TDDE15 Lab4 Gaussian Processes

Hannes Bengtsson

2023-10-16

Task 1 Implementing GP Regression

Description 1.1 - Implementing Algorithm 2.1

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 $\mathbf{A/b}$ means the vector $\mathbf{x}$ that solves the equation $\mathbf{Ax = b}$ (see p. xvii in the book). This is implemented in R with the help of the function *solve*.

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 (see the file GaussianProcesses.R on the course web page).

Solution 1.1 - Implementing Algorithm 2.1

Please see the code below.

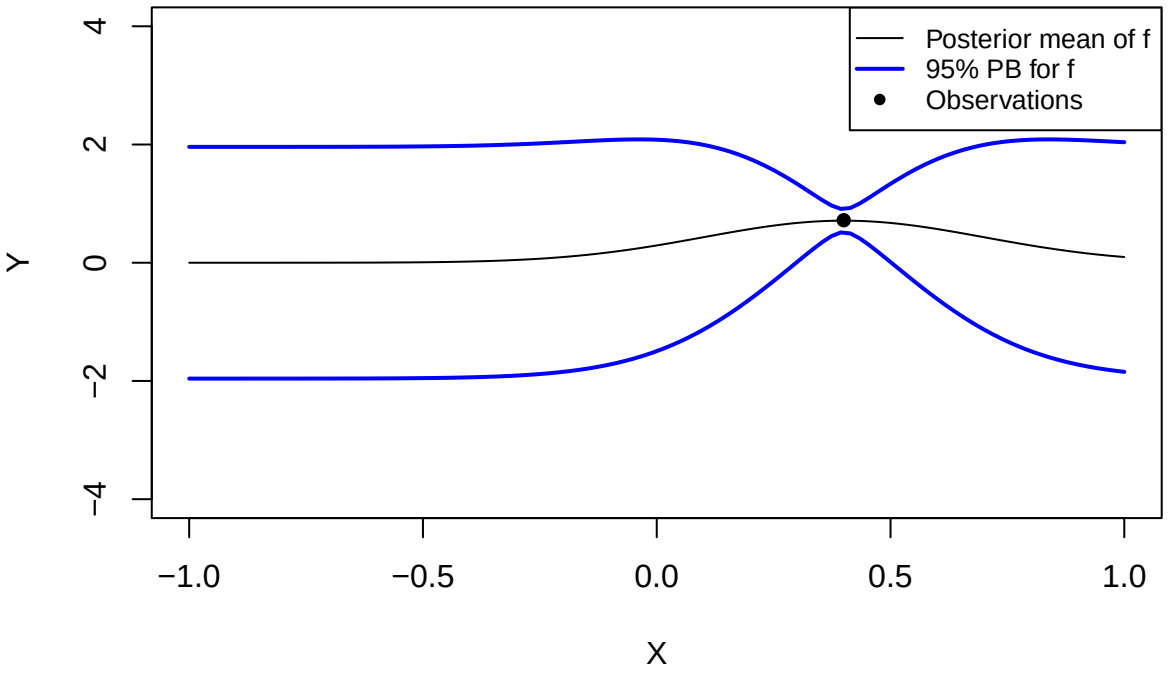
```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha
```

Description 1.2 - One Observation Mean and Probability Bands for f

Now, let the prior hyperparameters be $\sigma_f = 1$ and $l = 0.3$. 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.

Solution 1.2 - One Observation Mean and Probability Bands for f

With hyper parameters $\sigma_f = 1$ and $l = 0.3$ we get the following posterior mean for f over the interval $x \in [-1, 1]$, the black line. Lines in blue illustrates 95% probability bands for f and the black dot illustrates our observation $(x, y) = (0.4, 0.719)$.

