Lab 4

Kristian Sikiric (krisi211)

Assignment 1

1.1

First the algorithm from Rasmussen and Willams' book to be able to do gausian regression was implemented. Below the implementation in r can be seen.

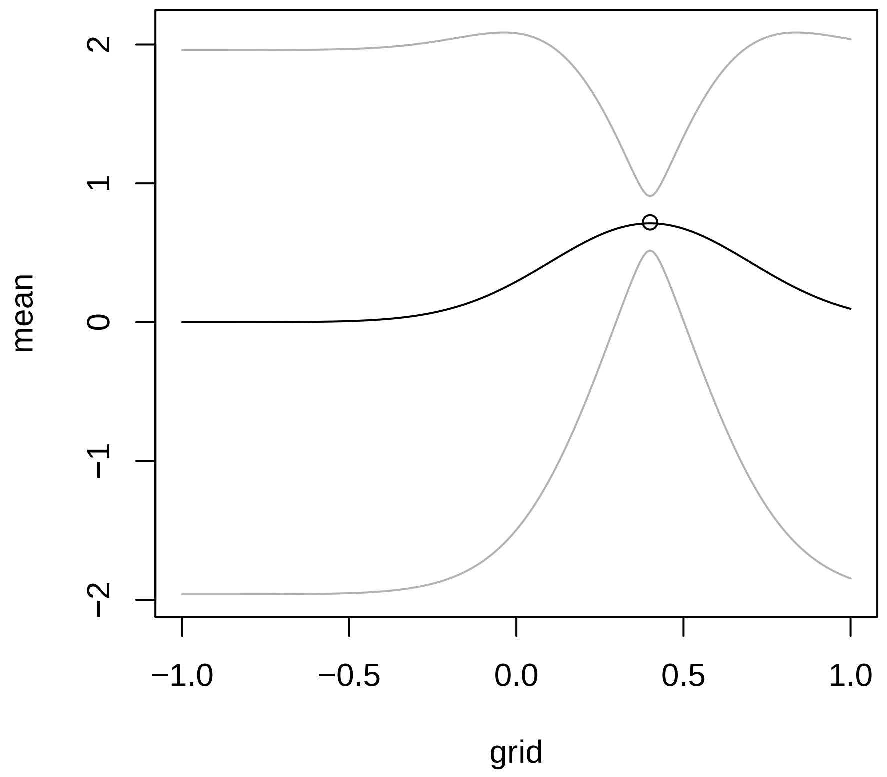
```r
posteriorGP = function(X,y,XStar,hyperParam,sigmaNoise){
  n = length((X))
  K = squeredExpKernel(X,X,hyperParam[1],hyperParam[2])
  L = t(chol(K+(sigmaNoise^2)*diag(n)))
  alpha = solve(t(L),solve(L,y))
  kstar = squeredExpKernel(X,XStar,hyperParam[1], hyperParam[2])
  fbarstar = t(kstar)%*%alpha

  v = solve(L, kstar)
  V.f = squeredExpKernel(XStar,XStar,hyperParam[1], hyperParam[2])-t(v)%*%v


  return(list("Mean" = fbarstar, "Variance" = diag(V.f)))
}
```

