732A96 Advanced Machine Learning

LAB 4: Gaussian processes

Duc Tran
William Wiik
Mikael Montén
Johannes Hedström

STIMA
Department of Computer and Information Science
Linköpings universitet

2024-10-24

Contents

```r
# Q1 ##############################################################################
set.seed(12345)

library(ggplot2)
library(vctrs)
library(knitr)
library(pracma)
library(kernlab)
```

1 Question 1

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

$$y = f(x) + \epsilon \text{ with } \epsilon \sim N(0, \sigma_n^2) \text{ and } f \sim GP(0, k(x, x^{'}))$$

You must implement Algorithm 2.1 on page 19 of Rasmussen and Willams' book. The algorithm uses the Cholesky decomposition (chol in R) to attain numerical stability. Note that L in the algorithm is a lower triangular matrix, whereas the R function returns an upper triangular matrix. So, you need to transpose the output of the R function. In the algorithm, the notation $A\ b$ means the vector x that solves the equation $Ax = b$ (see p. xvii in the book). This is implemented in R with the help of the function solve. Here is what you need to do:

Rasmussen and Willams' algortihm:

input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_n^2 (noise level), $\mathbf{x}_*$ (test input)

$L := \text{cholesky}(K + \sigma_n^2 I)$

$\alpha := L^\top \backslash (L \backslash \mathbf{y})$

$\bar{f}_* := \mathbf{k}_*^\top \alpha$ } predictive mean

$\mathbf{v} := L \backslash \mathbf{k}_*$

$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ } predictive variance

$$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \alpha - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$$

return: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)

(1) Write your own code for simulating from the posterior distribution of f using the squared exponential kernel. The function (name it posteriorGP) should return a vector 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. The function should have the following inputs:

- **x**: Vector of training inputs
- **y**: Vector of training targets/outputs.
- **Xstar**: Vector of inputs where the posterior distribution is evaluated, i.e. X_*
- **sigmaNoise**: Noise standard deviation σ_n
- **k**: Covariance function or kernel. That is, the kernel should be a separate function.

(2) Now, let the prior hyperparameters be $\sigma_f = 1$ and $\ell = 0.3$. Update this prior with a single observation $(x, y) = (0.4, 0.719)$. ASSume that $\sigma_n = 0.1$. Plot the posterior mean of f over the interval $x \in [-1, 1]$. Plot also 95 % probability (pointwise) bands for f.

(3) Update your posterior from (2) with another observation: $(x, y) = (-0.6, -0.044)$. Plot the posterior mean of f over the interval $x \in [-1, 1]$. Plot also 95 % probability (pointwise) bands for f.

(4) Compute the posterior distribution of f using all the five data points in the table below (note that the two previous observations are included in the table). Plot the posterior mean of f over the interval $x \in [-1, 1]$. Plot also 95 % probability (pointwise) bands for f.

x	-1.000	-0.600	-0.20	0.400	0.800
y	0.768	-0.044	-0.94	0.719	-0.664

(5) Repeat (4), this time with hyperparameters of $\sigma_f = 1$ and $\ell = 1$. Compare the results

```r
# (1) & (2)

SquaredExpKernel <- 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, sigmaNoise, k, sigmaF=1, l=3) {

  # 2
  n <- length(y)
  K <- k(X, X, sigmaF = sigmaF, l = l)
  L <- t(chol(K + sigmaNoise^2 * diag(nrow(K))))
  alpha <- solve(t(L), solve(L, y))

  # 4
  k_star <-  k(X, XStar, sigmaF = sigmaF, l = l)
  f_star <- t(k_star) %*% alpha
  v <- solve(L, k_star)

  # 6
  var_f_star <-  k(XStar, XStar, sigmaF = sigmaF, l = l) - (t(v) %*% v)
  #logmar <- -0.5*(t(y)%*%a)-sum(log(diag(L)))-(n/2)*log(2*pi)
  # validation instead of logmar can be problematic as they are not iid
  # last values do not come from same pdf as training data. Can predict n+1 and train again to pred n+2
  return(list("mean" = f_star, "var" = diag(var_f_star)))

}
```

```
result <- posteriorGP(X = 0.4,
                      y = 0.719,
                      XStar = seq(-1,1,0.01),
                      sigmaNoise = 0.1,
                      k = SquaredExpKernel,
                      sigmaF = 1,
                      l = 0.3)
```

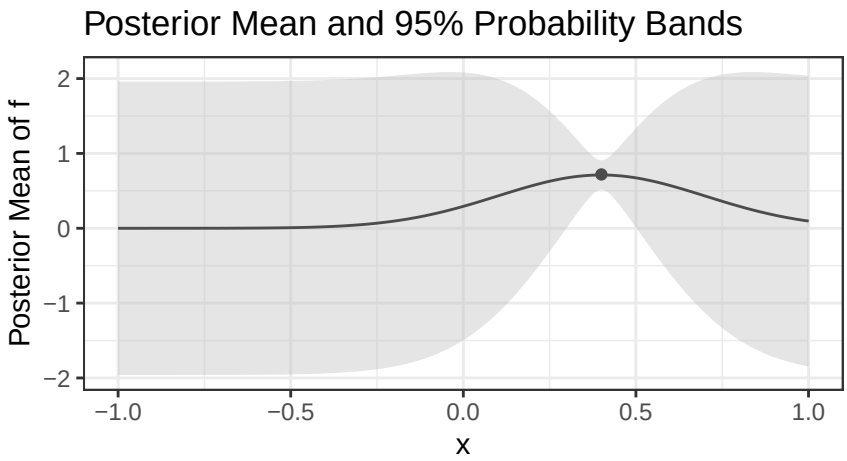


Figure 1: Posterior distribution with 1 observation

```
# (3)
result <- posteriorGP(X = c(0.4,-0.6), y = c(0.719, -0.044),
                      XStar = seq(-1,1,0.01), sigmaNoise = 0.1,
                      k = SquaredExpKernel, sigmaF = 1, l = 0.3)
```

```
# (4)

result <- posteriorGP(X = c(-1,-0.6,-0.2,0.4,0.8),
                      y = c(0.768, -0.044, -0.940,0.719, -0.664),
                      XStar = seq(-1,1,0.01), sigmaNoise = 0.1,
                      k = SquaredExpKernel, sigmaF = 1, l = 0.3)
```

```
# (5)

result <- posteriorGP(X = c(-1,-0.6,-0.2,0.4,0.8),
                      y = c(0.768, -0.044, -0.940,0.719, -0.664),
                      XStar = seq(-1,1,0.01), sigmaNoise = 0.1,
                      k = SquaredExpKernel, sigmaF = 1, l = 1)
```

