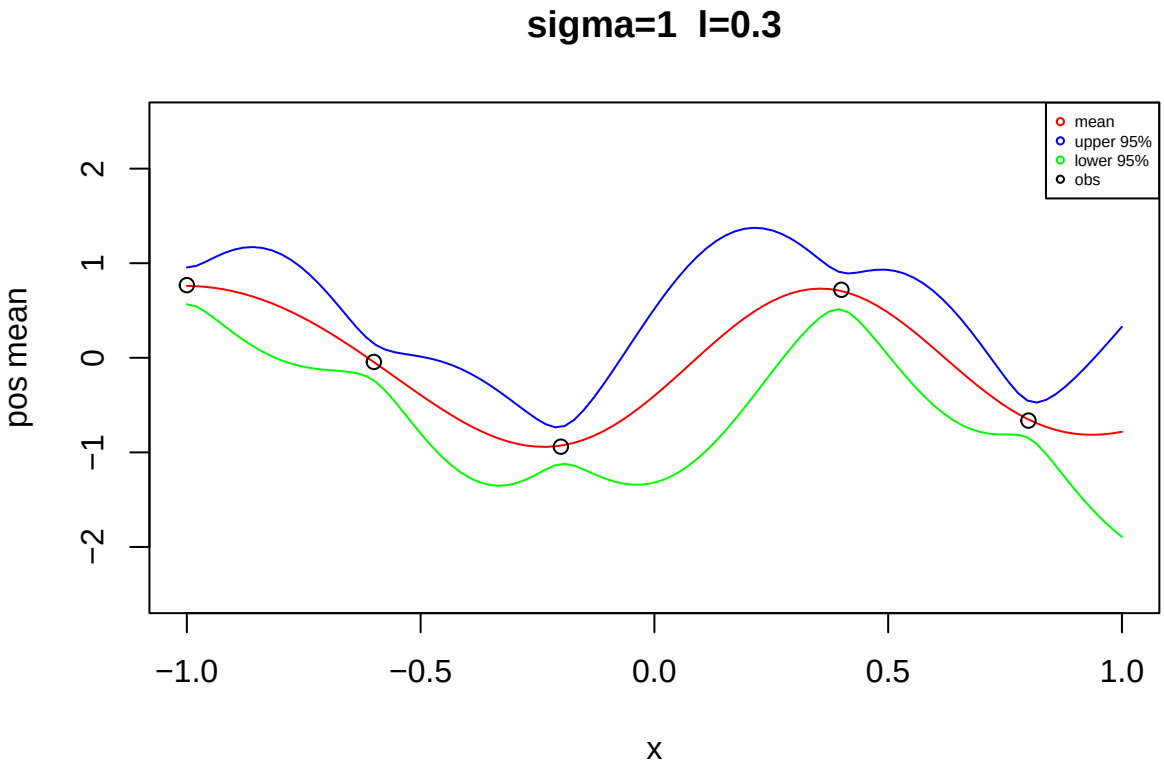
The previous exercise was repeated with one more observation. The result is in the following plot.



As we can see the same behavior the other observation also shrinks the 95% probability interval and moves the mean.