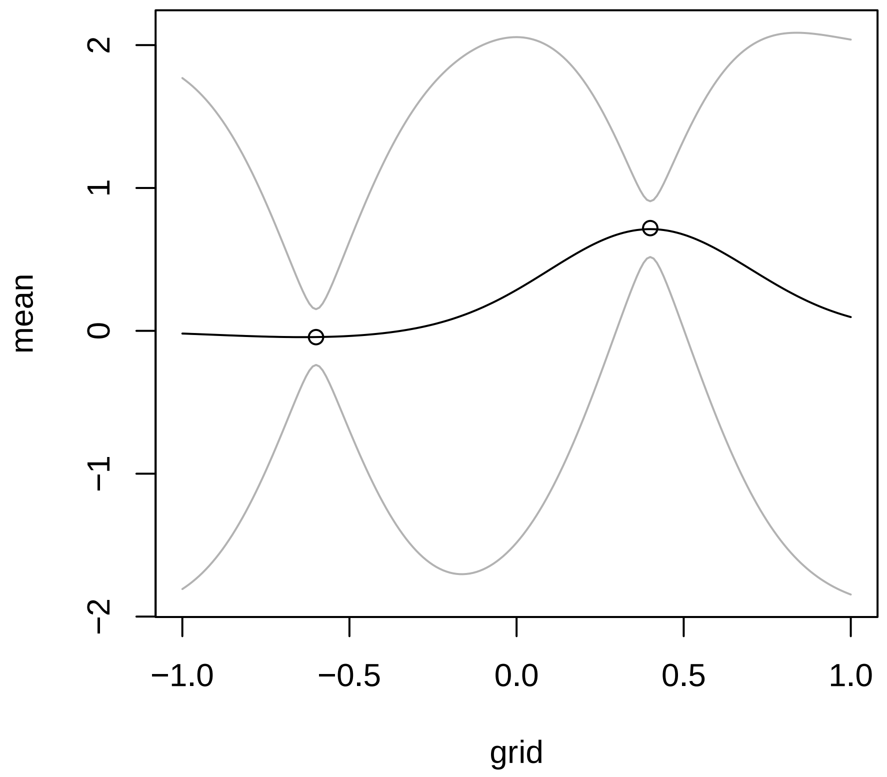
1.2

Now we were to update our prior with a single observation: $x = 0.4$ and $y = 0.719$. The hyperparameters were set to $\sigma_f = 1$ and $l = 0.3$ and we assumed $\sigma_n = 0.1$ The 95% probability bands were also calculated, see plot below.



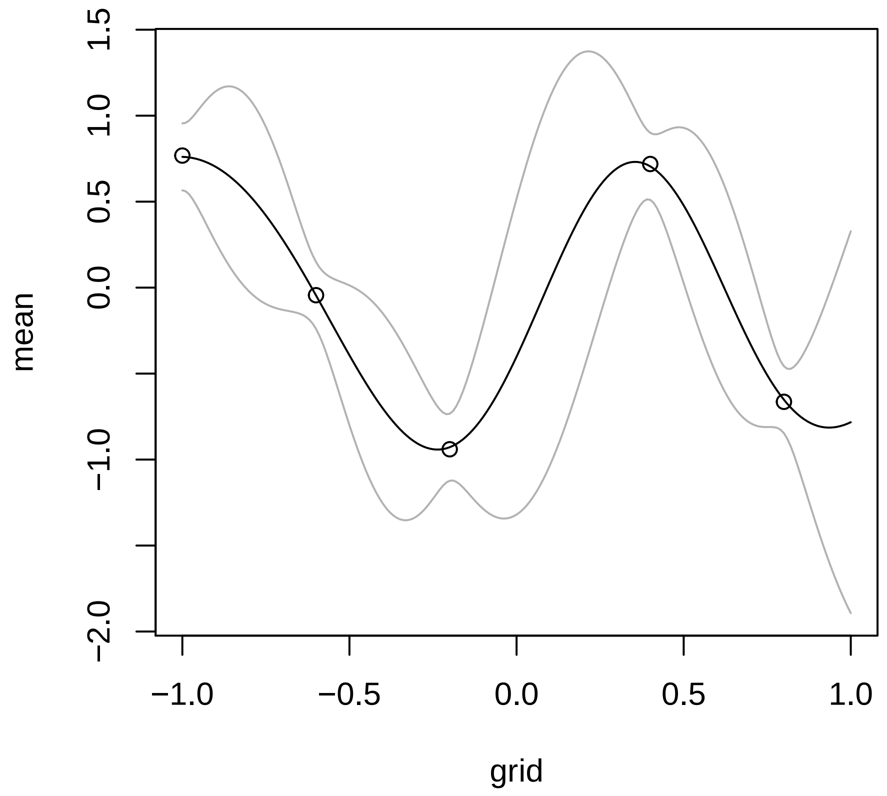
1.3

Now we were to update the postierior from 1.2 with a new observation $x = -0.6$ and $y = -0.044$. Since this is the same as updating the prior with two values at once, the values $x = (0.4, -0.6)$ and $y = (0.719, -0.044)$ was sent to the funciton. The following plot shows the mean and the 95% probability bands.