Comparing the plot **(3)** and **(4)**, we observe that plot **(4)** has a smoother mean prediction, which is due to the higher ℓ value of 1 instead of 0.3. When ℓ is smaller then the predictions can vary much more between the

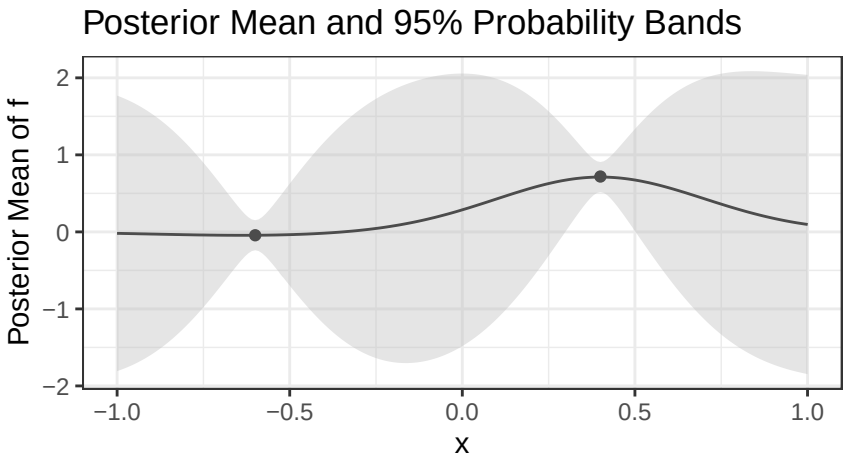


Figure 2: Posterior distribution with 2 observations

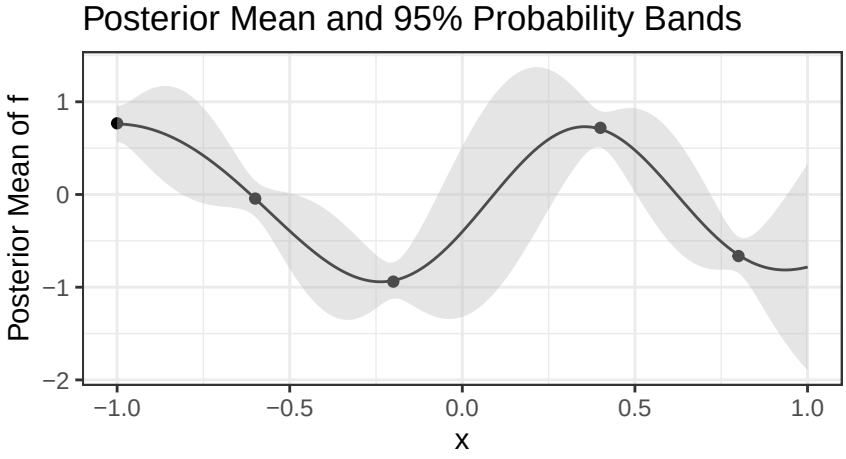


Figure 3: Posterior distribution with l=0.3

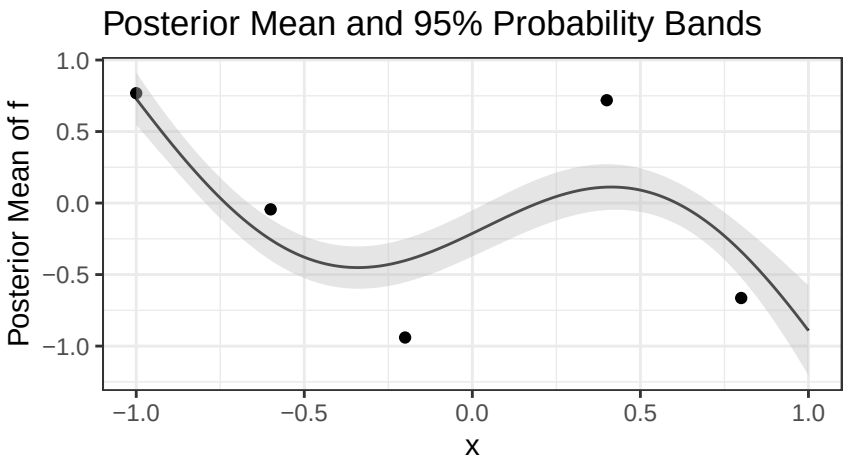


Figure 4: Posterior distribution with l=1

points, this is because ℓ controls the correlation to be high for points within ℓ distance of a point. This affects the values in the covariance matrix to be higher only when they are close and ℓ controls that distance.

From all the plots, we can observe that increasing the amount of training data results in a narrower confidence interval, and the confidence interval is particularly narrower around the training points.

TA comment: "When l is 1 the model is less flexible which can be seen in the more narrow probability bands.". While the length scale could have this effect on the width of the probability bands, the model flexibility and this width are not always directly linked. What can you say about the effect of l on how much the model predictions vary over the input interval?

2 Question 2

GP Regression with kernlab

In this exercise, you will work with the daily mean temperature in Stockholm (Tullinge) during the period January 1, 2010 - December 31, 2015. We have removed the leap year day February 29, 2012 to make things simpler. Create the variable time which records the day number since the start of the dataset (i.e., time= 1, 2, . . ., 365 x 6 = 2190). Also, create the variable day that records the day number since the start of each year (i.e., day= 1, 2, . . ., 365, 1, 2, . . ., 365). Estimating a GP on 2190 observations can take some time on slower computers, so let us subsample the data and use only every fifth observation. This means that your time and day variables are now time= 1, 6, 11, . . ., 2186 and day= 1, 6, 11, . . ., 361, 1, 6, 11, . . ., 361.