(4) Update prior with four observations

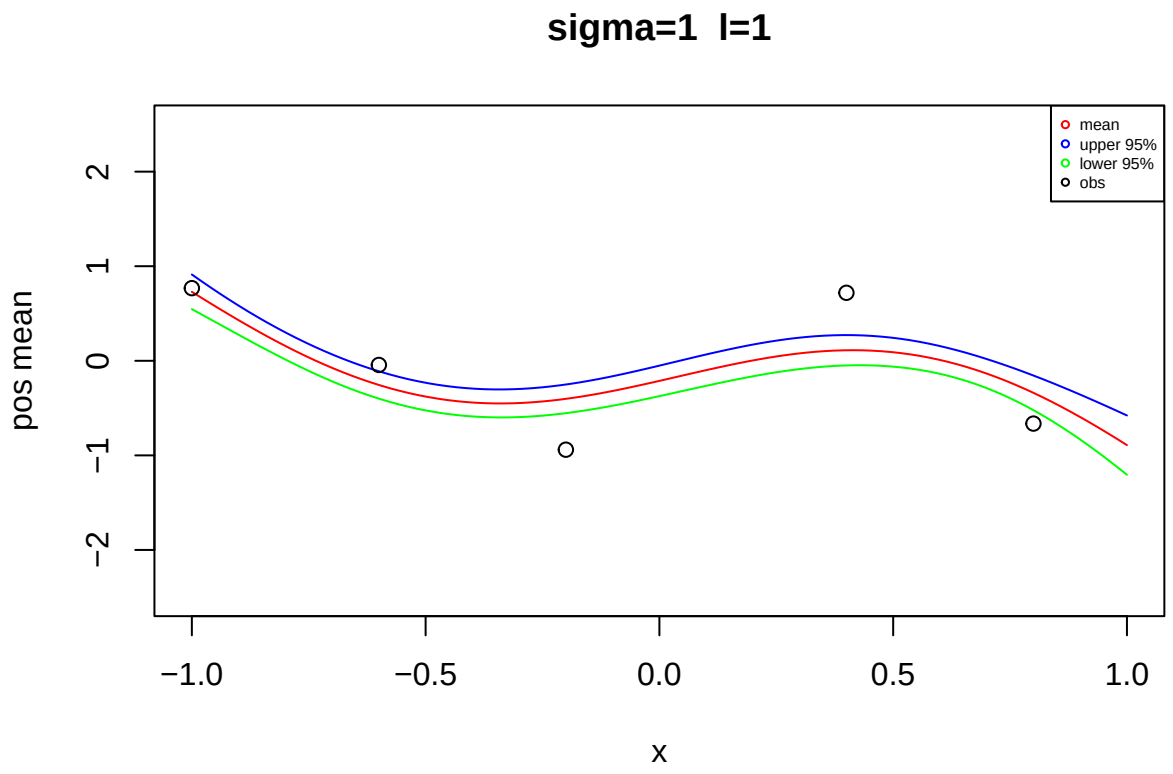
The previous exercises was repeated with four observations. The result is in the following plot.



As we can see we get the same behavior for all observations where the 95% probability interval shrinks and the mean moves. Now the 95% probability interval is fairly small (when compared to having 1 observation) all throughout the function domain.

(5) Update prior with four observations and high l

The previous exercise was repeated with the ℓ value raised from $\ell = 0.3$ to $\ell = 1$. The result is in the following plot.



Comparision: The ℓ value punishes complexity so by raising it we can see from the plot the function is less complex with higher probability. This can be seen as the mean doesn't reach the datapoints as it did before and the 95% probability interval is way smaller.

Question 2

(1) Define square exponential kernel function

```
# self defined SEkearnel
SEkernel_seldef <- function(sigmaf = 1, ell = 1)
{
  rval <- function(x, y) {
    r <- outer(x,y,function(x1,x2){x1-x2})
    return(sigmaf^2*exp(-r^2/2/ell^2))
  }
  class(rval) <- "kernel"
  return(rval)
}
```

```
## Evaluating the kernel in x1=1, x2=2:
```

```
##           [,1]
## [1,] 0.6065307
```

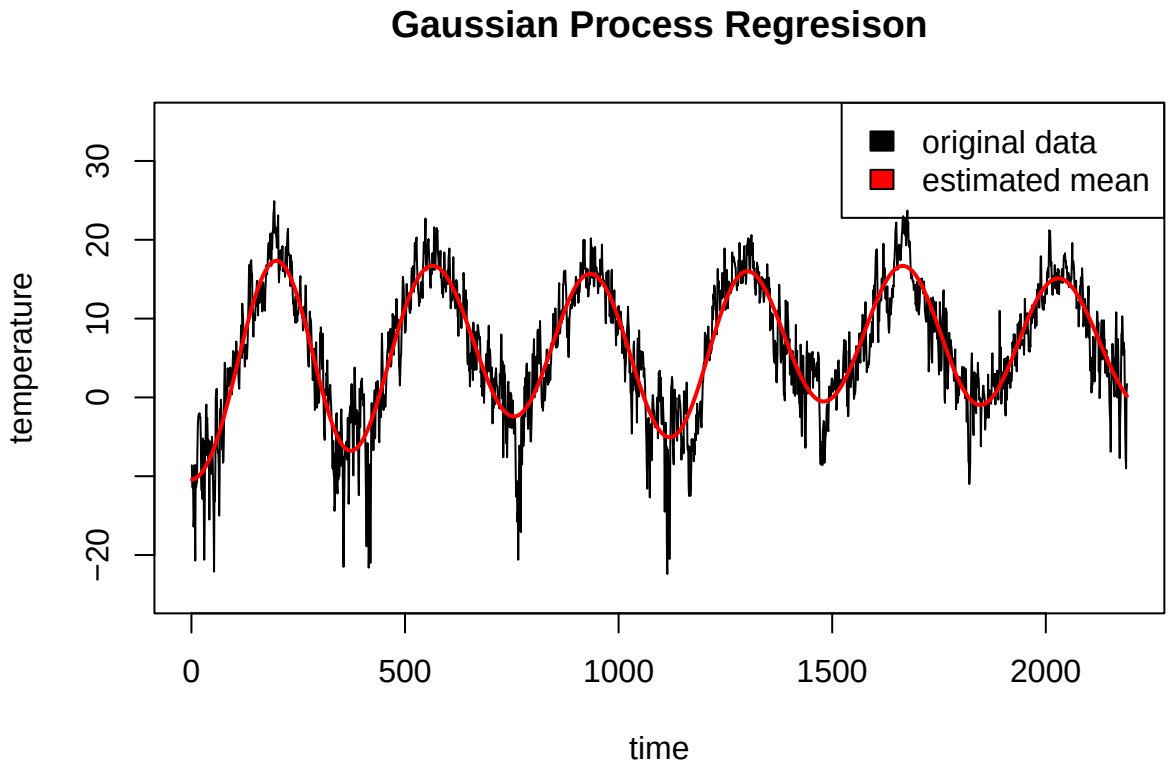
```
## Covariance matrix K(X,Xstar):
```

```
## 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
```