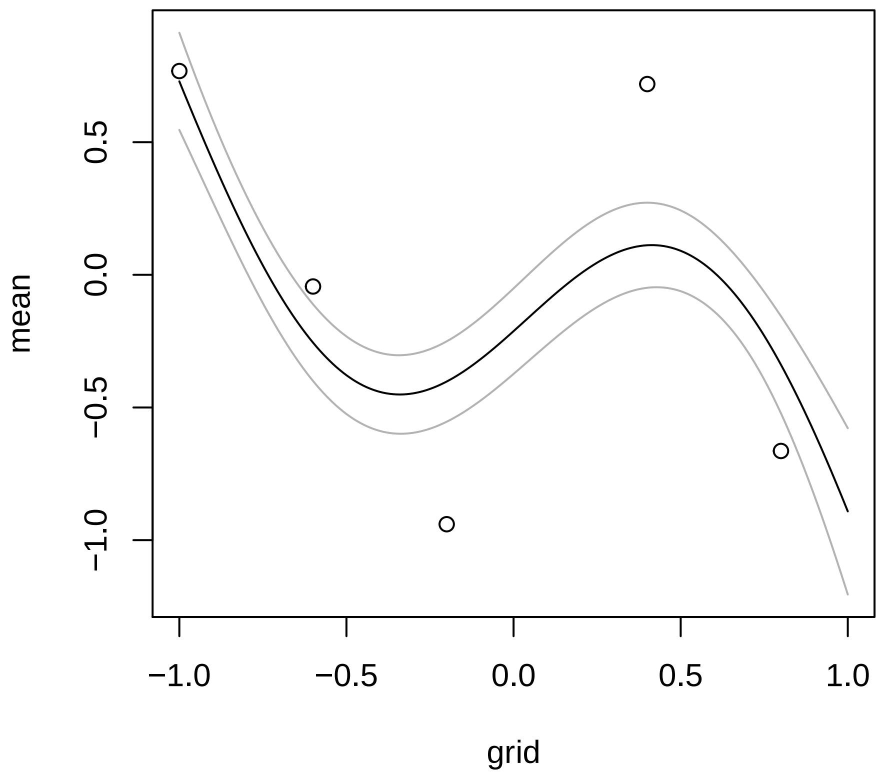
1.4

Now we were to compute the posterior with all the five data points below. $x = (-1.0, -0.6, -0.2, 0.4, 0.8)$ and $y = (0.768, -0.044, -0.940, 0.719, -0.664)$ Below is the plot of the mean and the 95% probability bands.



1.5

Now we were to repeat the last assignment but with $l = 1$. The following plot shows the resulting mean and 95% probability bands.



We see that the mean no longer goes through the five points sent to the function, as compared to assignment 1.4 were they do. The 95% probability bands are much more even than in assignment 1.4 as well. This is due to the l paramater specifying the smoothing of the curve.

Assignment 2

In this assignment we were to do GP regression with the library kernlab. The regression was to be done on temperature data for each day in a 6 year period. First the kernel was to be implemented, this is the same kernel as in assignment 1, but it is nested in a class.

```
GPkernel = function(sigmaf,ell){
  squeredExpKernel= function(x,y){
    n1 = length(x)
    n2 = length(y)
    k = matrix(NA,n1,n2)
    for (i in 1:n2){
      k[,i] = sigmaf^2*exp(-0.5*( (x-y[i])/ell)^2 )
    }
    return(k)
  }
  class(squeredExpKernel) <- "kernel"
  return(squeredExpKernel)
}
```

