732A96: Lab 3
Advanced Machine Learning

Carles Sans Fuentes

October 7, 2017

Assignment

Question 1

Implementing Gaussian process regression from scratch. This first exercise will have you writing your own code for the Gaussian process regression model:

$$y = f(x) + \epsilon, \epsilon \sim N(0, \sigma_n^2), with$$

$$f(x) \sim GP[0, k(x, x')]$$

. When it comes to the posterior distribution for f, I strongly suggest that you implement Algorithm 2.1 on page 19 of Rasmussen and Williams (RW) book. That algorithm uses the Cholesky decomposition (chol() in R) to attain numerical stability. Here is what you need to do:

a , b

Question a: Write your own code for simulating from the posterior distribution of f(x) using the squared exponential kernel. The function (name it posteriorGP) should return vectors with the posterior mean and variance of f, both evaluated at a set of x-values (x*). You can assume that the prior mean of f is zero for all x.

Question b: Now let the prior hyperparameters be $\sigma_f = 1$,$l = 0.3$. Update this prior with a single observation: $(x, y) = (0.4, 0.719)$. Assume that the noise standard deviation is known to be $\sigma_n = 0.1$. Plot the posterior mean of f over the interval$[-1,1]$. Plot also 95% probability (pointwise) bands for f.

Answer:

The function GaussianKernel() is the the squared exponential kernel. The SimGP() is the function related to the Gaussian Process. The prior has been updated as well as the other parameters. A plot with the mean and the confidence bands is shown below:

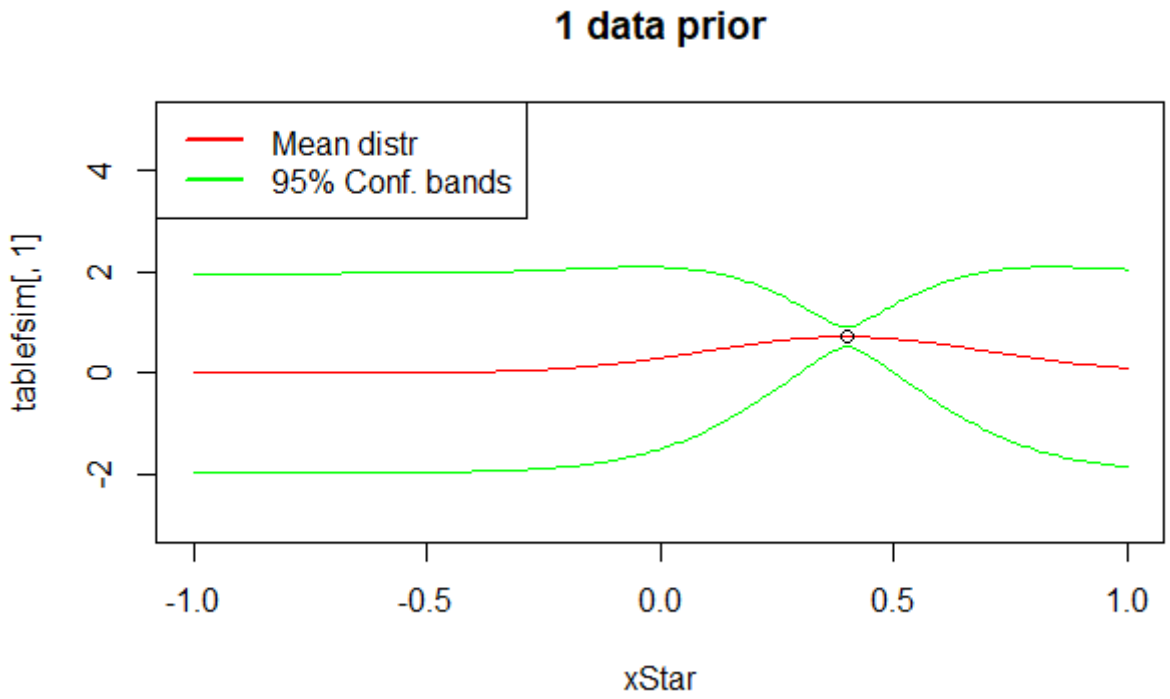

Figure 1: Posterior mean of f over the interval[−1,1]with 95% probability bands for f

c

Question: Update your posterior from 1b) with another observation: $(x, y) = (-0.6, -0.044)$. Plot the posterior mean of f over the interval $[-1, 1]$. Plot also 95% probability bands for f. [Hint: updating the posterior after one observation with a new observation gives the same result as updating the prior directly with the two observations. Bayes is beautiful!]

Answer: A plot with the mean and the confidence bands is shown below:

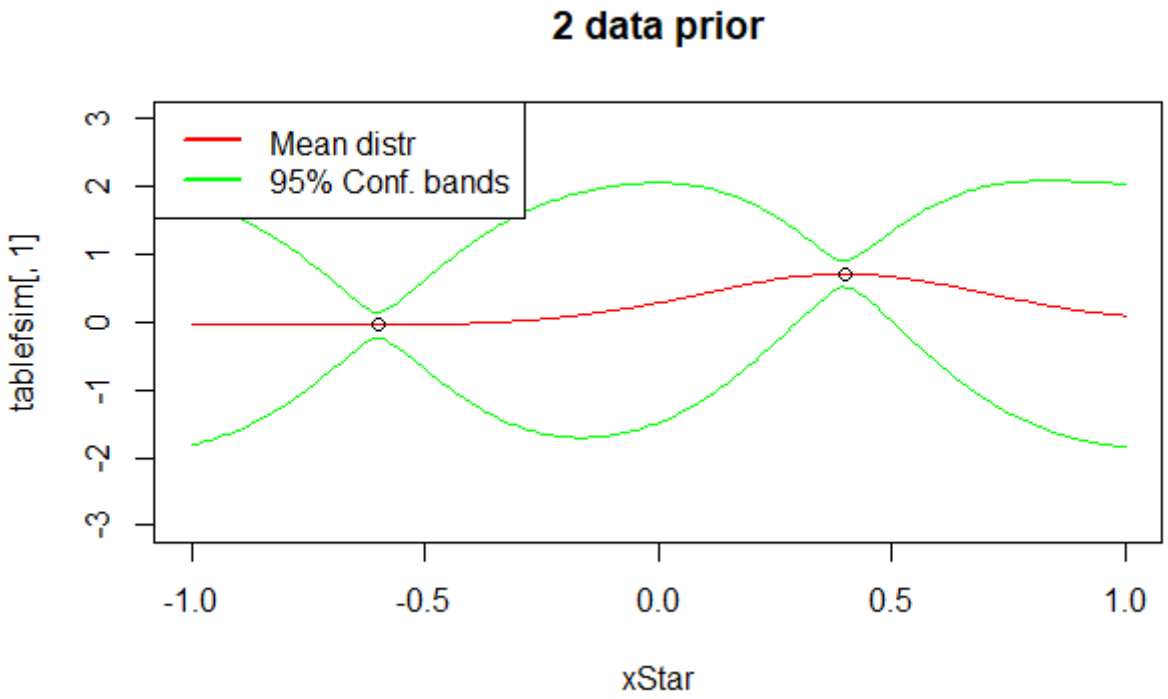

Figure 2: Posterior mean of f over the interval[−1,1]with 95% probability bands for f

d

Question: Compute the posterior distribution of f using all 5 data points in Table 1 below (note that the two previous observations are included in the table). Plot the posterior mean of f over the interval [−1,1]. Plot also 95% probability intervals for f.

Answer: A plot with the mean and the confidence bands is shown below:

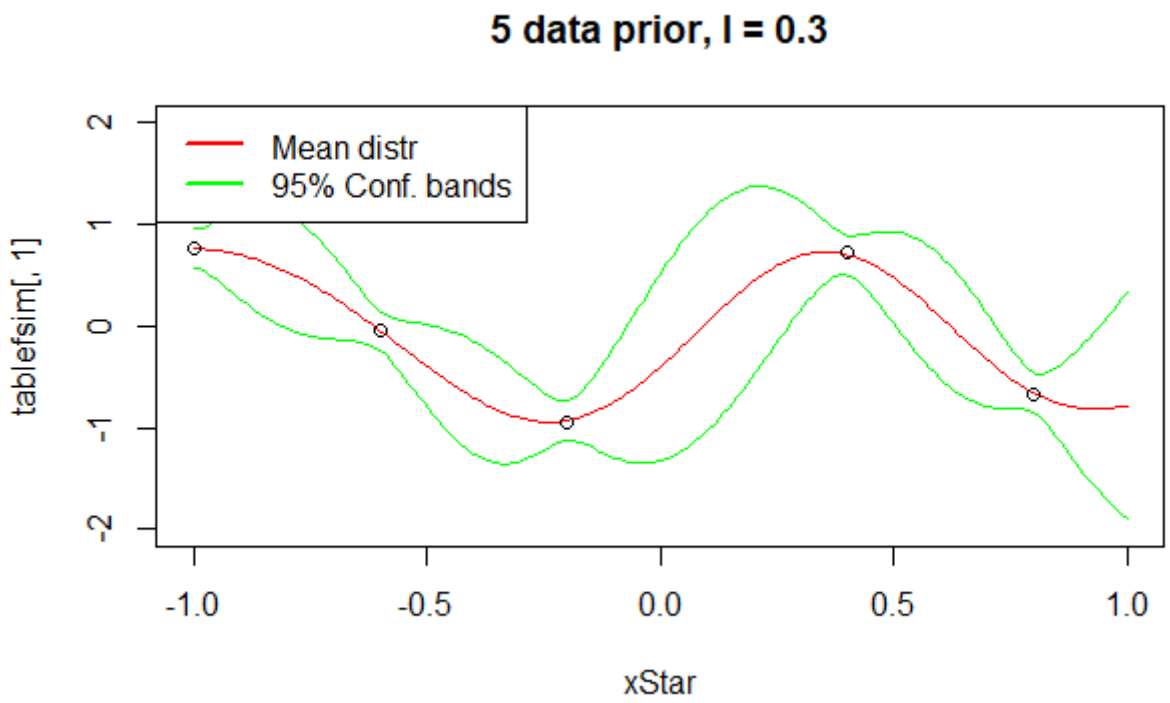

Figure 3: Posterior mean of f over the interval[−1,1]with 95% probability bands for f

e

Question: Repeat 1d), this time with the hyperparameters $\sigma_f = 1, l = 1$ Compare the results.

Answer: A plot with the mean and the confidence bands is shown below:

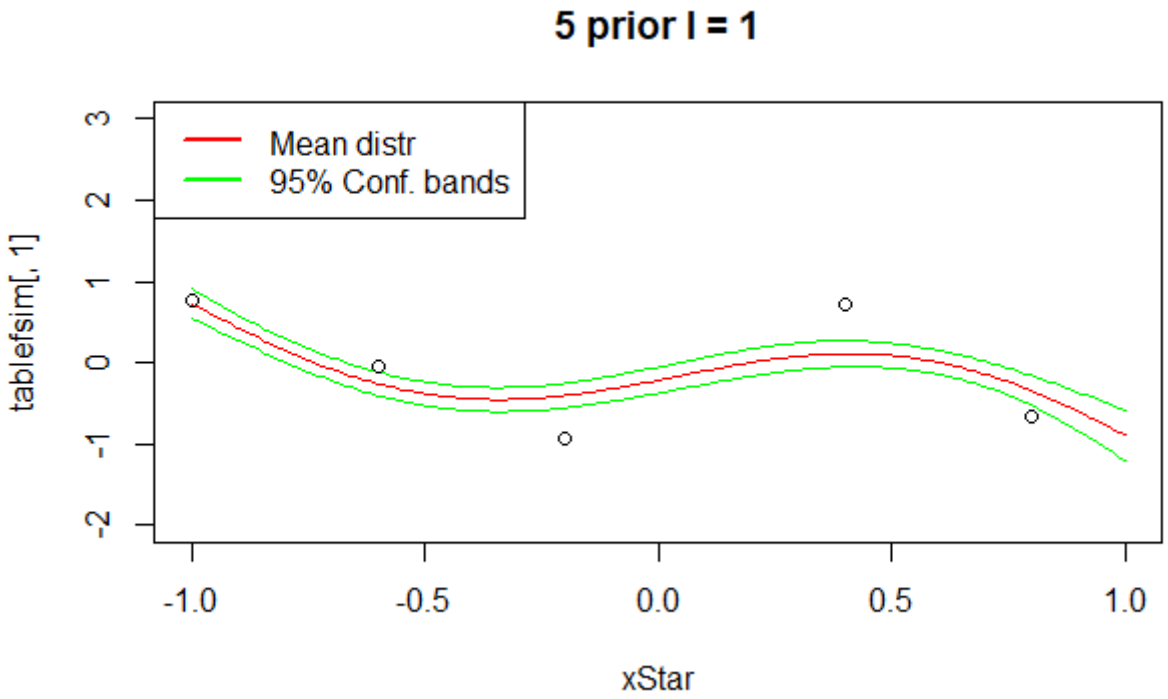

Figure 4: Posterior mean of f over the interval[−1,1]with 95% probability bands for f

It can be seen that the larger the l the more smoother our prediction will be, and the smaller our confidence bands get. This makes sense since the l parameter is a smoother hyper-parameter that the large it is the higher the covariance it is for a same σ_f, giving in this way more importance to σ_f. We are assuming by increasing l that we are more sure about our next distance training point. However, the choice of this l in our model is worse than having l equal to 3.

Question 2

Gaussian process regression on real data using the kernlab package. This exercise lets you explore the kernlab package on a data set of daily mean temperature in Stockholm (Tullinge) during the period January 1, 2010 - December 31, 2015. I have removed the leap year day February 29, 2012 to make your life simpler. A small version of it can be used in order to decrease computation time:

a

Question: Familiarize yourself with the following functions in kernlab, in particular the gausspr and kernelMatrix function. Do ?gausspr and read the input arguments and the output. Also,go through myKernLabDemo.R carefully; you will need to understand it. Now,define your own square exponential kernel function (with parameters l (ell) and σ_f (sigmaf)), evaluate it in the point $x = 1, x' = 2$, and use the kernelMatrix function to compute the covariance matrix $K(x, x_\star)$ for the input vectors $x = (1, 3, 4)^T and x_\star = (2, 3, 4)^T$

Answer: The results of the KernelMatrix and of my own square exponential kernel function called GaussianKernel2 (equal as in the previous activity but in a closure mode for convenience) has been displayed below.

```
> cov_matrix <- kernelMatrix(kernel =  GaussianKernel2, x=X, y = XStar)
> cov_matrix
An object of class "kernelMatrix"
          [,1]      [,2]       [,3]
[1,] 2.4261226 0.5413411 0.04443599
```

```
[2,] 2.4261226 4.0000000 2.42612264
[3,] 0.5413411 2.4261226 4.00000000

K<-GaussianKernel2(x=x1,y= x2)
> K
          [,1]
[1,] 2.426123
```

b

Question: Consider first the model:

$$temp = f(time) + \epsilon; \epsilon \sim N(0, \sigma_n^2)$$

$$f \sim GP(0, , k(time, time')$$

. Let σ_n^2 be the residual variance from a simple quadratic regression fit (using the lm() function in R). Estimate the above Gaussian process regression model using the squared exponential function from 2a) with $\sigma_f = 20$ and $l = 0.2$. Use the predict function to compute the posterior mean at every data point in the training datasets. Make a scatter plot of the data and superimpose the posterior mean of f as a curve (use $type =$ "l" in the plot function). Play around with different values on σ_f and l(no need to write this in the report though).

Answer: The plot is shown below:

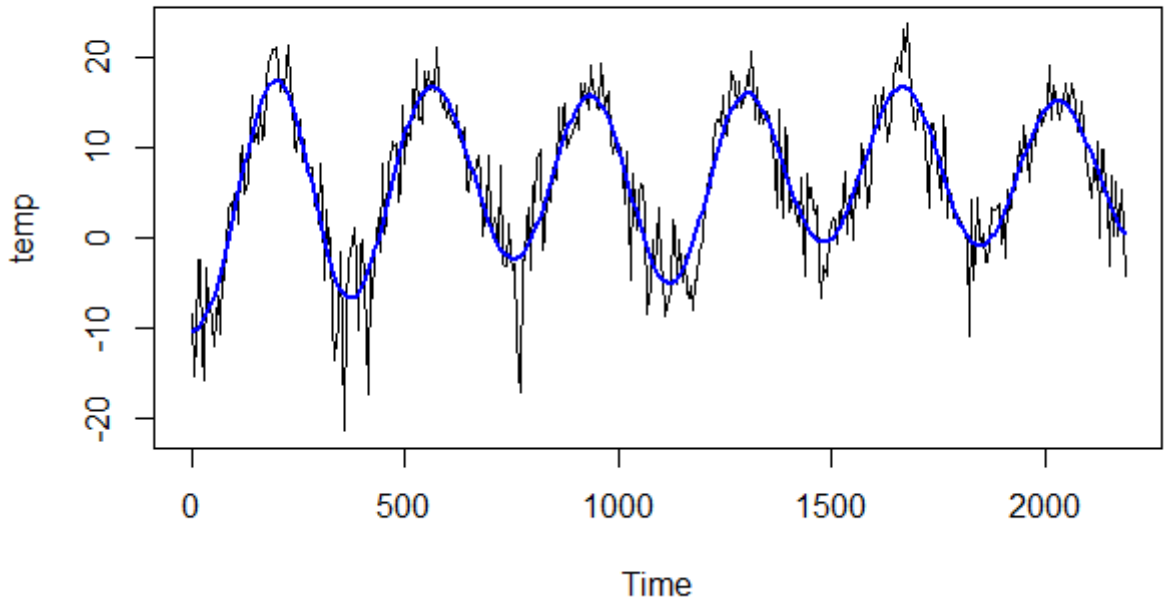

Figure 5: scatter plot of the data superimposed by the posterior mean of f as a curve

c

Question: Kernlab can compute the posterior variance of f, but I suspect a bug in the code (I get weird results). Do you own computations for the posterior variance of f (hint: Algorithm 2.1 in RW), and plot 95% (pointwise) posterior probability bands for f. Use $\sigma_f = 20$ and $l = 0.2$. Superimpose those bands on the figure with the posterior mean in 2b).