(2) Temperature - Time Model

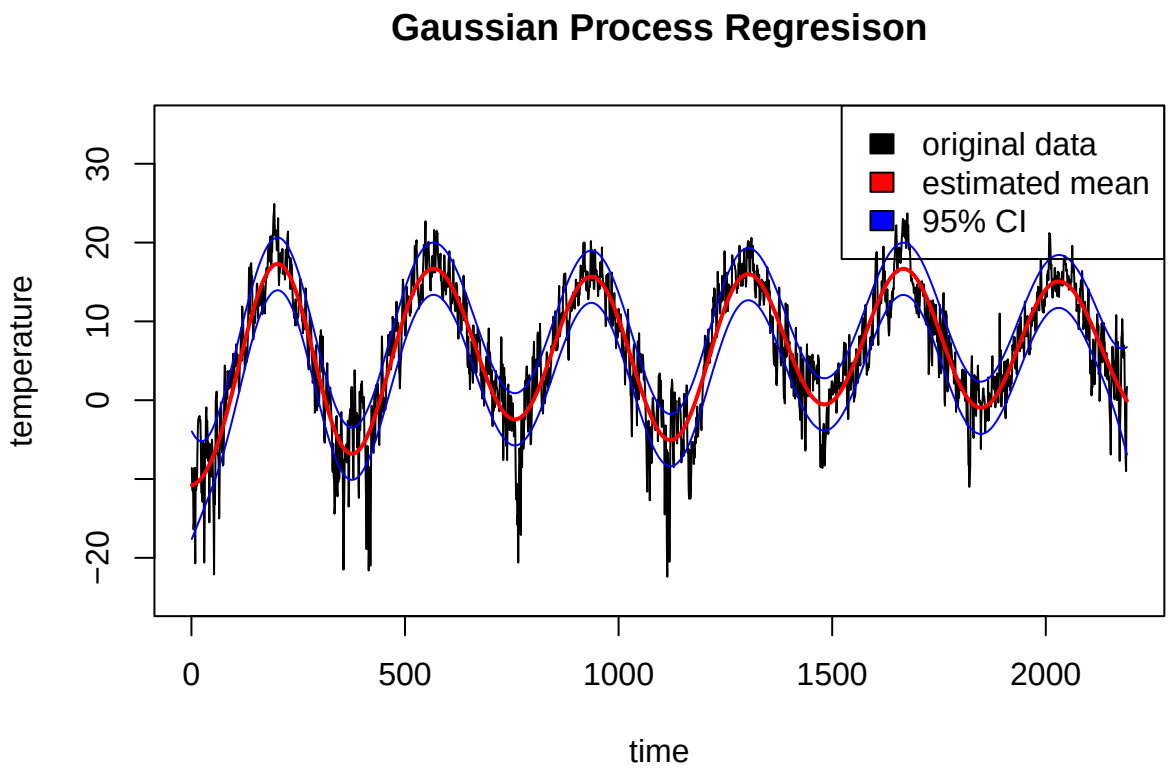
The Time model is built to predict temperature given *time* from the Gaussian process. The prediction (red) was plotted over the original data (black).



As we can see from the plot the Gaussian process seem to fit the data decently well.