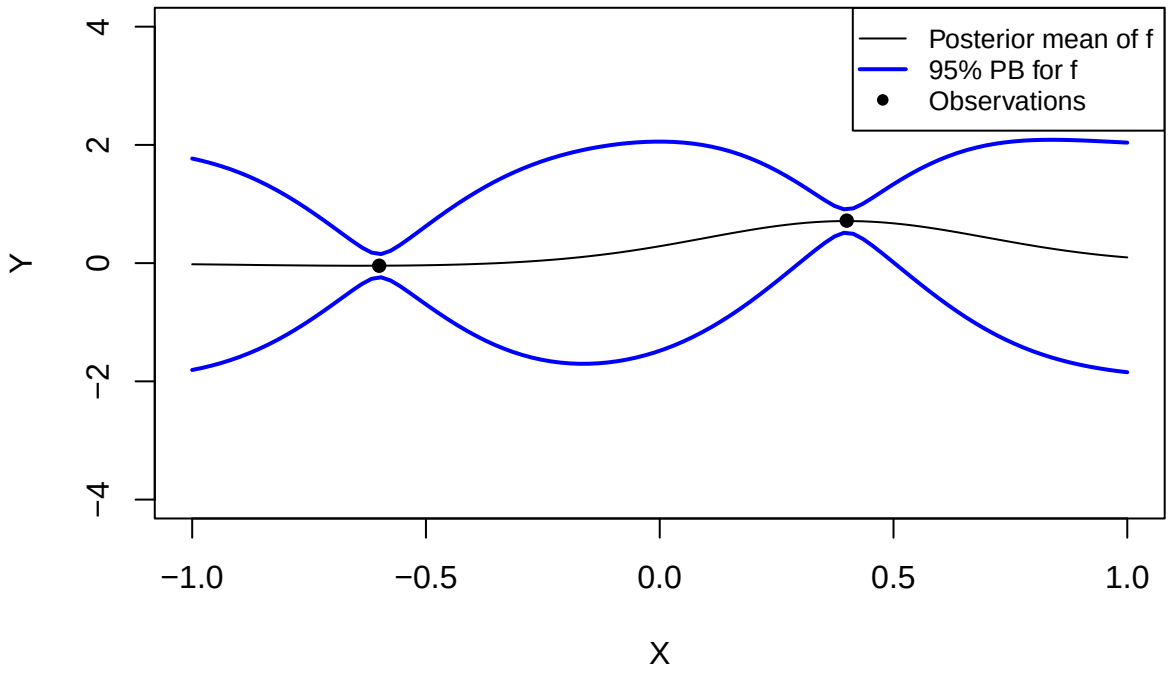


Description 1.3 - Two Observations Mean and Probability Bands for f

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 (point-wise) 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.

Solution 1.3 - Two Observations Mean and Probability Bands for f

Updating our posterior with another observation $(x, y) = (-0.6, -0.044)$ we get the following plot on the same interval. When comparing the two graphs we note that the posterior mean of f has changed.



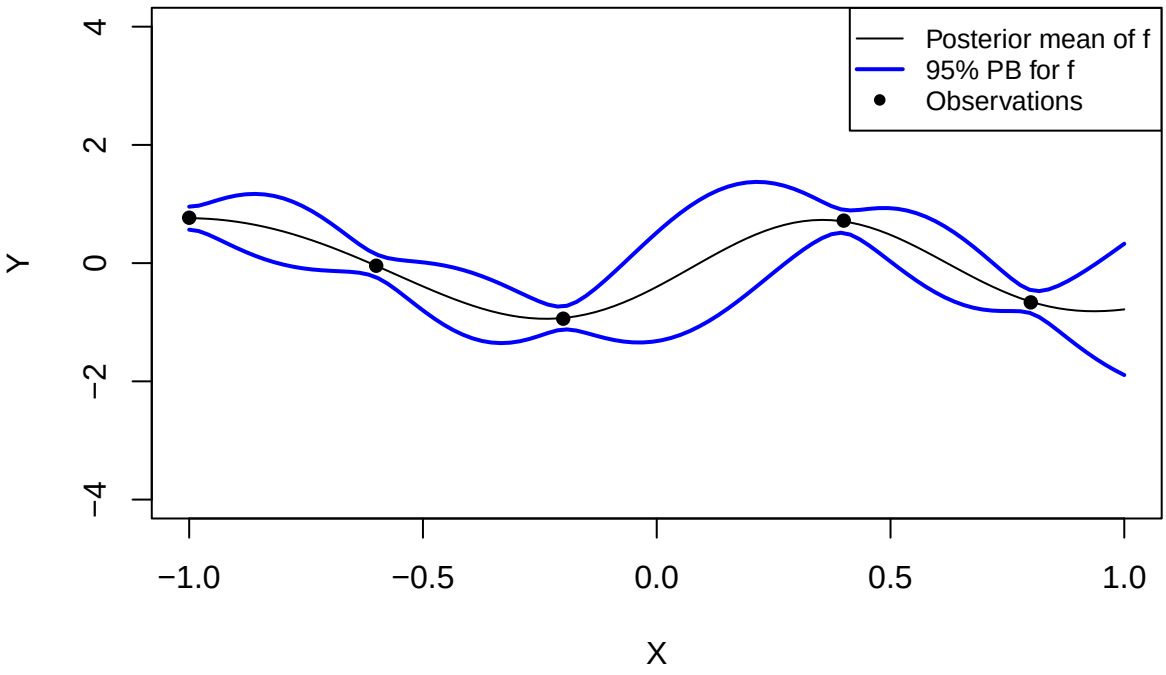
Description 1.4 - Five Observations Mean and Probability Bands for f