Answer: The plot is shown below:

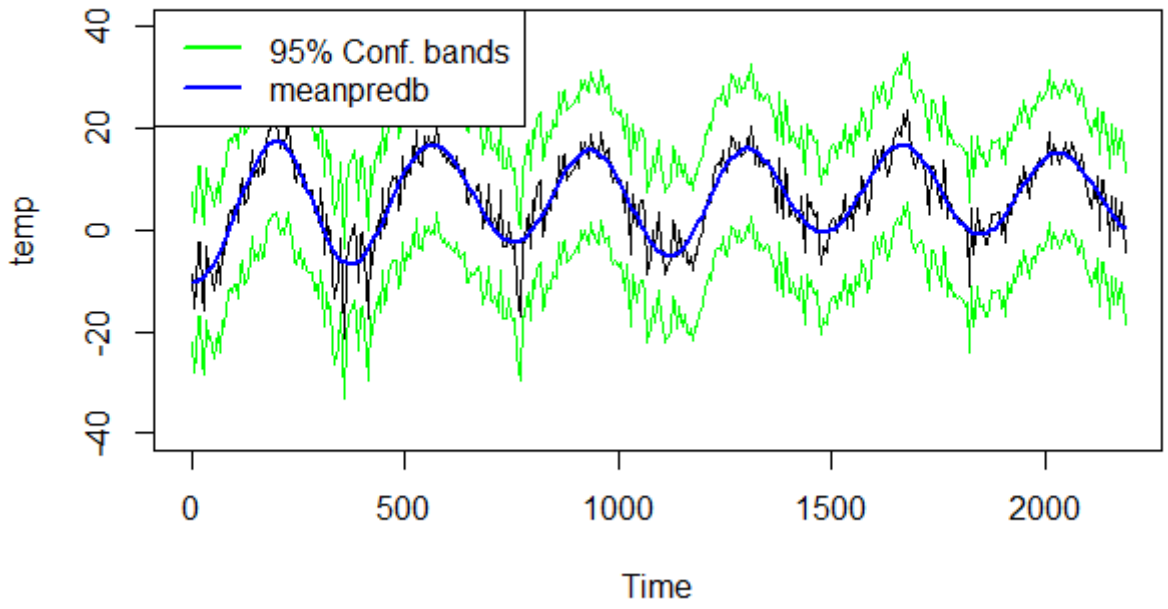

Figure 6: scatter plot of the data superimposed by the posterior mean of f as a curve with 95% probable confidence bands

d

Question:

$$temp = f(day) + \epsilon; \epsilon \sim N(0, \sigma_n^2)$$

$$f \sim GP(0, , k(day, day')$$

. Estimate the model using the squared exponential function from 2a) with $\sigma_f = 20$ and $l = 6 * 0.2 = 1.2$. (I multiplied ‘ by 6 compared to when you used time as input variable since kernlab automatically standardizes the data which makes the distance between points larger for day compared to time). Superimpose the posterior mean from this model on the fit (posterior mean) from the model with time using $\sigma_f = 20$,$l = 0.2$. Note that this plot should also have the time variable on the horizontal axis.

Answer: The plot is shown below:

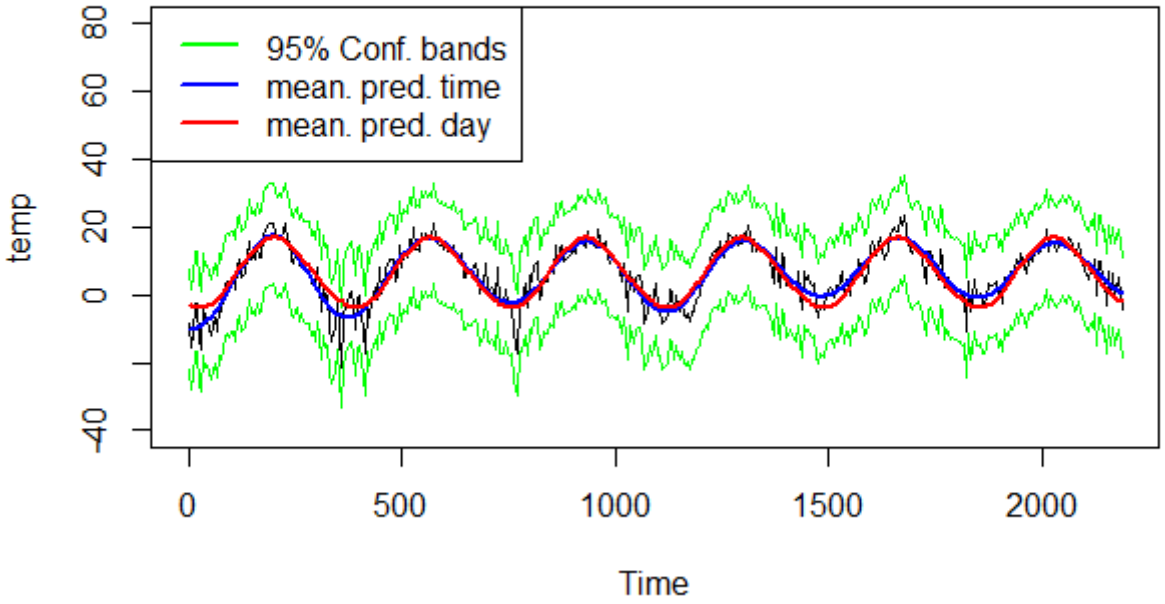

Figure 7: scatter plot of the data superimposed by the posterior mean of f as a curve with 95% probable confidence bands and new mean estimated by day model