(3) temperature - time model using self-define Gaussian process

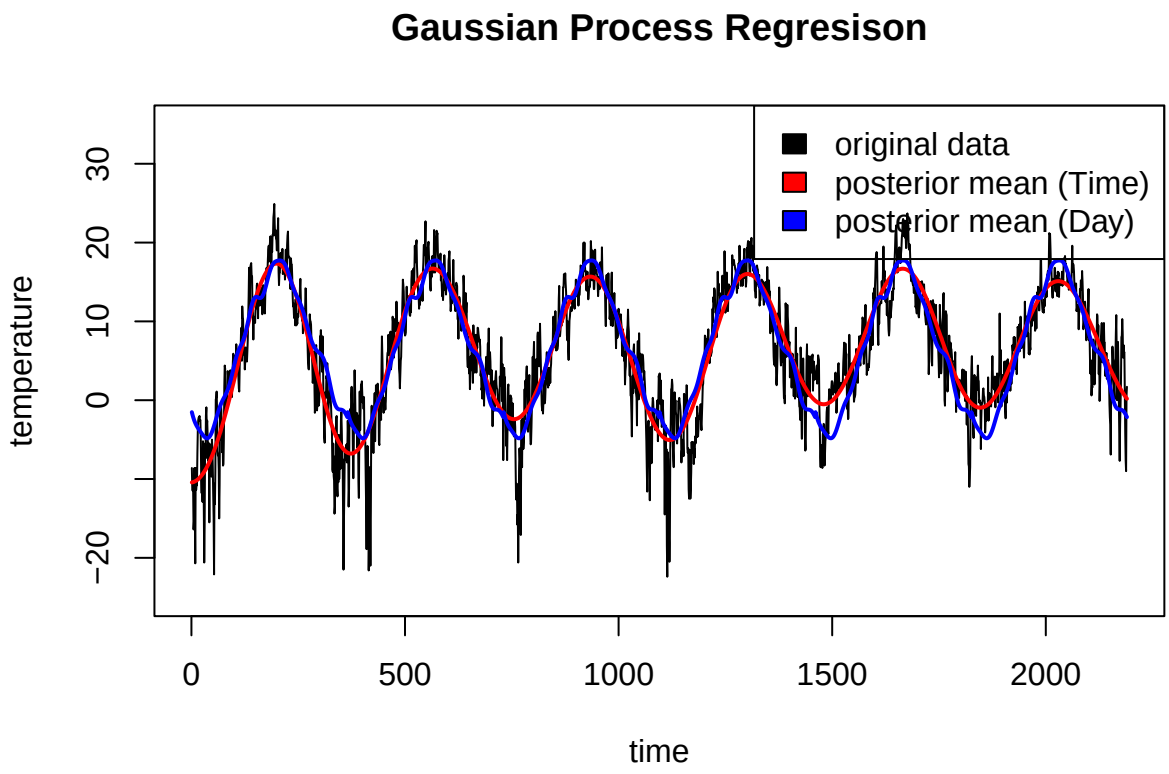
The following result shows the 95% probability intervals for the predictions in the previous question. The intervals was added to the plot (blue).



As we can see roughly 95% of the datapoints is within the 95% probability interval.

(4) temperature - day model

The Day model is constructed to predict temperature given variable *day* instead of *time*. The predictions using *day* was plotted (blue) over the predictions using *time* (red).