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	-1.0	-0.6	-0.2	0.4	0.8
y	0.768	-0.044	-0.940	0.719	-0.664

Solution 1.4 - Five Observations Mean and Probability Bands for f

Updating our posterior with five observations (three new) we get the following plot on the same interval. When comparing the this with our previous graphs we once again note that the posterior mean of f has changed.

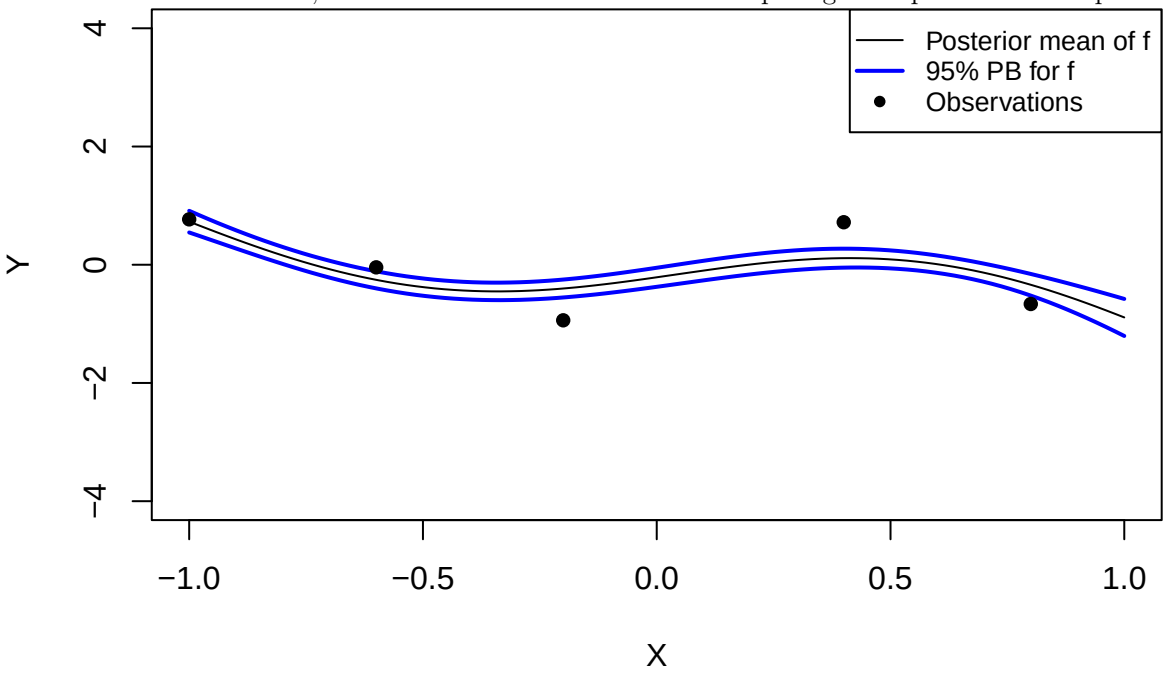


Description 1.5 - Five Observations New Hyperparameters

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

Solution 1.5 - Five Observations New Hyperparameters

Using the same observations as above but with hyperparamters $\sigma_f = 1$ and $l = 1$ we get the plot below. First and foremost we note that we only update the l hyperparameter compared with our previous plot. l controls the smoothness of the function and thus as l gets larger the smoothness of the function increases, which can be observed when comparing this plot with our previous one.