The mean prediction for time (blue line) seems to take into account more larger variance in the data, whereas in the case of the mean of the prediction day (red line), it seems to be more stable even for larger residuals. These makes sense in the day model we are imposing to predict the same result year after year averaging all our data, whereas in the other observations are taken separately as they are linear and not cyclical over time. The pros of the and cons of the model are a possible prediction problem: overfitting vs having a stable good model.

e

Question: Now implement a generalization of the periodic kernel given in my slides from Lecture 2 of the GP topic (Slide 6) Compare the fit to the previous two models (with $\sigma_f = 20$,$l = 0.2$). Discuss the results.

Answer: The plot is shown below:

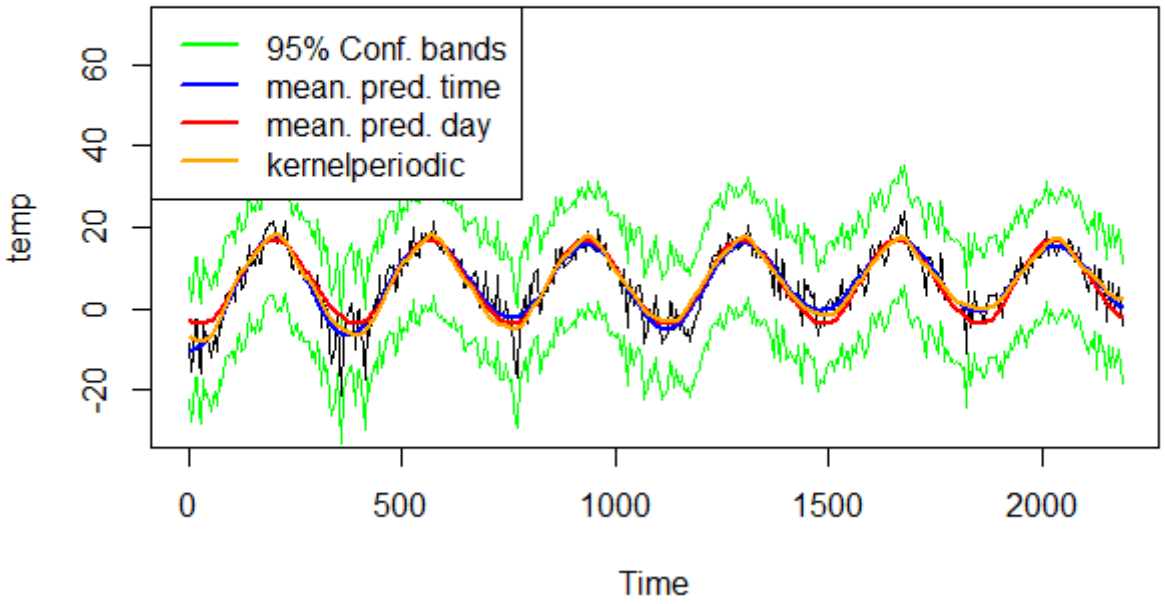

Figure 8: scatter plot of the data superimposed by the posterior mean of f as a curve with 95% probable confidence bands, new mean estimated by day model and new mean by the last periodic kernel

Having two l components enables to get a controlled stable mean model since it weights the seasonal and the every year different component. All in all, I think the (orange line) kernel periodic is better model since it combines both of the previous models in order to get a more stable prediction controlling just the parameters.

Question 3

Download the banknote fraud data:

a

Question: Use kernlab to fit a Gaussian process classification model for fraud on the training data, using kernlab. Use kernlab's the default kernel and hyperparameters. Start with using only the first two covariates varWave and skewWave in the model. (1) Plot contours of the prediction probabilities over a suitable grid of values for varWave and skewWave. Overlay the training data for $fraud = 1$ (as blue points) and $fraud = 0$ (as red points). You can take a lot of code for this from my KernLabDemo.R. (2)Compute the confusion matrix for the classifier and its accuracy.

Answer: The Confusion Matrix and the accuracy can be seen below

```
> predictionfunction(data = data, rows = SelectTraining)
     true
pred   0   1
   0 512  24
   1  44 420
[1] 0.932
```