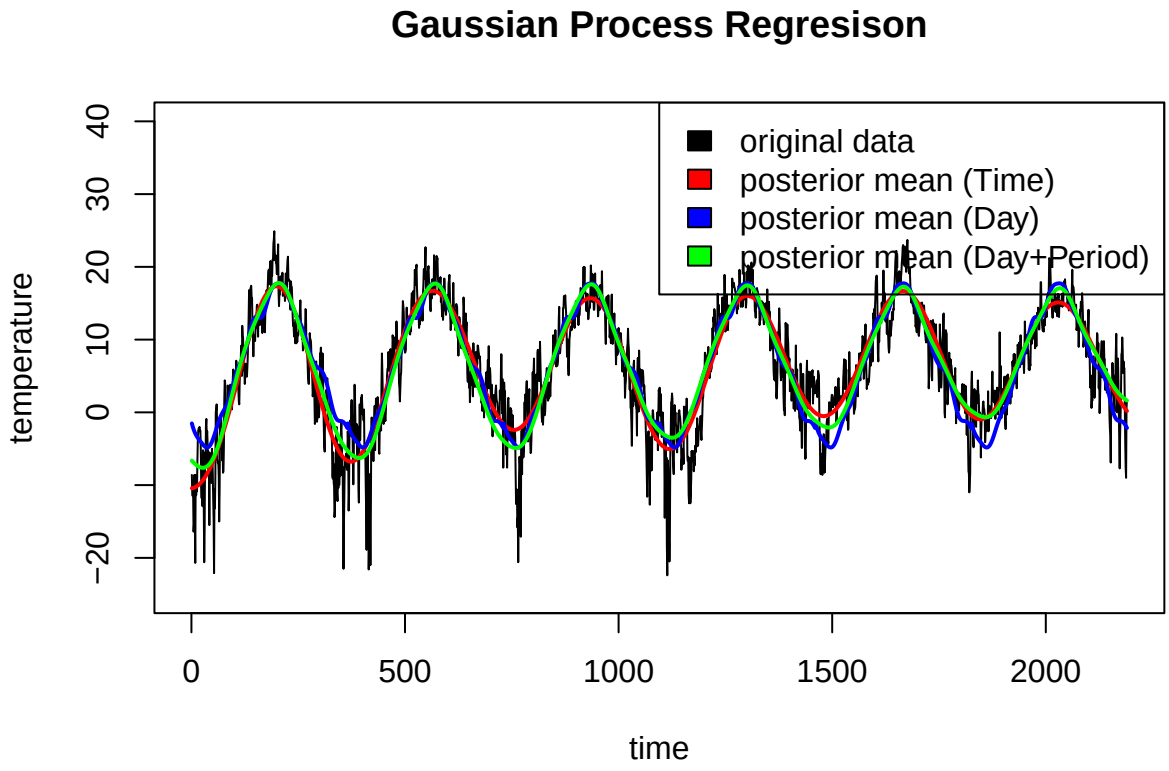
Time Model The time model considers the recent day in predicting the temperature while ignoring the information of the same date in each year. This model could be long at estimating the trend information, but neglect the periodic component.

Day Model In contrast, the Day model put higher correlation on the similar days in each year, for example, it will consider the day 6 for all years the most when predict the temperature in day 6 of one year. This model only consider the periodic information, compared with trend information in the time model. It could be clear that the predictions for *day* has a constant amplitude for all years.

(5) temperature - day model using periodic kernel

The previous exercise was repeated with a periodic kernel. The predictions (green) was compared to the previous two predictions (*day* in blue and *time* in red) and plotted the following.

```
# Redefine the Kernel
SEkernel_seldef <- function(sigmaf = 1, l1 = 1, l2,d)
{
  rval <- function(x, y) {
    r <- outer(x,y,function(x1,x2){abs(x1-x2)})
    return(sigmaf^2*exp(-(2*sin(pi*r/d)^2)/(l1^2))*exp(-1/2*(r^2/l2^2)))
  }
  class(rval) <- "kernel"
  return(rval)
}
```



Compared with the previous two models, periodic kernel considers both the recent time and the same period in each year, which combines the advantages of the time and day models. Thus, the result from it is more reliable.

Question 3

(1)

The following result is the prediction for $fraud$ based on $skewWave$ and $varWave$. The datapoints was plotted over the contour plot where blue meant $fraud$ and red meant not $fraud$.