Task 2 GP Regression with kernlab

Description 2.1 - Define Squared Exponential Kernel

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. You can read the dataset with the command: read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/ Code/TempTullinge.csv", header=TRUE, sep=";")

Create the variable *time* which records the day number since the start of the dataset (i.e., *time*= 1, 2, . . ., $365 \times 6 = 2190$). 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.

Familiarize yourself with the functions *gausspr* and 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 $X_* = (2, 3, 4)^T$.

Solution 2.1 - Define Squared Exponential Kernel

We define our squared exponential kernel according to the code-block below. When evaluating our kernel function in the point $x = 1$, $x' = 2$ we get the output of $(0.6065307, 0.6065307)$. Further we get the matrix below using the *kernelMatrix* function to compute the covariance matrix $K(X, X_*)$ for the input vectors $X = (1, 3, 4)^T$ and $X_* = (2, 3, 4)^T$. We obtain 1 in [2,2] and [3,3] because we have the same values in our input vector for the second and third element.

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

```
## 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
```

Description 2.2 - GP Regression, Temp as a Function of Time

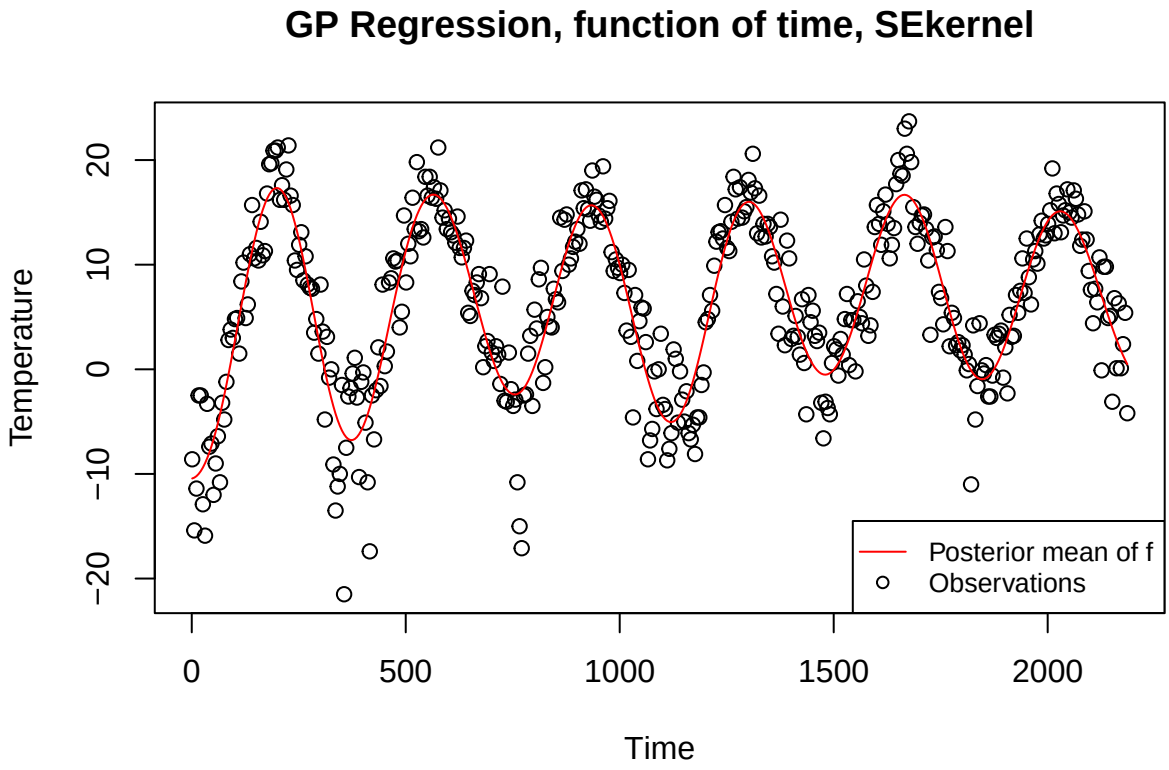
Consider first the following model:

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

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 squared exponential function from (1) with $\sigma_f = 20$ and $l = 0.2$. 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). Play around with different values on σ_f and l (no need to write this in the report though).