2.1

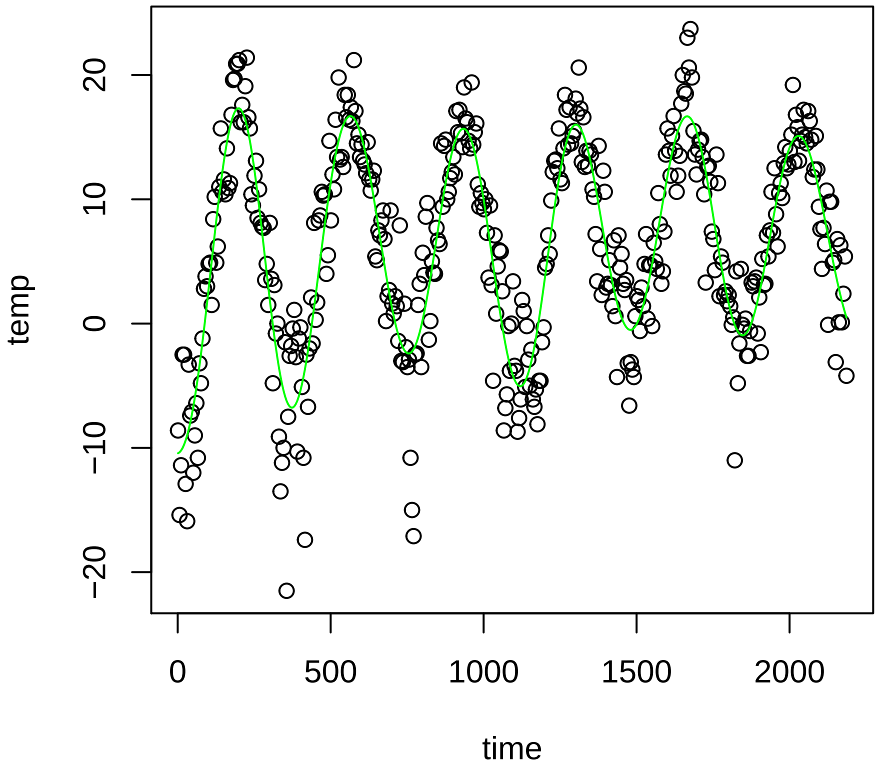
The kernel was evaluated for $sigma_f = 20$ and $l = 0.2$ giving the following result:

```
##              [,1]
## [1,] 0.001490661
```

The kernel was evaluated on the points $x = 1$ and $x' = 2$. This can be interpreted that the kernel does not affect nearby values much. This makes sense, since we are prediciting temperatures, and maybee the temperatures in one day does not affect the temperatures the next day, so it is reasonable that the kernel do not care to much about the temperatures for the next day.

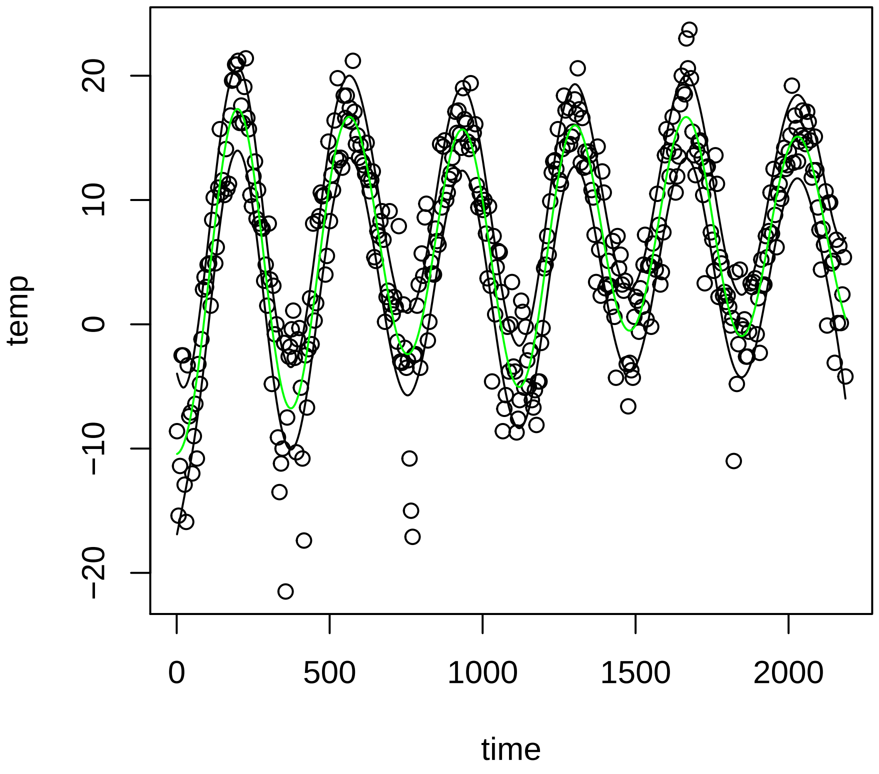
2.2

Now we were to predict the temperature given the time, were time is the number of days from the start of the dataset. $sigma_f = 20$ and $l = 0.2$ was used in the kernel, and the σ_n was calculated by first fitting a simple qudratic linear regression. The kernel from 2.1 was used in the GP regression, the following plot shows a scatter plot of the temperatures given the time, and the predicitve mean is plotted as the green line.