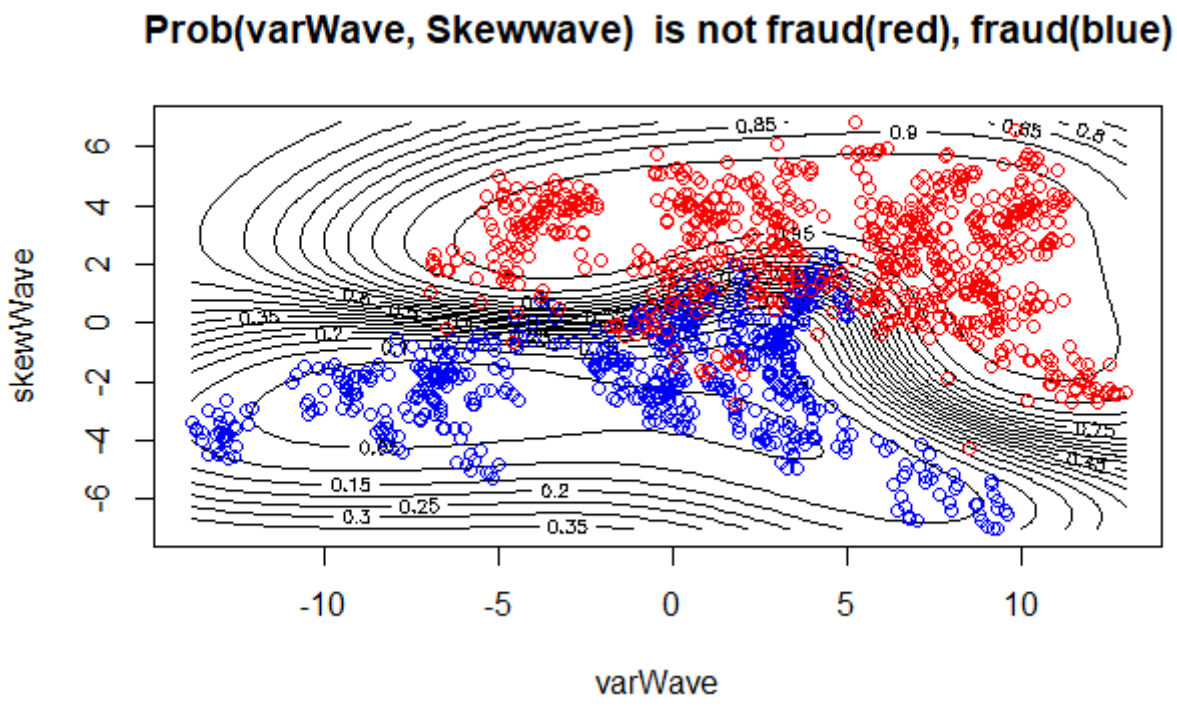

Figure 9: Plot contours of the prediction probabilities over a suitable grid of values for varWave and skewWave

b

Question: Using the estimated model from 3a), make predictions for the testset. Compute the accuracy

Answer: The Confusion Matrix and the accuracy can be seen below

```
> probPreds<-predictionfunction(data = data, rows = test)
     true
pred   0   1
   0 191   9
   1  15 157
[1] 0.9354839
```

c

Train a model using all four covariates. Make predictions on the test and compare the accuracy to the model with only two covariates.

Answer: The Confusion Matrix and the accuracy can be seen below

```
> predictionfunction(data = data, rows = SelectTraining)
     true
pred   0   1
   0 552   0
   1   4 444
[1] 0.996
> predictionfunction(data = data, rows = test)
     true
pred   0   1
   0 205   0
   1   1 166
[1] 0.9973118
```

With 4 covariates the prediction improves. This makes sense since we have more data to improve the prediction. Also these means that both covariates are important and significant to perform classification analysis.

Contributions

All results and comments presented have been developed and discussed together by the members of the group.