Solution 2.2 - GP Regression, Temp as a Function of Time

Considering the following model above: We get the plot below when scatterplotting the data (black dots) and the posterior mean of f (red line) with $\sigma_f = 20$ and $l = 0.2$. Notable is that the posterior mean of f seams to fit the data pretty well.

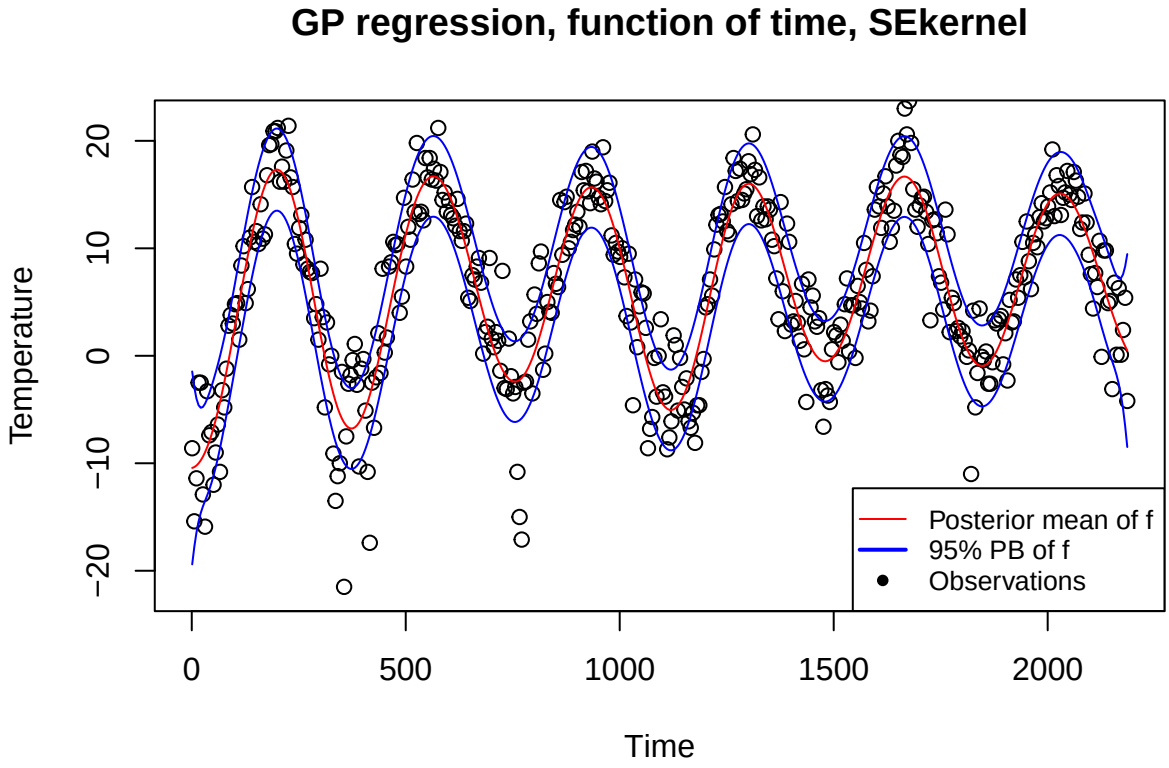


Description 2.3 - GP Regression add Probability Bands

kernlab can compute the posterior variance of f, but it seems to be a bug in the code. So, do your own computations for the posterior variance of f and plot the 95 % probability (pointwise) bands for f. Superimpose these bands on the figure with the posterior mean that you obtained in (2). **Hint**: Note that Algorithm 2.1 on page 19 of Rasmussen and Willams' book already does the calculations required. Note also that *kernlab* scales the data by default to have zero mean and standard deviation one. So, the output of your implementation of Algorithm 2.1 will not coincide with the output of *kernlab* unless you scale the data first. For this, you may want to use the R function *scale*. When plotting the results, do not forget to unscale by multiplying with the standard deviation and adding the mean.

Solution 2.3 - GP Regression add Probability Bands

Plotting the 95% probability bands (blue lines) for f we get the plot below.