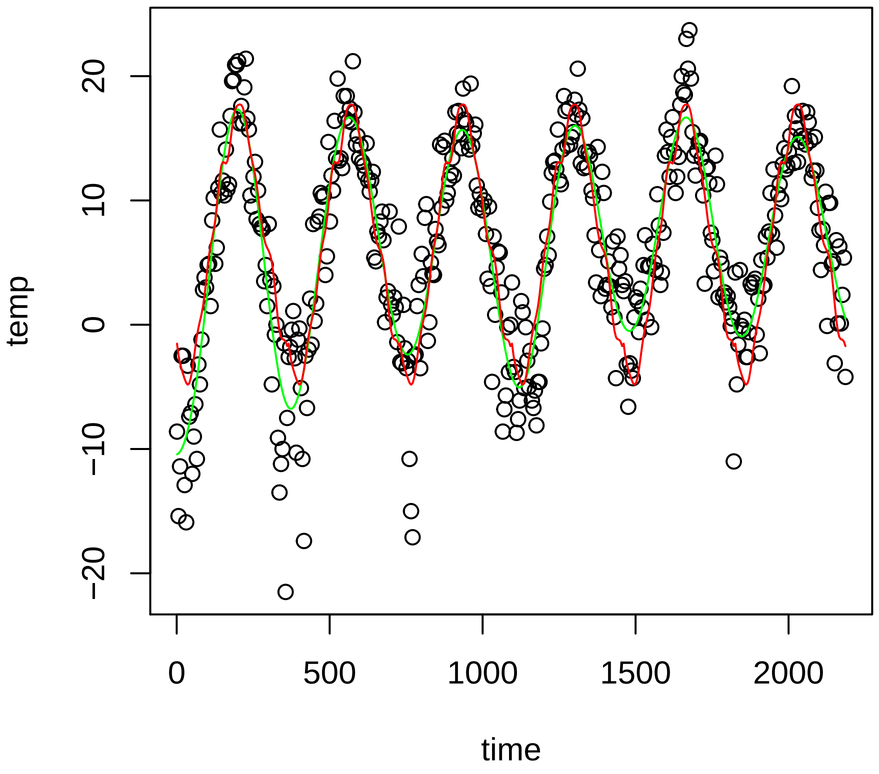
2.3

The variance was calculated using the algorithm in assignment 1. From the variance the 95% probability bands was computed and ploted, they are the black lines in the plot below.



2.4

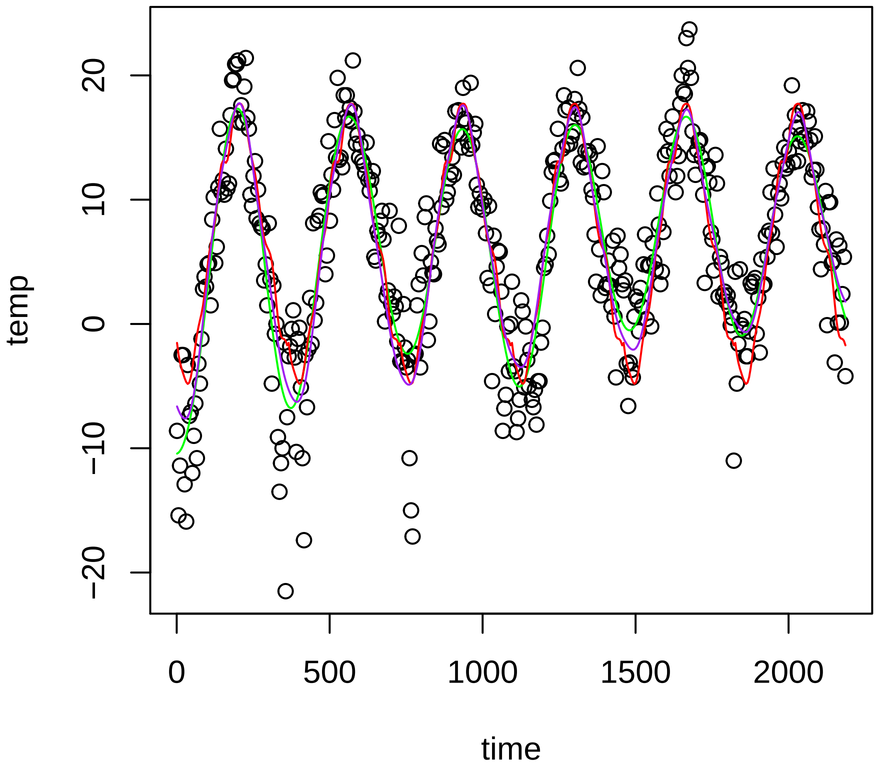
Now we were to redo assignment 2.2 but with days instead of time, were days are the days since the start of the year. The resulting predicitive mean is plotted below in red together with the mean from 2.2 (in green).