(1) Familiarize yourself with the functions gausspr and kernelMatrix in kernlab. Do ?gausspr and read the input arguments and the output. Also, go through the file KernLabDemo.R available on the course website. 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 $K(X, X_*)$ for the input vectors $X = (1, 3, 4)^T$ and $X_* = (2, 3, 4)^T$

(2) Consider first the following model:

$$temp = f(time) + \epsilon \text{ with } \epsilon \sim N(0, \sigma_n^2) \text{ and } f \sim GP(0, k(time, time^{'}))$$

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 gausspr function with the squared exponential function from (1) with $\sigma_f = 20$ and $\ell = 100$ (use the option scaled=FALSE in the gausspr function, otherwise these σ_f and ℓ values are not suitable). Use the predict function in R to compute the posterior mean at every data point in the training dataset. Make a scatterplot of the data and superimpose the posterior mean of f as a curve (use type="l" in the plot function). Plot also the 95 % probability (pointwise) bands for f. Play around with different values on σ_f and ℓ. (no need to write this in the report though)

(3) Repeat the previous exercise, but now use Algorithm 2.1 on page 19 of Rasmussen and Willams' book to compute the posterior mean and variance of f

(4) Consider now the following model:

$$temp = f(day) + \epsilon \text{ with } \epsilon \sim N(0, \sigma_n^2) \text{ and } f \sim GP(0, k(day, day^{'}))$$

Estimate the model using the gausspr function with the squared exponential function from (1) with $\sigma_f = 20$ and $\ell = 100$ (use the option scaled=FALSE in the gausspr function, otherwise these σ_f and ℓ values are not suitable). Superimpose the posterior mean from this model on the posterior mean from the model in (2). Note that this plot should also have the time variable on the horizontal axis. Compare the results of both models. What are the pros and cons of each model?

(5) Finally, implement the following extension of the squared exponential kernel with a periodic kernel (a.k.a. locally periodic kernel):

$$k(x, x') = \sigma_f^2 \exp\left\{ -\frac{2\sin^2(\pi |x - x'| / d)}{\ell_1^2} \right\} \exp\left\{ -\frac{1}{2} \frac{|x - x'|^2}{\ell_2^2} \right\}$$

Note that we have two different length scales in the kernel. Intuitively, ℓ_1 controls the correlation between two days in the same year, and ℓ_2 controls the correlation between the same day in different years. Estimate the

GP model using the time variable with this kernel and hyperparameters $\sigma_f = 20$, $\ell_1 = 1$, $\ell_2 = 100$ and d = 365. Use the gausspr function with the option scaled=FALSE, otherwise these σ_f , ℓ_1 and ℓ_2 values are not suitable. Compare the fit to the previous two models (with $\sigma_f = 20$ and $\ell = 100$). Discuss the results.

```
# Q2 ##############################################################################


data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullinge.c
                 header=TRUE, sep=";")


data$time <- 1:2190
data$day <- rep(1:365, 6)

# Subsample the data and (every fifth observation)
data_mini <- data[seq(1,2190, by = 5), ]
```

```
# (1)

# ?gausspr


SEkernel <- function(sigmaf = 1, ell = 1) {

  # Lecture 10 slide 4
  rval <- function(x, y = NULL) {

    r <- abs(x - y)
    return(sigmaf^2 * exp(- (r^2) / (2*ell^2)))

  }

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

kernel <- SEkernel(sigmaf = 1, ell = 1)
cat('Point evaluation = ',kernel(1,2))
```

```
## Point evaluation =  0.6065307
```

```
X <-  c(1,3,4)
X_star <- c(2,3,4)
K <- kernelMatrix(kernel = kernel, x = X, y = X_star)


knitr::kable(K, caption='Covariance matrix for (X,X*)')
```

Table 2: Covariance matrix for (X,X*)

0.6065307	0.1353353	0.0111090
0.6065307	1.0000000	0.6065307
0.1353353	0.6065307	1.0000000

The further apart two points are in x values, the less correlated their f values are.

```r
# (2)
# Simple quadratic regression fit
qrfit <- lm(temp ~ time + I(time**2), data = data_mini)
#summary(qrfit)


GPfit <- gausspr(x = data_mini$time,
                 y = data_mini$temp ,
                 kernel = SEkernel,
                 kpar = list(sigmaf = 20, ell = 100),
                 var = var(qrfit$residuals),
                 variance.model = TRUE,
                 scaled = FALSE)


pred_time_Q2 <- predict(GPfit, data_mini$time)
upper_pred_time <- pred_time_Q2 + 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")
lower_pred_time <- pred_time_Q2 - 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")

plot_data <- data.frame(
  time = data_mini$time,
  temp = data_mini$temp,
  posterior_mean = pred_time_Q2,
  upper = upper_pred_time,
  lower = lower_pred_time
)
```

```r
# (3)

result <- posteriorGP(X =  data_mini$time, y =  data_mini$temp,
                      XStar =  data_mini$time, sigmaNoise = sd(qrfit$residuals),
                      k = SquaredExpKernel, sigmaF = 20, l = 100)


# Plot data with upper and lower bounds
plot_data <- data.frame(time = data_mini$time,
                        temp = data_mini$temp,
                        posterior_mean = result$mean,
                        lower = result$mean - 1.96 * sqrt(result$var),
                        upper = result$mean + 1.96 * sqrt(result$var))
```