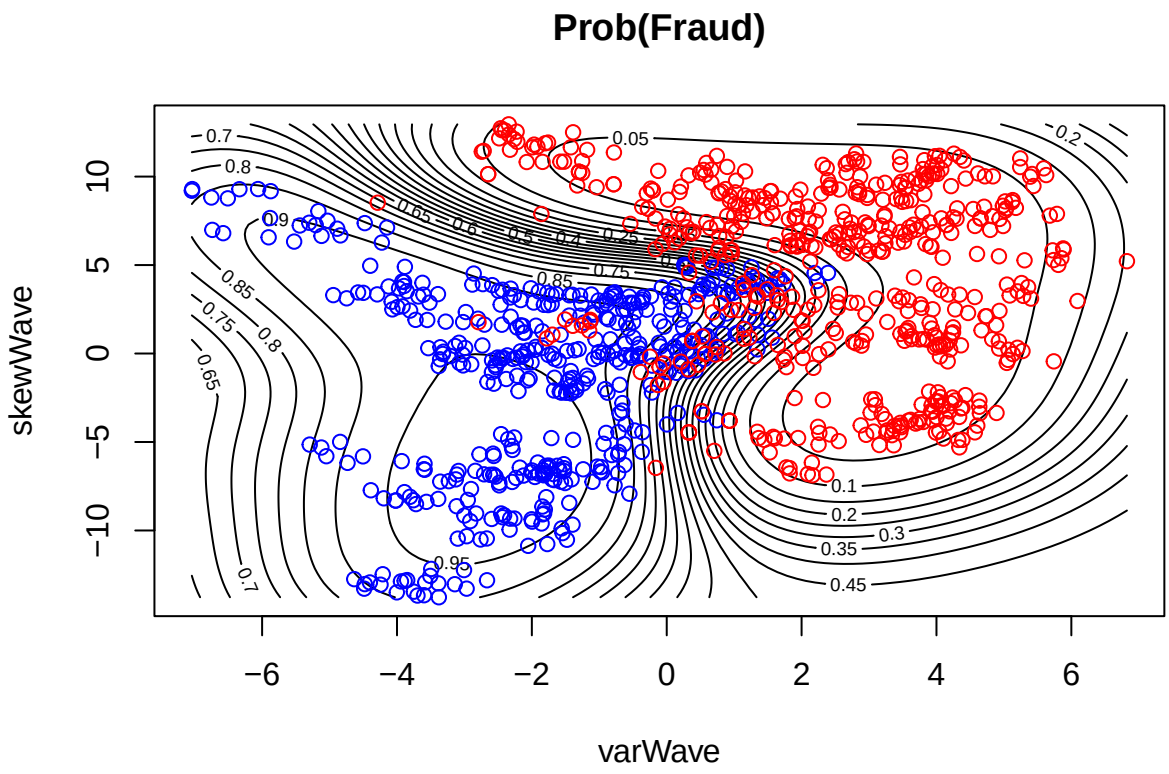
```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
## The confusion matrix:
```

```
##            true
## prediction   0   1
##          0 503  18
##          1  41 438
```

```
## The accuracy rate:
```

```
## [1] 0.941
```



As we can see there is a lot of uncertainty in the middle where a lot of blue and red points overlap (probability is ≈ 0.5). Where the blue and red points are "alone" we have high certainty of classification (probability is ≈ 1 or ≈ 0).

(2) Accuracy for test set prediction

Fraud was predicted on the test set and the accuracy was calculated as following.

```
## The confusion matrix:
```

```
##            true
## prediction   0   1
##          0 199   9
##          1  19 145
```

```
## The accuracy rate:
```

```
## [1] 0.9247312
```

The accuracy was 0.9247312 which seems pretty good.

(3) Accuracy with four covariates

The Gaussian process was now trained on all four covariates and accuracy was calculated as following.

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

```
## The confusion matrix:
```

```
##            true
## prediction   0   1
##          0 216   0
```

```
##          1   2 154
```

```
## The accuracy rate:
```

```
## [1] 0.9946237
```

This resulted in an accuracy of 0.9946237 which is a lot better than the accuracy in the previous exercise. This makes sense since we used more information to base our predictions on.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(AtmRay)
library(kernlab)
# Kernel function from course code
SquaredExpKernel <- function(x1,x2,sigmaF=1,l=0.3){
  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,sigmaNoise,k,...)
{
  n = length(X)
  K = k(X,X,...)
  Kstar  =k(X,Xstar,...)
  # GP_L2 P11 LINE:2
  L = t(chol(K+(sigmaNoise**2)*diag(n)))
  alpha = solve(t(L),solve(L,y))
  # GP_L2 P11 LINE:4
  f = t(Kstar) %*% alpha
  v = solve(L,Kstar,...)
  # GP_L2 P11 LINE:6
  vf = k(Xstar,Xstar)-t(v)%*%v
  res = list('f_mean'=f,'f_variance'=vf)
}

x2 = c(0.4)
y2= c(0.719)
xstar = seq(-1,1,length=100)
sigmaNoise=0.1
k = SquaredExpKernel
gp_q2 <- posteriorGP(x2, y2, xstar, sigmaNoise, k)


plot_res <- function(x, y,obx,oby,sigma,l)
{
  plot(x,y$f_mean,col='red',ylab='pos mean',type='l',ylim=c(-2.5,2.5),main=
         paste0('sigma=',sigma,'  l=',l))
  lines(x,y$f_mean+1.96*sqrt(diag(y$f_variance)),col='blue')
  lines(x,y$f_mean-1.96*sqrt(diag(y$f_variance)),col='green')
```