We see that there is not much difference, both models fit the data good.

2.5

Now we were to redo assignment 2.2 but with a different kernel. The kernel was given by the following function: $k(x, x') = \sigma_f^2 exp(-\frac{2*sin^2(pi|x-x'|/d)}{l_1^2})exp(-0.5\frac{|x-x'|^2}{l_2^2})$ were $\sigma_f = 20, l_1 = 1, l_2 = 10, d = 365/sd(time)$. Now a GP regression model was fitted with this kernel and plotted below in purple together with the two models from the previous assignments.



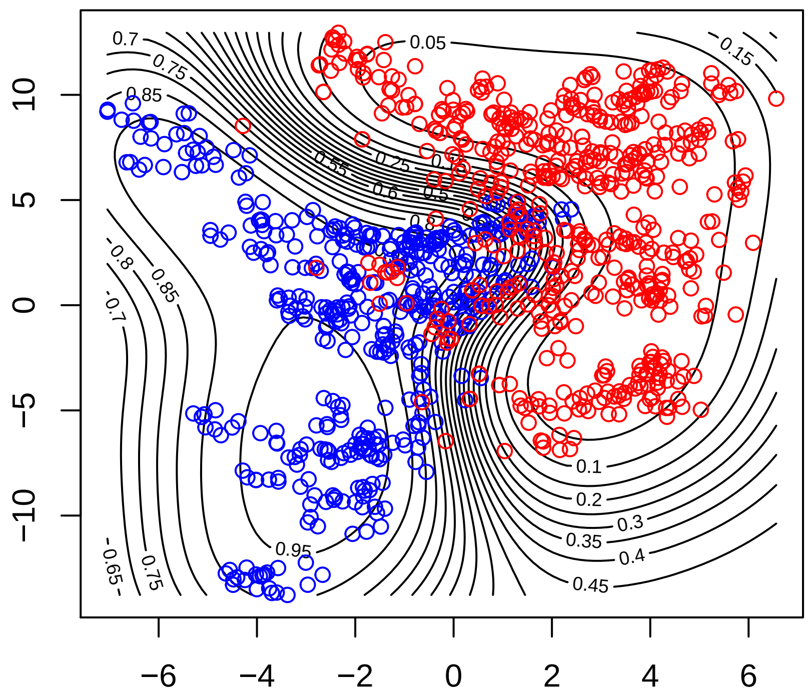
All three models looks similar and fits the data very i good in my opinion.

Assignment 3

3.1

Now we were to do classification with the kernlab library. First we were to train a model with only 2 out of 4 covariates. Below are the contour of the prediction probabilities overlayed with the target variable, fraud. The dots are colored blue when fraud is equal to 1 and red when fraud is equal to 0.

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



Below is the confusion matrix and accuracy when predicting on the train data.

```
##
##       0   1
##   0 512  24
##   1  44 420
```

```
## [1] 0.932
```

3.2

Now we used the model from 3.1 and computed the confusion matrix and accuracy on the test data set.

```
##
##       0   1
##   0 191   9
##   1  15 157
```

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

We ee that the test accuracy is similar to the train data, showing that we did not overfit when training.