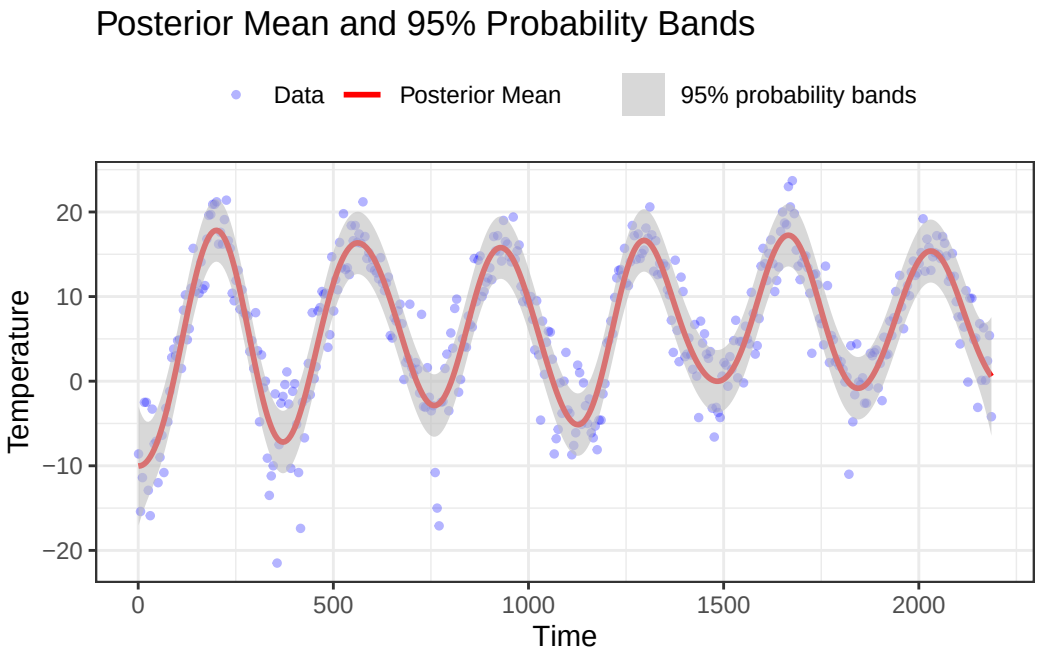


Figure 5: Posterior distribution with covariate time

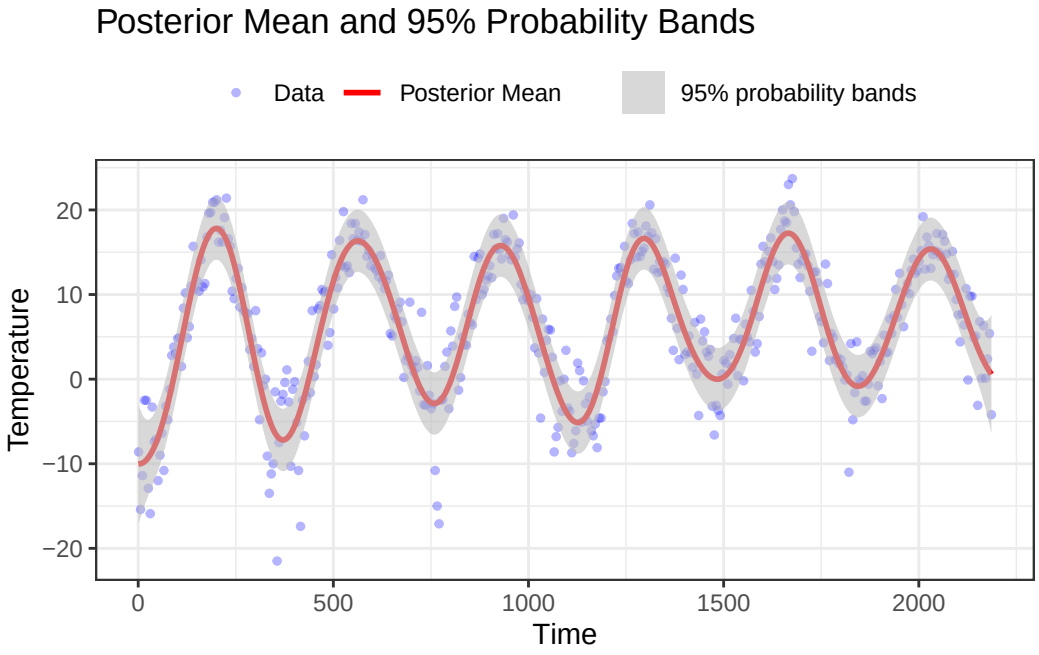


Figure 6: Posterior distribution with covariate days

```r
# (4)

qrfit <- lm(temp ~ day + I(day**2), data = data_mini)


GPfit <- gausspr(x = data_mini$day,
                 y = data_mini$temp ,
                 kernel = SEkernel,
                 kpar = list(sigmaf = 20, ell = 100),
                 var = var(qrfit$residuals),
                 variance.model = TRUE,
                 scaled = FALSE)

pred_day_Q4 <- predict(GPfit, data_mini$day)

plot_data <- data.frame(
  time = data_mini$time,
  temp = data_mini$temp,
  posterior_mean_day = pred_day_Q4,
  posterior_mean_time = pred_time_Q2,
  upper = upper_pred_time,
  lower = lower_pred_time
)
```

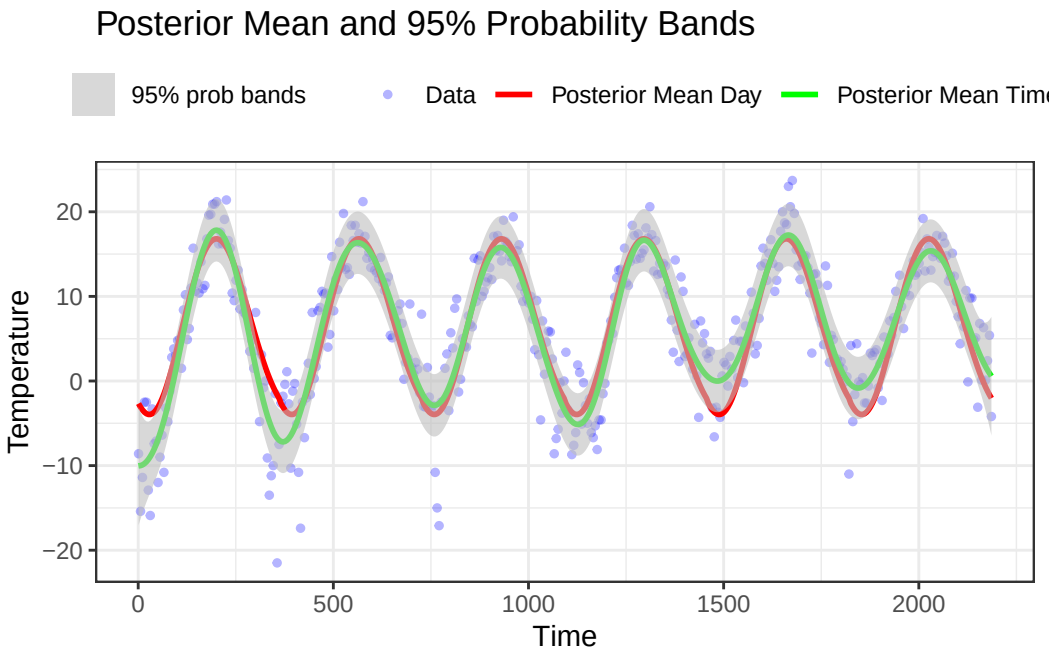


Figure 7: Posterior distribution with periodic kernel

The results from the two models are quite similar. The posterior means follows each other closely, however,

there is a notable difference: the model with $x = day$ have a repeating seasonal pattern, whereas the model with $x = time$ does not.