Appendix

Poisson regression-the MCMC way

```
1
2
3  ##Advanced ML lab 3
4  library("mvtnorm")
5
6  ##1a
7  x <-  c(0.4)
8  y <- c(0.719)
9  xStar <- seq(-1,1,length=200)
10 sigma_f<-1
11 l<-0.3
12 hyperParam<- c(sigma_f, l)
13 sigmaNoise<- 0.1
14 nSim <- 100
15
16 ###Calculating K from y and x, that are two vector inputs
17 GaussianKernel<- function(x1,x2, sigma_f =sigma_f, l=l){
18   n1<- length(x1)
19   n2<- length(x2)
20   K<- matrix(NA, n1, n2)
21   for(i in 1:n2){
22     K[,i]<-  sigma_f^2*exp(-0.5*( (x1-x2[i])/l)^2 )
23   }
24   return(K)
25 }
26
27 GaussianKernel(c(1,4), c(3,1), sigma = 2, l=1)
28 GaussianKernel(c(1,4), c(3,1), sigma = 2, l=2)
29 GaussianKernel(c(1,4), c(3,1), sigma = 2, l=3)
30 GaussianKernel(c(1,4), c(3,1), sigma = 2, l=4)
31
32
33
34 SimGP <- function(K,x,y, xStar, nSim, sigmaNoise, ...){
35   # Simulates nSim realizations (function) form a Gaussian process with mean m(x) and
       covariance K(x,x')
36   # over a grid of inputs (x)
37   n <- length(xStar)
38   #if (is.numeric(m)) meanVector <- rep(0,n) else meanVector <- m(xStar)
39
40
41   covMat    <- K(x, x,...)
42   covMatdiff2<- K(x, xStar,...)
43   covMatStar<- K(xStar, xStar,...)
44   L<-t(chol(covMat+ (sigmaNoise^2)*diag(dim(covMat)[2])))
45   alpha<- solve(t(L),solve(L,y))
46   fStar<- t(covMatdiff2)%*%alpha
47   v<-solve(L,covMatdiff2)
48   covfStar <- covMatStar-t(v)%*%v
49   return(list(mean=fStar, covfStar= diag(covfStar)))
50 }
51
52 ##b
53 fSim <- SimGP(y=y, xStar = xStar, x= x, K=GaussianKernel, nSim = nSim, sigma_f =sigma_f, l=l,
       sigmaNoise = sigmaNoise )
54
55 plot(fSim$mean)
56 plot(fSim$covfStar)
57 tablefsim<-cbind(mean = fSim$mean, low= fSim$mean-1.96*sqrt(fSim$covfStar),high= fSim$mean+1.96
       *sqrt(fSim$covfStar))
58 plot(x= xStar,y = tablefsim[,1], col = "red", type = "l", ylim = c(-3,5), main = "1 data prior"
       )
59 lines(x= xStar, y =tablefsim[,2], col ="green")
60 lines(x= xStar, y = tablefsim[,3], col = "green")
61 points(x = x, y = y)
62 legend("topleft", # places a legend at the appropriate place
63        c("Mean distr ", "95% Conf. bands"), # puts text in the legend
64        lty=c(1,1), # gives the legend appropriate symbols (lines)
65        lwd=c(2.5,2.5),col=c("Red","Green")) # gives the legend lines the correct color and
            width
66
67 ##c
68
69 x<-c(x,-0.6)
70 y<-c(y, -0.044)
71
72 fSim <- SimGP(y=y, xStar = xStar, x= x, K=GaussianKernel, nSim = nSim, sigma_f =sigma_f, l=l,
       sigmaNoise = sigmaNoise )
73
74 tablefsim<-cbind(mean = fSim$mean, low= fSim$mean-1.96*sqrt(fSim$covfStar),high= fSim$mean+1.96
       *sqrt(fSim$covfStar))
```

```
plot(x= xStar,y = tablefsim[,1], col = "red", type = "l", ylim = c(-3,3), main = "2 data prior")
lines(x= xStar, y =tablefsim[,2], col ="green")
lines(x= xStar, y = tablefsim[,3], col = "green")
points(x = x, y = y)
legend("topleft", # places a legend at the appropriate place
       c("Mean distr ", "95% Conf. bands"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Red","Green")) # gives the legend lines the correct color and width

###d
x<- c(-1.0,-0.6,-0.2,0.4,0.8)
y<- c(0.768,-0.044,-0.940,0.719, -0.664)

fSim <- SimGP(y=y, xStar = xStar, x= x, K=GaussianKernel, nSim = nSim, sigma_f =sigma_f, l=l, sigmaNoise = sigmaNoise )

tablefsim<-cbind(mean = fSim$mean, low= fSim$mean-1.96*sqrt(fSim$covfStar),high= fSim$mean+1.96*sqrt(fSim$covfStar))
plot(x= xStar,y = tablefsim[,1], col = "red", type = "l", main = "5 data prior, l = 0.3", ylim = c(-2,2))
lines(x= xStar, y =tablefsim[,2], col ="green")
points(x = x, y = y)
lines(x= xStar, y = tablefsim[,3], col = "green")
legend("topleft", # places a legend at the appropriate place
       c("Mean distr ", "95% Conf. bands"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Red","Green")) # gives the legend lines the correct color and width

###e

sigma_f<-1
l<-1
hyperParam<- c(sigma_f, l)

fSim <- SimGP(y=y, xStar = xStar, x= x, K=GaussianKernel, nSim = nSim, sigma_f =sigma_f, l=l, sigmaNoise = sigmaNoise )

tablefsim<-cbind(mean = fSim$mean, low= fSim$mean-1.96*sqrt(fSim$covfStar),high= fSim$mean+1.96*sqrt(fSim$covfStar))
plot(x= xStar,y = tablefsim[,1], col = "red", type = "l", ylim = c(-2,3), main = "5 prior l = 1")
lines(x= xStar, y =tablefsim[,2], col ="green")
lines(x= xStar, y = tablefsim[,3], col = "green")
points(x = x, y = y)
legend("topleft", # places a legend at the appropriate place
       c("Mean distr ", "95% Conf. bands"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Red","Green")) # gives the legend lines the correct color and width

###################################2
#
#
#
# plot(xStar, fSim[1,], type="l", ylim = c(-3,3))
# for (i in 2:dim(fSim)[1]) {
#   lines(xStar, fSim[i,], type="l")
# }
# lines(xStar,MeanFunc(xStar), col = "red", lwd = 3)
#
#
# # Plotting using manipulate package
# #install.packages("manipulate")
# library(manipulate)
#
# plotGPPrior <- function(sigma_f, l, nSim=20){
#   fSim <- SimGP(y=y, xStar = xStar, x= x, K=GaussianKernel, nSim = nSim, sigma_f =sigma_f, l=l, sigmaNoise = sigmaNoise )
#   plot(xStar, fSim[1,], type="l", ylim = c(-3,3), ylab="f(x)", xlab="x")
#   for (i in 2:nSim) {
#     lines(xStar, fSim[i,], type="l")
#   }
#   title(paste('length scale =',l,', sigmaf =',sigma_f))
# }
#
# manipulate(
#   plotGPPrior(sigma_f, l, nSim = 20),
#   sigma_f = slider(0, 2, step=0.1, initial = 1, label = "SigmaF"),
#   l = slider(0, 2, step=0.1, initial = 1, label = "Length scale, l")
# )
#

###2