3.3

Now we were to train the model on all covariates and make predictions on the test set, below is the confusion matrix and accuracy.

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

```
##
##       0   1
##   0 205   0
##   1   1 166
```

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

Appendix - Code

```
###### Functions #######
posteriorGP = function(X,y,XStar,hyperParam,sigmaNoise){
  n = length((X))
  K = squeredExpKernel(X,X,hyperParam[1],hyperParam[2])
  L = t(chol(K+(sigmaNoise^2)*diag(n)))
  alpha = solve(t(L),solve(L,y))
  kstar = squeredExpKernel(X,XStar,hyperParam[1], hyperParam[2])
  fbarstar = t(kstar)%*%alpha

  v = solve(L, kstar)
  V.f = squeredExpKernel(XStar,XStar,hyperParam[1], hyperParam[2])-t(v)%*%v


  return(list("Mean" = fbarstar, "Variance" = diag(V.f)))
}

squeredExpKernel= function(x1,x2,sigmaF,l){
  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)
}

GPkernel = function(sigmaf,ell){
  squeredExpKernel= function(x,y){
    n1 = length(x)
    n2 = length(y)
    k = matrix(NA,n1,n2)
    for (i in 1:n2){
      k[,i] = sigmaf^2*exp(-0.5*( (x-y[i])/ell)^2 )
    }
    return(k)
  }
  class(squeredExpKernel) <- "kernel"
  return(squeredExpKernel)
}

plotGP = function(mean,variance,grid,x,y){
  plot(grid,mean,ylim = c(min(mean-1.96*sqrt(variance))
                          ,max(mean+1.96*sqrt(variance))),
       type = "l")
  lines(grid,
        mean+1.96*sqrt(variance),
        col = rgb(0, 0, 0, 0.3))
  lines(grid,
        mean-1.96*sqrt(variance),
        col = rgb(0, 0, 0, 0.3))
  points(x,y)
```

```r
}

GP.periodic.kernel = function(sigmaf, l1,l2,d){
  periodicKernel = function(x1,x2){
    return(  (sigmaF^2)*exp(-(2*sin(pi*abs(x1-x2)/d)^2)/l1^2)*
               exp(-0.5*(abs(x1-x2)^2)/l2^2))
  }
  class(periodicKernel) <- "kernel"
  return(periodicKernel)
}




###########################
###### Assignment 1 #######
###########################
###### 1.2 ######
sigmaF = 1
l = 0.3
hyperParam = c(sigmaF,l)
sigmaNoise = 0.1
X = 0.4
y = 0.719

XStar = seq(-1,1,by=0.01)
GP = posteriorGP(X,y,XStar,hyperParam,sigmaNoise)
plotGP(GP$Mean,GP$Variance,XStar,X,y)

##### 1.3 ######
sigmaF = 1
l = 0.3
hyperParam = c(sigmaF,l)
sigmaNoise = 0.1
X = c(0.4,-0.6)
y = c(0.719,-0.044)

XStar = seq(-1,1,by=0.01)
GP = posteriorGP(X,y,XStar,hyperParam,sigmaNoise)
plotGP(GP$Mean,GP$Variance,XStar,X,y)

##### 1.4 ######
sigmaF = 1
l = 0.3
hyperParam = c(sigmaF,l)
sigmaNoise = 0.1
X = c(-1.0, -0.6, -0.2, 0.4, 0.8)
y = c(0.768, -0.044, -0.940, 0.719, -0.664)

XStar = seq(-1,1,by=0.01)
GP = posteriorGP(X,y,XStar,hyperParam,sigmaNoise)
plotGP(GP$Mean,GP$Variance,XStar,X,y)
```

```
##### 1.5 ######
sigmaF = 1
l = 1
hyperParam = c(sigmaF,l)
sigmaNoise = 0.1
X = c(-1.0, -0.6, -0.2, 0.4, 0.8)
y = c(0.768, -0.044, -0.940, 0.719, -0.664)

XStar = seq(-1,1,by=0.01)
GP = posteriorGP(X,y,XStar,hyperParam,sigmaNoise)
plotGP(GP$Mean,GP$Variance,XStar,X,y)

##########################
###### Assignment 2 ######
##########################

###### Innit ######
library(kernlab)
data = read.csv(
  "https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullinge.csv",
  header=TRUE,
  sep=";")

time = seq(1,2190,by=5)
day = rep(seq(1,365,by=5),6)
###### 2.1 ######
X = c(1,3,4)