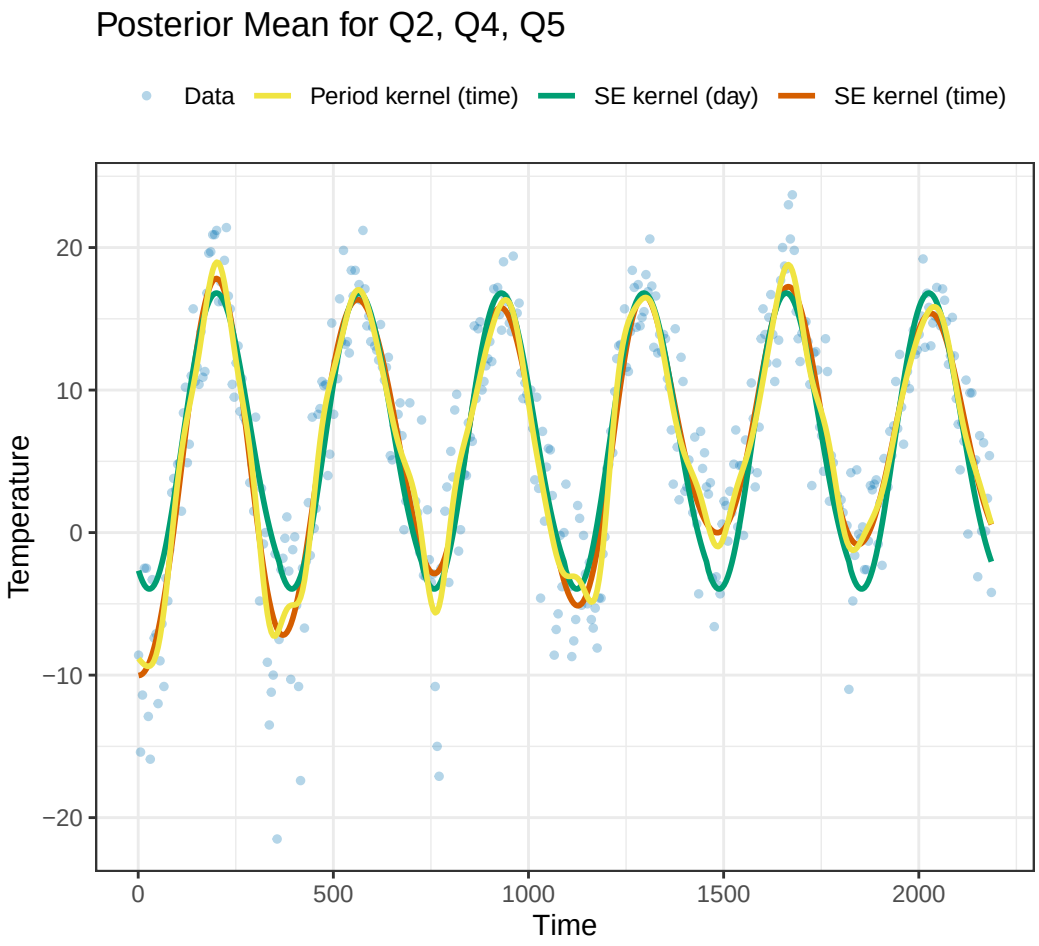
The pros with using a model with $x = day$ is that we can capture seasonal variations compared with the model $x = time$ where the model might struggle to capture the seasonal patterns as effectively. The day model is using 6 observations to each prediction instead of 1. That is because the Day variable is repeating the same values 6 times, the covariance between the values will then be repeated in the covariance matrix. January first one year will have a correlation of 1 with January the first in all the other years.

The cons with using a model with $x = day$ is that the model may not capture long-term trends as good as the model with $x = time$.

- **TA comment:** "The pros with using a model with x = day is that we can capture seasonal variations compared with the model x = time where the model might struggle to capture the seasonal patterns as effectively." But it still does not disregard seasonal patterns. Why is it that the first model (with x=day) have more "stable" predictions over the years?

```
# (5)

PeriodicKernel <- function(sigmaf = 1, ell_1 = 1, ell_2 = 100, d) {

  rval <- function(x, y = NULL) {

    r <- abs(x - y)
    eq1 <- sigmaf^2 * exp(-(2*sin( (pi*r) / d)^2) / ell_1^2)
    eq2 <- exp(-0.5 * (r^2 / ell_2^2))
    return(eq1 * eq2)

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

qrfit <- lm(temp ~ time + I(time**2), data = data_mini)

GPfit <- gausspr(x = data_mini$time,
                 y = data_mini$temp ,
                 kernel = PeriodicKernel,
                 kpar = list(sigmaf = 20, ell_1 = 1, ell_2 = 100, d = 365),
                 var = var(qrfit$residuals),
                variance.model = TRUE,
                 scaled = FALSE)

pred_time_Q5 <- predict(GPfit, data_mini$time)
upper_pred_time <- pred_time_Q5 + 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")
lower_pred_time <- pred_time_Q5 - 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")

plot_data <- data.frame(
  time = data_mini$time,
  temp = data_mini$temp,
  posterior_mean_Q2 = pred_time_Q2,
```

```
  posterior_mean_Q4 = pred_day_Q4,
  posterior_mean_Q5 = pred_time_Q5
)
```



When using a periodic kernel the posterior mean is not as smooth as when using a square exponential kernel, fitting the data better.

- TA comment: "When using a periodic kernel the posterior mean is not as smooth as when using a square exponential kernel, fitting the data better.". What are the general benefits of using a periodic kernel?

The periodic kernel uses both the seasonal part which day utilizes and the long term trend part, which then becomes a more powerful kernel. The cost is that it might be more computational heavy.