```
  legend('topright',pch=1,cex=0.5,legend = c('mean','upper 95%','lower 95%','obs'),
  col=c('red','blue','green','black'))
  points(obx,oby,col='black')
}
plot_res(xstar,gp_q2,x2,y2,1,0.3)

x3 = c(0.4,-0.6)
y3 = c(0.719,-0.044)
gp_q3 <- posteriorGP(x3, y3, xstar, sigmaNoise, k)
plot_res(xstar,gp_q3,x3,y3,1,0.3)

x4 = c(-1,-0.6,-0.2,0.4,0.8)
y4 = c(0.768,-0.044,-0.94,0.719,-0.664)
gp_q4 <- posteriorGP(x4, y4, xstar, sigmaNoise, k)
plot_res(xstar,gp_q4,x4,y4,1,0.3)
x5 = c(-1,-0.6,-0.2,0.4,0.8)
y5 = c(0.768,-0.044,-0.94,0.719,-0.664)
gp_q5 <- posteriorGP(x5, y5, xstar, sigmaNoise, k,sigmaF=1,l=1)
plot_res(xstar,gp_q5,x5,y5,1,1)

# self defined SEkearnel
SEkernel_seldef <- function(sigmaf = 1, ell = 1)
{
  rval <- function(x, y) {
    r <- outer(x,y,function(x1,x2){x1-x2})
    return(sigmaf^2*exp(-r^2/2/ell^2))
  }
  class(rval) <- "kernel"
  return(rval)
}
data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/
Code/TempTullinge.csv", header=TRUE, sep=";")

N <- dim(data)[1]
time <- seq(1,N,5)
day <- rep(seq(1,365,5),6)

# combine the variables needed in the following questions
data2 <- data.frame(data,time = 1:N, day = rep(seq(1:365),6))

# Testing our own defined kernel function.
X <- c(1,3,4) # Simulating some data.
Xstar <- c(2,3,4)
SEkernel_seldef_t <- SEkernel_seldef(sigmaf = 1, ell = 1) # MaternFunc is a kernel FUNCTION.
cat('Evaluating the kernel in x1=1, x2=2:','\n')
SEkernel_seldef_t(1,2) # Evaluating the kernel in x=c(1,1), x'=c(2,2).
# Computing the whole covariance matrix K from the kernel.
cat('Covariance matrix K(X,Xstar):','\n')
kernelMatrix(kernel = SEkernel_seldef_t, x = X, y = Xstar) # So this is K(X,Xstar).
#Add the time^2 to compute the quadratic regression
data2$time2 <- data2$time^2
lm_model <- lm(formula = temp ~ time + time2,data = data2[time,])
sigma2_n <- var(lm_model$residuals)
```

```r
# set the kernel function
kern <- SEkernel_seldef(20,0.2)

# Gaussian Process Regression
res_gp <- gausspr(x = data2$time[time],
                  y = data2$temp[time],
                  var = sigma2_n,
                  data = data2[time,],
                  kernel = kern)

# Prediction
pred_temp <- predict(res_gp,data2[,3])

# Plot Figure
plot(data2$time, data2$temp,type = 'l', ylim = c(-25,35),
     xlab = 'time',ylab = 'temperature',main = 'Gaussian Process Regresison')
lines(data2$time,pred_temp,col = 'red', lwd = 2)
legend('topright',
        legend = c('original data','estimated mean'),
        fill = c('black','red'))

# Scale X and X_star
x_star <- data2[,3]
x_star <- scale(x_star)[,1]
X <- data2$time[time]
X_scale <- scale(X)[,1]
y <- data2$temp[time]
kern <- SEkernel_seldef(20,0.2)
# Compute 95% CI
res_23 <- posteriorGP(X = X_scale,
                      y = y,
                      Xstar = x_star,
                      sigmaNoise = sqrt(sigma2_n),
                      k = kern)

# Plot Figure
plot(data2$time, data2$temp,type = 'l', ylim = c(-25,35),
     xlab = 'time',ylab = 'temperature',main = 'Gaussian Process Regresison')
lines(data2$time, res_23$f_mean, col = 'red',lwd = 2)
lines(data2$time, res_23$f_mean + 1.96*(sqrt(diag(res_23$f_variance)) ),col = 'blue')
lines(data2$time, res_23$f_mean - 1.96*(sqrt(diag(res_23$f_variance)) ),col = 'blue')
legend('topright',
        legend = c('original data','estimated mean','95% CI'),
        fill = c('black','red','blue'))
# Add day^2 for variance computation
data2$day2 <- data2$day^2
lm_model_day <- lm(formula = temp ~ day + day2,data = data2[time,])
sigma2_n_day <- var(lm_model_day$residuals)