Description 2.4 - GP Regression, Temp as a Function of Day

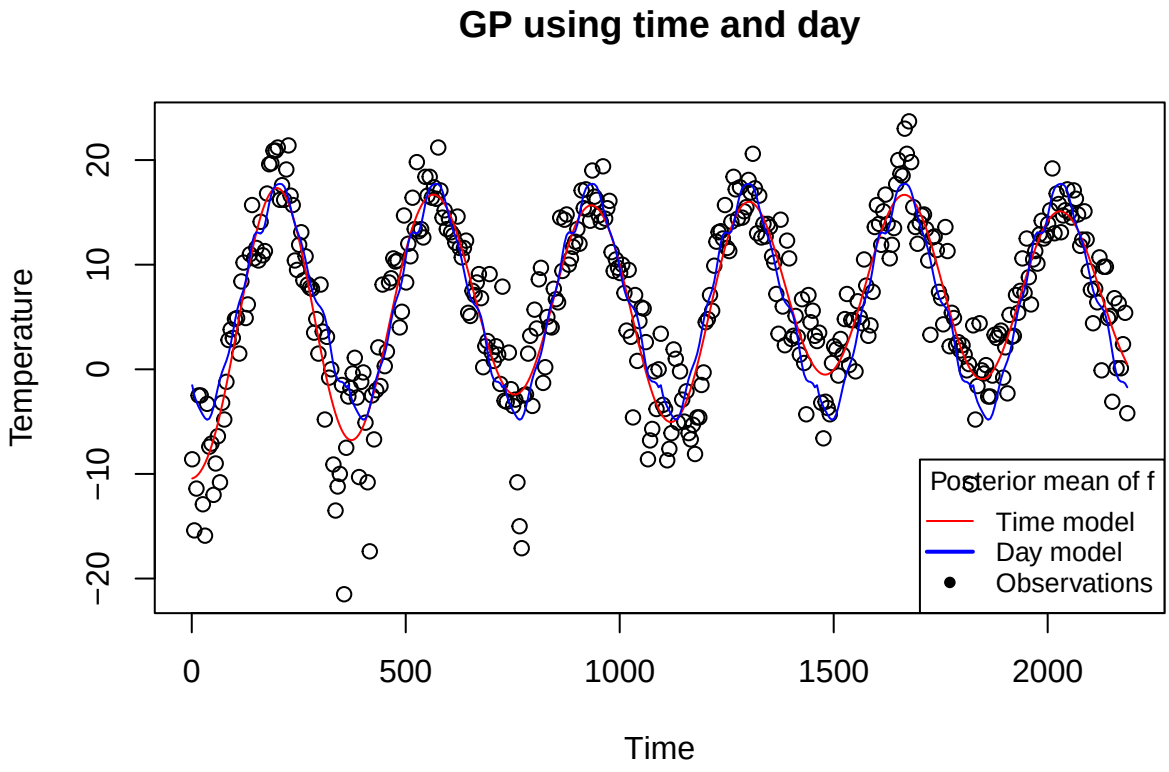
Consider now the following model:

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

Estimate the model using the squared exponential function with $\sigma_f = 20$ and $l = 0.2$. 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.

Solution 2.4 - GP Regression, Temp as a Function of Day

We get the plot below when scatterplotting the data (black dots) and the posterior mean of f (blue line) with $\sigma_f = 20$ and $l = 0.2$. The red line is the posterior mean of f from the previous model. Notable is that the posterior mean of f with temp as a model of day seams to fit the data pretty well but that we obtain the same posterior mean for every year. I.e., this model only takes what day it is into consideration and not data from previous years and trends over the years.