3 Question 3

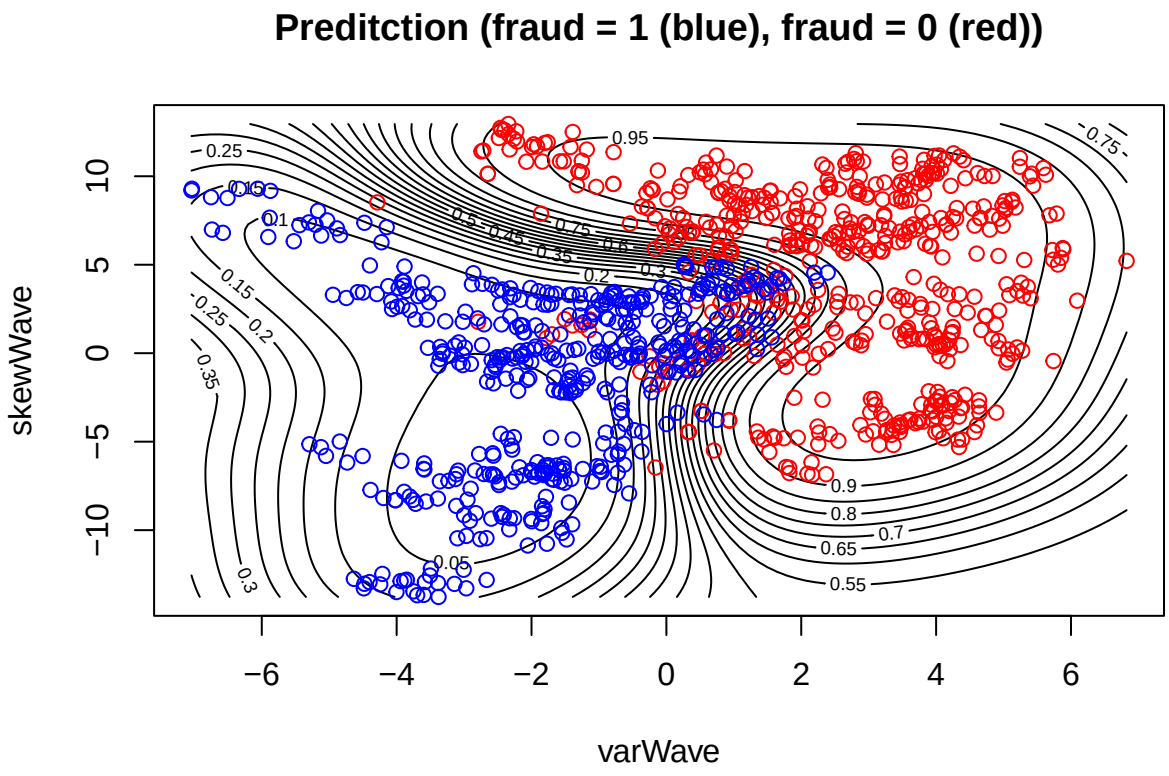
GP Classification with kernlab.

Choose 1000 observations as training data using the following command (i.e., use the vector SelectTraining to subset the training observations):

```
# Q3 ##############################################################################

data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/banknoteFraud.
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)
```

(1) Use the R package kernlab to fit a Gaussian process classification model for fraud on the training data. Use the default kernel and hyperparameters. Start using only the covariates varWave and skewWave in the model. 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 reuse code from the file KernLabDemo.R available on the course website. Compute the confusion matrix for the classifier and its accuracy.

```
# (1)

train <- data[SelectTraining, ]
test <- data[-SelectTraining, ]

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

## Using automatic sigma estimation (sigest) for RBF or laplace kernel
pred <- predict(model, 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(train)[1:2]
probPreds <- predict(model, gridPoints, type="probabilities")


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




```
# Confusion matrix
conf_mat <- table(pred, train$fraud)
conf_mat
```

```
##
## pred   0   1
##    0 503  18
##    1  41 438
```

```
# Accuracy train
acc <- sum(diag(conf_mat)) / sum(conf_mat)
acc
```

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

(2) Using the estimated model from (1), make predictions for the test set. Compute the accuracy.

```
# (2)
pred <- predict(model, test)
conf_mat <- table(pred, test$fraud)

# Accuracy test
acc <- sum(diag(conf_mat)) / sum(conf_mat)
acc
```

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

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

```
# (3)
model <- gausspr(fraud ~ varWave + skewWave + kurtWave + entropyWave, data = train)
```

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

```
pred <- predict(model, test)
conf_mat <- table(pred, test$fraud)

# Accuracy test
acc <- sum(diag(conf_mat)) / sum(conf_mat)
acc
```

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

When we used a model with only two covariates, we achieved an accuracy of 0.92, compared to 0.99 when using all four covariates. This suggests that adding `kurtWave` and `entropyWave` improves the models ability to make more accurate predictions.

4 Statement of contribution

We compared everyone results and decided to go for the code structure from William and Mikael. To ensure consistency, we compared each individual report. This allowed us to verify that everyone had implemented the correct functions with the appropriate arguments. In the discussion section, we collectively summarized the input from all people. We settled on the code and interpretations that resonated best with everyone. Each person contributed to the discussions by sharing insights and perspectives on the topic.

5 Appendix

```
# Q1 ##################################################################################
set.seed(12345)

library(ggplot2)
library(vctrs)
library(knitr)
library(pracma)
library(kernlab)

x <- c(-1,-0.6,-0.2,0.4,0.8)
y <- c(0.768, -0.044, -0.940,0.719, -0.664)

kable(t(data.frame(x,y)))



# (1) & (2)

SquaredExpKernel <- 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, sigmaNoise, k, sigmaF=1, l=3) {

  # 2
  n <- length(y)
  K <- k(X, X, sigmaF = sigmaF, l = l)
  L <- t(chol(K + sigmaNoise^2 * diag(nrow(K))))
  alpha <- solve(t(L), solve(L, y))

  # 4
```

```r
  k_star <-  k(X, XStar, sigmaF = sigmaF, l = l)
  f_star <- t(k_star) %*% alpha
  v <- solve(L, k_star)