```

```r
library(kernlab)
Data<- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/
    TempTullinge.csv", header=TRUE,
                sep = ";")

TData<-data.frame(time = 1:2190, date = Data$date,day =rep(1:365,6), temp = Data$temp)
GaussKernel<- function(sigmaF , ell){
  rval <- function(x,y, sigma_f = sigmaF, l = ell){
    n1 <- length(x)
    n2 <- length(y)
    K <- matrix(NA,n1,n2)
    for (i in 1:n2){
      K[,i] <- sigma_f^2*exp(-0.5*( (x-y[i])/l)^2 )
    }
    return(K)
  }

  class(rval) <- "kernel"
  return(rval)
}

SmallData<-  TData[seq(1, nrow(TData), 5), ]
sigma_f<-2
l<-1
X<-matrix(c(1,3,4), ncol = 1)
XStar<-matrix(c(2,3,4), ncol = 1)

GaussianKernel2 <- GaussKernel(sigmaF =  sigma_f, ell = l)
cov_matrix <- kernelMatrix(kernel =  GaussianKernel2, x=X, y = XStar)

x1<-1
x2<-2

K<-GaussianKernel2(x=x1,y= x2)
##b
sigma_f<-20
l <-0.2

x<-SmallData$time
x_prime<- seq(1,365,1)
GaussianKernel3<-GaussKernel(sigmaF = sigma_f, ell = l)
K<-kernelMatrix(GaussianKernel3, x,x_prime)

mylm<-lm(temp~time+ I(time^2), SmallData)
sigma_2_n<-var(mylm$residuals)


GPfit <- gausspr(temp ~ time, data = SmallData ,
                 kernel = GaussianKernel3,
                 kpar = list(sigma = sigma_f, ell = l), var = sigma_2_n)

myrange<-range(SmallData$time)
Xgrid<- myrange[1]:myrange[2]
meanPred<-predict(GPfit, SmallData)
length(meanPred)
plot(x = SmallData$time, SmallData$temp, type = "l",
     xlab = "Time", ylab = "temp")
lines(SmallData$time, meanPred, col="blue", lwd = 2)


##c

fSim <- SimGP(y=SmallData$temp, xStar = SmallData$time, x= SmallData$time,
              K=GaussianKernel3, nSim = nSim,
              sigma_f =sigma_f, l=l,
              sigmaNoise = sqrt(sigma_2_n ))


tablefsim<-cbind(mean = fSim$mean, low= fSim$mean-1.96*sqrt(fSim$covfStar),high= fSim$mean+1.96
    *sqrt(fSim$covfStar))
plot(x = SmallData$time, SmallData$temp, type = "l",
     xlab = "Time", ylab = "temp", ylim= c(-40,40))
lines(SmallData$time, meanPred, col="blue", lwd = 2)
lines(x= SmallData$time, y =tablefsim[,2], col ="green")
lines(x= SmallData$time, y = tablefsim[,3], col = "green")
legend("topleft", # places a legend at the appropriate place
       c( "95% Conf. bands", "meanpredb"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Green", "blue")) # gives the legend lines the correct color and
            width

####d


mylm<-lm(temp~day+ I(day^2), SmallData)
sigma_2_n<-var(mylm$residuals)
sigma_f<-20
```

```
l <- 1.2


GaussianKernel4<-GaussKernel(sigmaF = sigma_f, ell = l)


GPfit <- gausspr(temp ~ day, data = SmallData ,
                 kernel = GaussianKernel4,
                 kpar = list(sigma = sigma_f, ell = l), var = sigma_2_n)


meanPred2<-predict(GPfit, SmallData)

plot(x = SmallData$time, SmallData$temp, type = "l",
     xlab = "Time", ylab = "temp", ylim= c(-40,80))
lines(SmallData$time, meanPred, col="blue", lwd = 2)
lines(x= SmallData$time, y =tablefsim[,2], col ="green")
lines(x= SmallData$time, y = tablefsim[,3], col = "green")
lines(SmallData$time, meanPred2, col="red", lwd = 2)

legend("topleft", # places a legend at the appropriate place
       c( "95% Conf. bands", "mean. pred. time", "mean. pred. day"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Green", "blue", "red")) # gives the legend lines the correct color
            and width



##e

periodicKernel<- function(sigmaF, ell1, ell2, di){
  rval<-function(x1,x2, sigma_f =sigmaF, l_1=ell1, l_2=ell2, d=di){

      K<-  sigma_f^2*exp((-2*sin(pi*abs(x1-x2)/d)**2)/l_1^2)*exp((-1/2*abs(x1-x2)**2)/l_2^2)

  }
  class(rval) <- "kernel"
  return(rval)
}
sigma_f<-20
l_1<- 1
l_2<-10
d <- 365/sd(SmallData$time)
periodicKernel5<-periodicKernel(sigmaF=sigma_f, ell1=l_1, ell2=l_2, di=d)

GPfit <- gausspr(temp ~ time, data = SmallData ,
                 kernel = periodicKernel5,
                 kpar = list(sigma = sigma_f, ell = l), var = sigma_2_n)

meanPred3<-predict(GPfit, SmallData)

plot(x = SmallData$time, SmallData$temp, type = "l",
     xlab = "Time", ylab = "temp", ylim= c(-30,70))
lines(SmallData$time, meanPred, col="blue", lwd = 2)
lines(x= SmallData$time, y =tablefsim[,2], col ="green")
lines(x= SmallData$time, y = tablefsim[,3], col = "green")
lines(SmallData$time, meanPred2, col="red", lwd = 2)
lines(SmallData$time, meanPred3, col="orange", lwd = 2)
legend("topleft", # places a legend at the appropriate place
       c( "95% Conf. bands", "mean. pred. time", "mean. pred. day", "kernelperiodic"), # puts
           text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Green", "blue", "red", "orange")) # gives the legend lines the
           correct color and width



##3


data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/
    banknoteFraud.csv", 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)
model<-gausspr(fraud~varWave+skewWave, data = data[SelectTraining,])
##Part III
data<- data
predictionfunction<- function(data, rows){
  prediction<-predict(model,data[rows,])
  confusionMat<-table(pred=prediction, true = data[rows,5]) # confusion matrix
  Accuracy<-sum(diag(confusionMat))/sum(confusionMat)
  print(confusionMat)
  print(as.numeric(Accuracy))
```

```
318
319 }
320
321 predictionfunction(data = data, rows = SelectTraining)
322
323 ##contour
324 x1 <- seq(min(data[,"varWave"]),max(data["varWave"]),length=100)
325 x2 <- seq(min(data["skewWave"]),max(data["skewWave"]),length=100)
326 gridPoints <- meshgrid(x1, x2)
327 gridPoints <- cbind(c(gridPoints$x), c(gridPoints$y))
328
329 gridPoints <- data.frame(gridPoints)
330 names(gridPoints) <- c("varWave", "skewWave")
331 ######
332 probPreds <- predict(model, gridPoints, type="probabilities")
333 # Plotting for Prob(setosa)
334 contour(x2,x1,matrix(probPreds[,1],100), 20, xlab = "skewWave", ylab = "varWave", main = "Prob(
      varWave, Skewwave)  is not fraud(red), fraud(blue)")
335 points(data[data[,5]==1,"skewWave"],data[data[,5]==1,"varWave"],col="blue")
336 points(data[data[,5]==0,"skewWave"],data[data[,5]==0,"varWave"],col="red")
337
338 ###b
339
340 test<-setdiff( 1:dim(data)[1], SelectTraining)
341 probPreds<-predictionfunction(data = data, rows = test)
342
343
344
345
346
347 #c
348
349 model<-gausspr(fraud~., data = data[SelectTraining,])
350 #train
351 predictionfunction(data = data, rows = SelectTraining)
352 #test
353 predictionfunction(data = data, rows = test)
```