XStar = c(2,3,4)
x = 1
xprim = 2

gpK = GPkernel(sigmaf = 20,ell = 0.2)
gpK(x = 1, y = 2)
K = kernelMatrix(gpK,x=X,y=XStar)

###### 2.2 ######
sigmaF = 20
l = 0.2
temp = data[time,]$temp
lm.fit = lm(temp ~ time + I(time^2))
sigmaNoise = sd(lm.fit$residuals)

GPfit = gausspr(x = time,
                y = temp,
                kernel = GPkernel,
                kpar = list(sigmaf = sigmaF, ell = l),
                var = sigmaNoise^2)
meanPred = predict(GPfit,time) #Predict on train data
plot(time,temp)
lines(time,meanPred, col = "green")

###### 2.3 ######
var = posteriorGP(X = scale(time),
                  y = scale(temp),
                  XStar = scale(time),
                  hyperParam = c(sigmaF,l),
                  sigmaNoise = sigmaNoise)$Variance
lines(time,meanPred+1.96*sqrt(var), col = "black")
lines(time,meanPred-1.96*sqrt(var), col = "black")

###### 2.4 ######
sigmaF = 20
l = 0.2
temp = data[time,]$temp
lm.fit = lm(temp ~ day + I(day^2))
sigmaNoise = sd(lm.fit$residuals)

GPfit = gausspr(x = day,
                y = temp,
                kernel = GPkernel,
                kpar = list(sigmaf = sigmaF, ell = l),
                var = sigmaNoise^2)
meanPred = predict(GPfit,day) #Predict on train data
lines(time,meanPred, col = "red")

###### 2.5 ######
sigmaf = 10
l1 = 1
l2 = 10
d = 365/sd(time)
temp = data[time,]$temp
lm.fit = lm(temp ~ time + I(time^2))
sigmaNoise = sd(lm.fit$residuals)

GPfit = gausspr(x = time,
                y = temp,
                kernel = GP.periodic.kernel,
                kpar = list(sigmaf = sigmaf, l1 = l1, l2 = l2, d=d),
                var = sigmaNoise^2)
meanPred = predict(GPfit,time) #Predict on train data
#plot(time,temp)
lines(time,meanPred, col = "purple")


##########################
###### Assignment 3 ######
##########################
###### Innit ######
library(AtmRay)
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])
set.seed(111); SelectTraining <- sample(1:dim(data)[1], size = 1000,
                         replace = FALSE)
```

```r
train = data[SelectTraining,]
test = data[-SelectTraining,]
###### 3.1 ######
GP.fit = gausspr(fraud ~ varWave+skewWave, data = train)

x1 <- seq(min(train$varWave),max(train$varWave),length=100)
x2 <- seq(min(train$skewWave),max(train$skewWave),length=100)
gridPoints <- meshgrid(x1, x2)
gridPoints <- cbind(c(gridPoints$x), c(gridPoints$y))
gridPoints <- data.frame(gridPoints)
names(gridPoints) <- names(data)[1:2]
probPreds <- predict(GP.fit, gridPoints, type="probabilities")

contour(x1,x2,matrix(probPreds[,2],100,byrow = TRUE), 20)

points(train[train$fraud == 1,"varWave"],
       train[train$fraud == 1,"skewWave"],
       col = "blue")

points(train[train$fraud == 0,"varWave"],
       train[train$fraud == 0,"skewWave"],
       col = "red")

con.mat.train = table(predict(GP.fit,train),train$fraud)
acc.train = sum(diag(con.mat.train)/sum(con.mat.train))

###### 3.2 ######
con.mat.test = table(predict(GP.fit,test),test$fraud)
acc.test = sum(diag(con.mat.test)/sum(con.mat.test))

###### 3.3 ######
GP.fit.all = gausspr(fraud ~ ., data = train)
con.mat.test.all = table(predict(GP.fit.all,test),test$fraud)
acc.test.all = sum(diag(con.mat.test.all)/sum(con.mat.test.all))
```