  # 6
  var_f_star <-  k(XStar, XStar, sigmaF = sigmaF, l = l) - (t(v) %*% v)
  #logmar <- -0.5*(t(y)%*%a)-sum(log(diag(L)))-(n/2)*log(2*pi)
  # validation instead of logmar can be problematic as they are not iid
  # last values do not come from same pdf as training data. Can predict n+1 and train again to pred n+2
  return(list("mean" = f_star, "var" = diag(var_f_star)))

}



result <- posteriorGP(X = 0.4,
                      y = 0.719,
                      XStar = seq(-1,1,0.01),
                      sigmaNoise = 0.1,
                      k = SquaredExpKernel,
                      sigmaF = 1,
                      l = 0.3)



points_df <- data.frame(
  x = c(0.4),
  y = c( 0.719)
)

# Plot data with upper and lower bounds
data <- data.frame(x =  seq(-1, 1, 0.01),
                   mean = result$mean,
                   lower = result$mean - 1.96 * sqrt(result$var),
                   upper =  result$mean + 1.96 * sqrt(result$var))

ggplot(data, aes(x = x)) +
  geom_line(aes(y = mean), color = "black") +
  geom_point(data = points_df, aes(x = x, y = y)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "grey", alpha = 0.4) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "x",
       y = "Posterior Mean of f") +
  theme_bw()


# (3)
result <- posteriorGP(X = c(0.4,-0.6), y = c(0.719, -0.044),
                      XStar = seq(-1,1,0.01), sigmaNoise = 0.1,
```

```
                      k = SquaredExpKernel, sigmaF = 1, l = 0.3)


points_df <- data.frame(
  x = c(0.4,-0.6),
  y =  c(0.719, -0.044)
)


# Plot data with upper and lower bounds
data <- data.frame(x =  seq(-1, 1, 0.01),
                   mean = result$mean,
                   lower = result$mean - 1.96 * sqrt(result$var),
                   upper =  result$mean + 1.96 * sqrt(result$var))

ggplot(data, aes(x = x)) +
  geom_line(aes(y = mean), color = "black") +
  geom_point(data = points_df, aes(x = x, y = y)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "grey", alpha = 0.4) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "x",
       y = "Posterior Mean of f") +
  theme_bw()


# (4)

result <- posteriorGP(X = c(-1,-0.6,-0.2,0.4,0.8),
                      y = c(0.768, -0.044, -0.940,0.719, -0.664),
                      XStar = seq(-1,1,0.01), sigmaNoise = 0.1,
                      k = SquaredExpKernel, sigmaF = 1, l = 0.3)



points_df <- data.frame(
  x = c(-1, -0.6, -0.2, 0.4, 0.8),
  y = c(0.768, -0.044, -0.940, 0.719, -0.664)
)

# Plot data with upper and lower bounds
data <- data.frame(x =  seq(-1, 1, 0.01),
                   mean = result$mean,
                   lower = result$mean - 1.96 * sqrt(result$var), # change to
                  # sqrt(results$var + sigma2^2) for prob band for Y
                   upper =  result$mean + 1.96 * sqrt(result$var))

ggplot(data, aes(x = x)) +
  geom_line(aes(y = mean), color = "black") +
  geom_point(data = points_df, aes(x = x, y = y)) +
```

```
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "grey", alpha = 0.4) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "x",
       y = "Posterior Mean of f") +
  theme_bw()


# loop this and add X <- c(X,0.01*which.max(sqrt(result$var+2**2)))
# and  its Y value, this way chooses point where we are most the uncertain

# (5)

result <- posteriorGP(X = c(-1,-0.6,-0.2,0.4,0.8),
                      y = c(0.768, -0.044, -0.940,0.719, -0.664),
                      XStar = seq(-1,1,0.01), sigmaNoise = 0.1,
                      k = SquaredExpKernel, sigmaF = 1, l = 1)




# Plot data with upper and lower bounds
data <- data.frame(x =  seq(-1, 1, 0.01),
                   mean = result$mean,
                   lower = result$mean - 1.96 * sqrt(result$var),
                   upper =  result$mean + 1.96 * sqrt(result$var))

ggplot(data, aes(x = x)) +
  geom_line(aes(y = mean), color = "black") +
  geom_point(data = points_df, aes(x = x, y = y)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "grey", alpha = 0.4) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "x",
       y = "Posterior Mean of f") +
  theme_bw()


# Q2 ##############################################################################


data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullinge.c
                 header=TRUE, sep=";")


data$time <- 1:2190
data$day <- rep(1:365, 6)

# Subsample the data and (every fifth observation)
data_mini <- data[seq(1,2190, by = 5), ]
```

```r
# (1)

# ?gausspr


SEkernel <- function(sigmaf = 1, ell = 1) {

  # Lecture 10 slide 4
  rval <- function(x, y = NULL) {

    r <- abs(x - y)
    return(sigmaf^2 * exp(- (r^2) / (2*ell^2)))

  }

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

kernel <- SEkernel(sigmaf = 1, ell = 1)
cat('Point evaluation = ',kernel(1,2))
X <-  c(1,3,4)
X_star <- c(2,3,4)
K <- kernelMatrix(kernel = kernel, x = X, y = X_star)


knitr::kable(K, caption='Covariance matrix for (X,X*)')

# (2)
# Simple quadratic regression fit
qrfit <- lm(temp ~ time + I(time**2), data = data_mini)
#summary(qrfit)


GPfit <- gausspr(x = data_mini$time,
                 y = data_mini$temp ,
                 kernel = SEkernel,
                 kpar = list(sigmaf = 20, ell = 100),
                 var = var(qrfit$residuals),
                 variance.model = TRUE,
                 scaled = FALSE)


pred_time_Q2 <- predict(GPfit, data_mini$time)
upper_pred_time <- pred_time_Q2 + 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")
lower_pred_time <- pred_time_Q2 - 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")
```

```r
plot_data <- data.frame(
  time = data_mini$time,
  temp = data_mini$temp,
  posterior_mean = pred_time_Q2,
  upper = upper_pred_time,
  lower = lower_pred_time
)



ggplot(plot_data, aes(x = time)) +
  geom_point(aes(y = temp, color = "Data"), size = 1,alpha = 0.3) +
  geom_line(aes(y = posterior_mean, color = "Posterior Mean"), size = 1) +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = "95% probability bands"), alpha = 0.6) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "Time", y = "Temperature", color = "Legend", fill = "Legend") +
  scale_color_manual(values = c("Data" = "blue", "Posterior Mean" = "red"), name = NULL) +
  scale_fill_manual(values = c("95% probability bands" = "grey"), name = NULL) +
  theme_bw() +
  theme(legend.position = "top")