Description 2.5 - Periodic Kernel

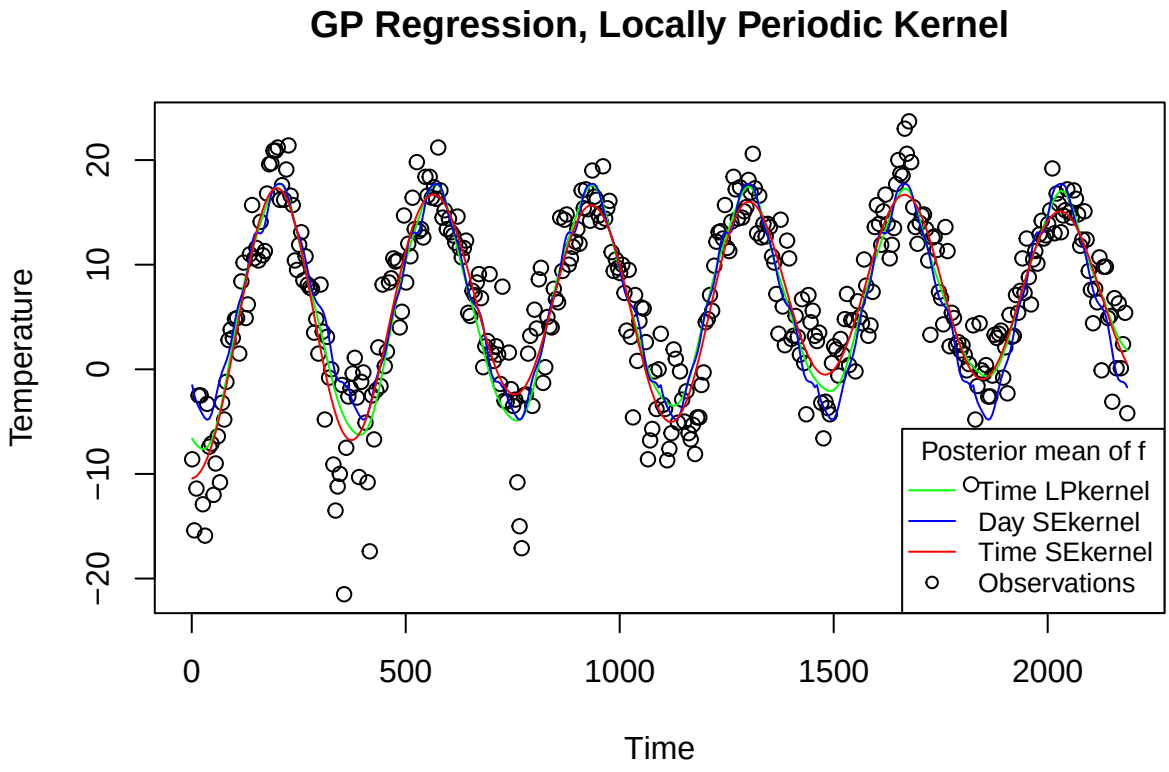
Finally, implement the following extension of the squared exponential kernel with a periodic kernel (a.k.a. locally periodic kernel):

$$k(x, x') = \sigma_f^2 \exp\left\{-\frac{2\sin^2(\pi|x - x'|/d)}{l_1^2}\right\} \exp\left\{-\frac{1}{2}\frac{|x - x'|^2}{l_2^2}\right\}$$

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 $\sigma_f = 20$, $l_1 = 1$, $l_2 = 10$ and $d = 365/sd(time)$. The reason for the rather strange period here is that *kernlab* standardizes the inputs to have standard deviation of 1. Compare the fit to the previous two models (with $\sigma_f = 20$ and $l = 0.2$). Discuss the results.

Solution 2.5 - Logically Periodic Kernel

Finally we implement an extension of the squared exponential kernel with a periodic kernel. Estimating the GP model using the time variable, this new kernel and the parameters $\sigma_f = 20$, $l_1 = 1$, $l_2 = 10$, and $d = 365/sd(time)$ we can plot the posterior mean of f (green line) in the plot above. When doing so we get the plot below. Comparing the three lines we don't note any significant differences appart from the blue line staying the same over the years (as said before). Between the green line and the red we don't note any mayor differences, if anything the green line seem to follow the appearance of the blue with the given parameters.