# Initialize Kernel
kern <- SEkernel_seldef(20,0.2)

# Do GPR
```

```r
res_gp_day <- gausspr(x = data2$day[time],
                      y = data2$temp[time],
                      var = sigma2_n_day,
                      data = data2[time,],
                      kernel = kern)

# Predict
pred_temp_day <- predict(res_gp_day,data2$day)

# Plot Figure
plot(data2$time,data2$temp,type = 'l', ylim = c(-25,35),
     xlab = 'time',ylab = 'temperature',main = 'Gaussian Process Regresison')
lines(data2$time,pred_temp,col = 'red',lwd = 2)
lines(data2$time,pred_temp_day,col = 'blue',lwd = 2)
legend('topright',
        legend = c('original data','posterior mean (Time)','posterior mean (Day)'),
        fill = c('black','red','blue'))
# Redefine the Kernel
SEkernel_seldef <- function(sigmaf = 1, l1 = 1, l2,d)
{
  rval <- function(x, y) {
    r <- outer(x,y,function(x1,x2){abs(x1-x2)})
    return(sigmaf^2*exp(-(2*sin(pi*r/d)^2)/(l1^2))*exp(-1/2*(r^2/l2^2)))
  }
  class(rval) <- "kernel"
  return(rval)
}
# Initialize parameter
d <- 365/sd(data2$time[time])

# Initialize Kernel
kern_period <- SEkernel_seldef(20,1,10,d)

# Do GPR
res_25 <- gausspr(x = data2$time[time],
                  y = data2$temp[time],
                  var = sigma2_n,
                  kernel = kern_period,
                  data = data2[time,])

# Prediction
pred_temp_day_period <- predict(res_25,data2[,3])

# Plot Figure
plot(data2$time, data2$temp,type = 'l',ylim = c(-25,40),
     xlab = 'time',ylab = 'temperature',main = 'Gaussian Process Regresison')
lines(data2$time,pred_temp,col = 'red',lwd = 2)
lines(data2$time,pred_temp_day,col = 'blue',lwd = 2)
lines(data2$time,pred_temp_day_period,col = 'green',lwd = 2)
legend('topright',
        legend = c('original data','posterior mean (Time)','posterior mean (Day)','posterior mean (Day+
        fill = c('black','red','blue','green'))
```

```
data <- read.csv("https://raw.githubusercontent.com/STIMALiU/AdvMLCourse/master/GaussianProcess/Code/ba
names(data) <- c("varWave","skewWave","kurtWave","entropyWave","fraud")
data[,5] <- as.factor(data[,5])

set.seed(111); SelectTraining <- sample(1:dim(data)[1], size = 1000,
                                        replace = FALSE)
train = data[SelectTraining,]
SelectTest = setdiff(1:dim(data)[1], SelectTraining)
test = data[SelectTest,]
pr = gausspr(fraud~varWave + skewWave, train)
pred_fraud <- predict(pr, train[,1:2])

tab1 <- table('prediction' = pred_fraud, 'true'=train$fraud)
cat('The confusion matrix:', '\n')
tab1

acc1 <- sum(diag(tab1))/sum(tab1)
cat('The accuracy rate:','\n')
acc1

probPreds <- predict(pr, train[,1:2], type="probabilities")
x1 <- seq(min(train[,1]),max(train[,1]),length=100)
x2 <- seq(min(train[,2]),max(train[,2]),length=100)
gridPoints <- meshgrid(x1, x2)
gridPoints <- cbind(c(gridPoints$x), c(gridPoints$y))

gridPoints <- data.frame(gridPoints)
names(gridPoints) <- names(train)[1:2]
probPreds <- predict(pr, gridPoints, type="probabilities")

contour(x1,x2,matrix(probPreds[,2],100,byrow = TRUE), 20, xlab = "varWave", ylab = "skewWave", main = '
points(train[train$fraud ==1, 1],train[train$fraud ==1, 2],col="blue")
points(train[train$fraud ==0, 1],train[train$fraud ==0, 2],col="red")
pred = predict(pr, test)

tab2 <- table('prediction' = pred, 'true'=test$fraud)
cat('The confusion matrix:', '\n')
tab2

cat('The accuracy rate:','\n')
mean(pred == test$fraud)
pr = gausspr(fraud~., train)
pred = predict(pr, test)

tab3 <- table('prediction' = pred, 'true'=test$fraud)
cat('The confusion matrix:', '\n')
tab3

cat('The accuracy rate:','\n')
mean(pred == test$fraud)
```