# (3)

result <- posteriorGP(X =  data_mini$time, y =  data_mini$temp,
                      XStar =  data_mini$time, sigmaNoise = sd(qrfit$residuals),
                      k = SquaredExpKernel, sigmaF = 20, l = 100)


# Plot data with upper and lower bounds
plot_data <- data.frame(time = data_mini$time,
                        temp = data_mini$temp,
                        posterior_mean = result$mean,
                        lower = result$mean - 1.96 * sqrt(result$var),
                        upper = result$mean + 1.96 * sqrt(result$var))




ggplot(plot_data, aes(x = time)) +
  geom_point(aes(y = temp, color = "Data"), size = 1,alpha = 0.3) +
  geom_line(aes(y = posterior_mean, color = "Posterior Mean"), size = 1) +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = "95% probability bands"), alpha = 0.6) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "Time", y = "Temperature", color = "Legend", fill = "Legend") +
  scale_color_manual(values = c("Data" = "blue", "Posterior Mean" = "red"), name = NULL) +
```

```r
  scale_fill_manual(values = c("95% probability bands" = "grey"), name = NULL) +
  theme_bw() +
  theme(legend.position = "top")



# (4)

qrfit <- lm(temp ~ day + I(day**2), data = data_mini)


GPfit <- gausspr(x = data_mini$day,
                 y = data_mini$temp ,
                 kernel = SEkernel,
                 kpar = list(sigmaf = 20, ell = 100),
                 var = var(qrfit$residuals),
                 variance.model = TRUE,
                 scaled = FALSE)

pred_day_Q4 <- predict(GPfit, data_mini$day)

plot_data <- data.frame(
  time = data_mini$time,
  temp = data_mini$temp,
  posterior_mean_day = pred_day_Q4,
  posterior_mean_time = pred_time_Q2,
  upper = upper_pred_time,
  lower = lower_pred_time
)


ggplot(plot_data, aes(x = time)) +
  geom_point(aes(y = temp, color = "Data"), size = 1,alpha = 0.3) +
  geom_line(aes(y = posterior_mean_day, color = "Posterior Mean Day"), size = 1) +
  geom_line(aes(y = posterior_mean_time, color = "Posterior Mean Time"), size = 1) +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = "95% prob bands"), alpha = 0.6) +
  labs(title = "Posterior Mean and 95% Probability Bands",
       x = "Time", y = "Temperature", color = "Legend", fill = "Legend") +
  scale_color_manual(values = c("Data" = "blue",
                                "Posterior Mean Day" = "red",
                                "Posterior Mean Time" = "green"), name = NULL) +
  scale_fill_manual(values = c("95% prob bands" = "grey"), name = NULL) +
  theme_bw() +
  theme(legend.position = "top")



# (5)
```

```r
PeriodicKernel <- function(sigmaf = 1, ell_1 = 1, ell_2 = 100, d) {

  rval <- function(x, y = NULL) {

    r <- abs(x - y)
    eq1 <- sigmaf^2 * exp(-(2*sin( (pi*r) / d)^2) / ell_1^2)
    eq2 <- exp(-0.5 * (r^2 / ell_2^2))
    return(eq1 * eq2)

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

qrfit <- lm(temp ~ time + I(time**2), data = data_mini)

GPfit <- gausspr(x = data_mini$time,
                 y = data_mini$temp ,
                 kernel = PeriodicKernel,
                 kpar = list(sigmaf = 20, ell_1 = 1, ell_2 = 100, d = 365),
                 var = var(qrfit$residuals),
                variance.model = TRUE,
                 scaled = FALSE)

pred_time_Q5 <- predict(GPfit, data_mini$time)
upper_pred_time <- pred_time_Q5 + 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")
lower_pred_time <- pred_time_Q5 - 1.96 * predict(GPfit, data_mini$time, type ="sdeviation")

plot_data <- data.frame(
  time = data_mini$time,
  temp = data_mini$temp,
  posterior_mean_Q2 = pred_time_Q2,
  posterior_mean_Q4 = pred_day_Q4,
  posterior_mean_Q5 = pred_time_Q5
)

ggplot(plot_data, aes(x = time)) +
  geom_point(aes(y = temp, color = "Data"), size = 1,alpha = 0.3) +
  geom_line(aes(y = posterior_mean_Q2, color = "SE kernel (time)"), size = 1) +
  geom_line(aes(y = posterior_mean_Q4, color = "SE kernel (day)"), size = 1) +
  geom_line(aes(y = posterior_mean_Q5, color = "Period kernel (time)"), size = 1) +
  labs(title = "Posterior Mean for Q2, Q4, Q5",
       x = "Time", y = "Temperature", color = "Legend", fill = "Legend") +
   scale_color_manual(values = c("Data" = "#0072B2",  # Blue
                                 "SE kernel (time)" = "#D55E00",
                                 "SE kernel (day)" = "#009E73",
                                 "Period kernel (time)" = "#F0E442"),
                      name = NULL) +
  theme_bw() +
```

```r
  theme(legend.position = "top")


# Q3 ##############################################################################

data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/banknoteFraud.
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)


# (1)

train <- data[SelectTraining, ]
test <- data[-SelectTraining, ]

model <- gausspr(fraud ~ varWave + skewWave, data = train)
pred <- predict(model, 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(train)[1:2]
probPreds <- predict(model, gridPoints, type="probabilities")


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



# Confusion matrix
conf_mat <- table(pred, train$fraud)
conf_mat
# Accuracy train
acc <- sum(diag(conf_mat)) / sum(conf_mat)
acc

# (2)
pred <- predict(model, test)
conf_mat <- table(pred, test$fraud)
```

```r
# Accuracy test
acc <- sum(diag(conf_mat)) / sum(conf_mat)
acc

# (3)
model <- gausspr(fraud ~ varWave + skewWave + kurtWave + entropyWave, data = train)

pred <- predict(model, test)
conf_mat <- table(pred, test$fraud)

# Accuracy test
acc <- sum(diag(conf_mat)) / sum(conf_mat)
acc
```