Task 3 GP Classification with kernlab

Description 3.1 - Gaussian Process Classification

Download the banknote fraud data:

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

You can read about this dataset here. Choose 1000 observations as training data using the following command (i.e., use the vector SelectTraining to subset the training observa- tions):

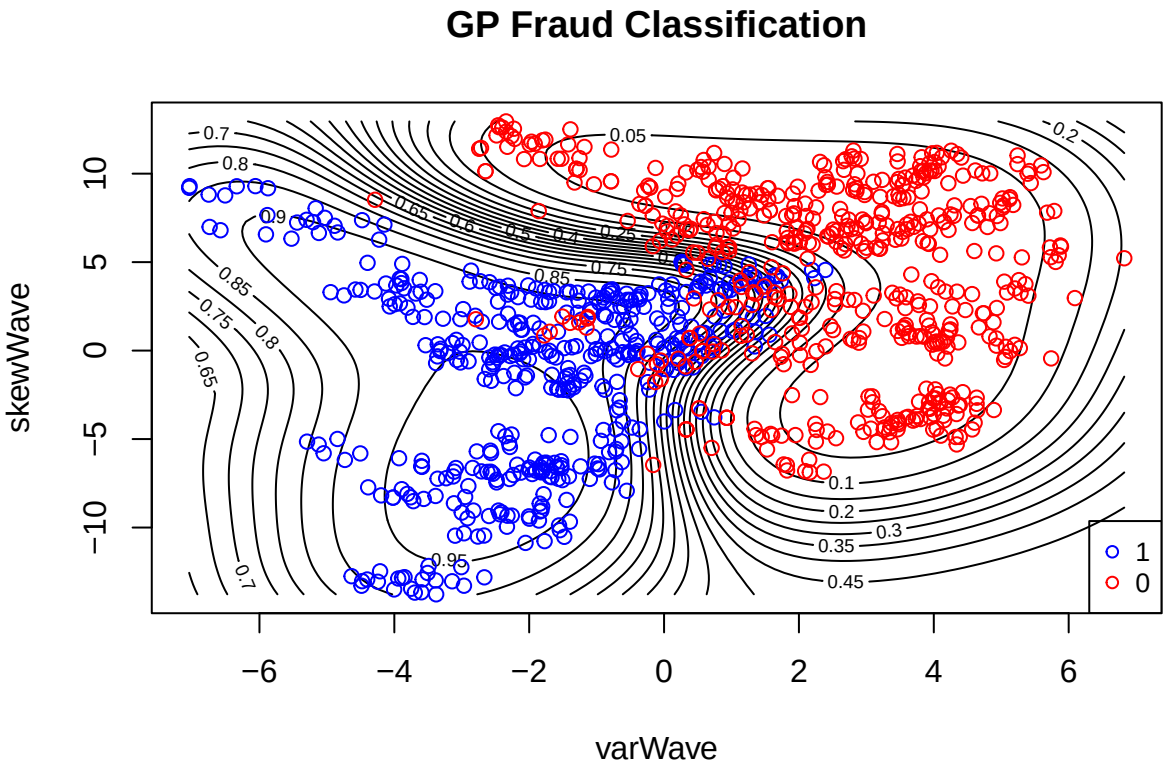
```
set.seed(111);
SelectTraining <- sample(1:dim(data)[1], size = 1000, replace = FALSE)
```

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.

Solution 3.1 - Gaussian Process Fraud Classification

The circles with respective probability represent the probability of an observation being fraud and the points are the actual fraud values from the training data. Looking at the plot we note that fraud from the training data (in blue) are typically associated with a lower varWave but that there are some outliers. For some points we are as confident as 95% that it is fraud and for others 5% confident that there isn't (majority of the red points).

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



Using the varWare and skewWave to predict fraud we get an accuracy of 0.941 for the training data and a confusion matrix as follows:

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

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

Description 3.2 - Predictions with the GP Classification Model

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

Solution 3.2 - Predictions with the GP Classification Model

Using our model to make prediction for the test dataset we get an accuracy of 0.9247312 and the confusion matrix below:

```
##
## pred   0   1
##    0 199   9
##    1  19 145
```

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

Comparing the two we note that the accuracy for the training data is higher than for the test data, which makes sense since the data is trained on the training data. The difference isn't that large though, suggesting the model generalize well.

Description 3.3 - Add Features to the GP Classification Model

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

Solution 3.3 - Add More Feautres to the GP Classification Model

Training a model using all four covariates we get an accuracy of 0.997 for the training data and 0.9946237 for the test data which is higher than what we got using only two covariates. This suggest that this second model that uses all four covariates are better since we obtain a higher accuracy for the test data.

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

```
##
## pred   0   1
##    0 541   0
##    1   3 456
```

```
## [1] 0.997
```

```
##
## pred   0   1
##    0 216   0
##    1   2 154
```

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