732A96: Advance Machine Learning

LAB 3: Gaussian Processes

Arian Barakat, ariba405
Rebin Hosini, rebho150
Hao chi Kiang, haoki222
Joshua Hudson, joshu107
Carles Sans Fuentes, carsa564

Question 1: Implementing Gaussian process regression from scratch

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

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

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

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

(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 ($x^\star$). 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)
- x_* (vector of inputs where the posterior distribution is evaluated)
- hyperParam (vector with two elements σ_f and ℓ)
- sigmaNoise (σ_n)

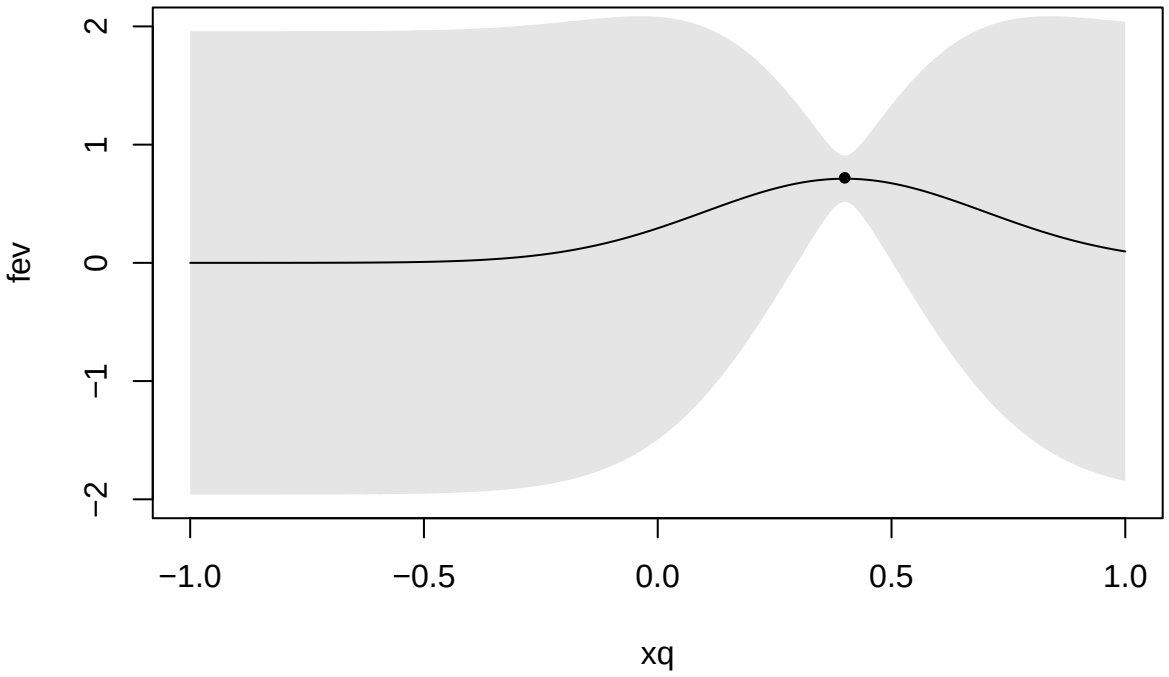
Answer:

See code in appendix

(b)

Now let the prior hyperparameters be $\sigma_f = 1, \ell = 0.3$. 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 $x \in [-1, 1]$. Plot also 95% probability (pointwise) bands for f.

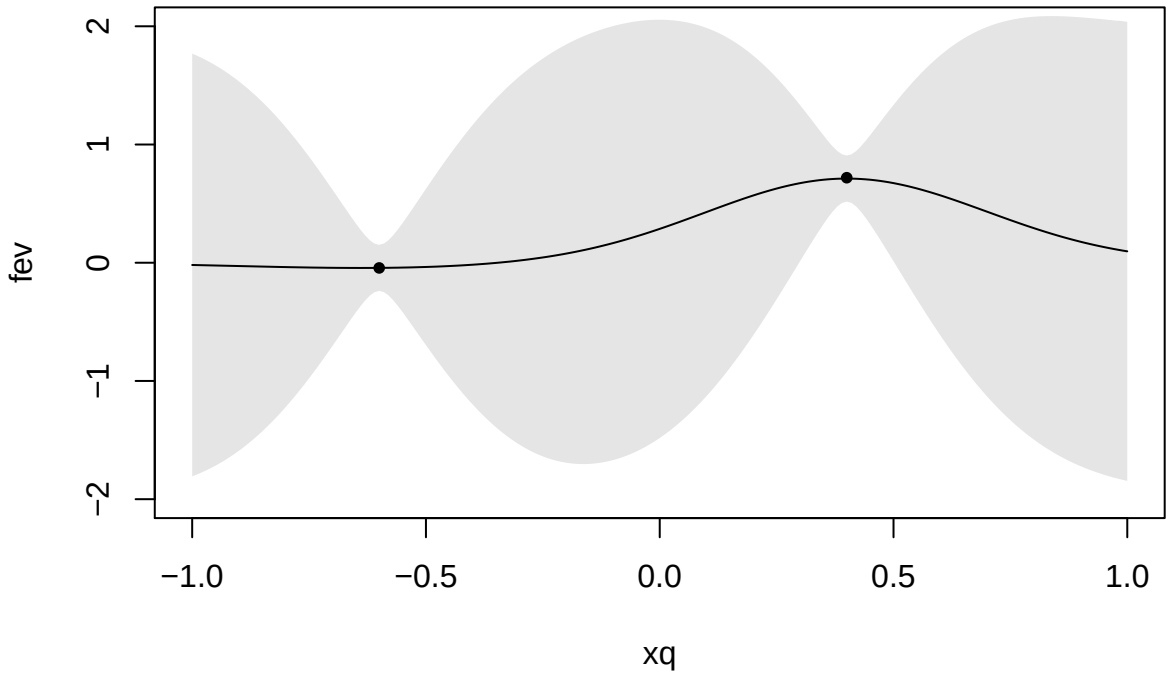
Answer:



(c)

Update your posterior from 1b) 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 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!]

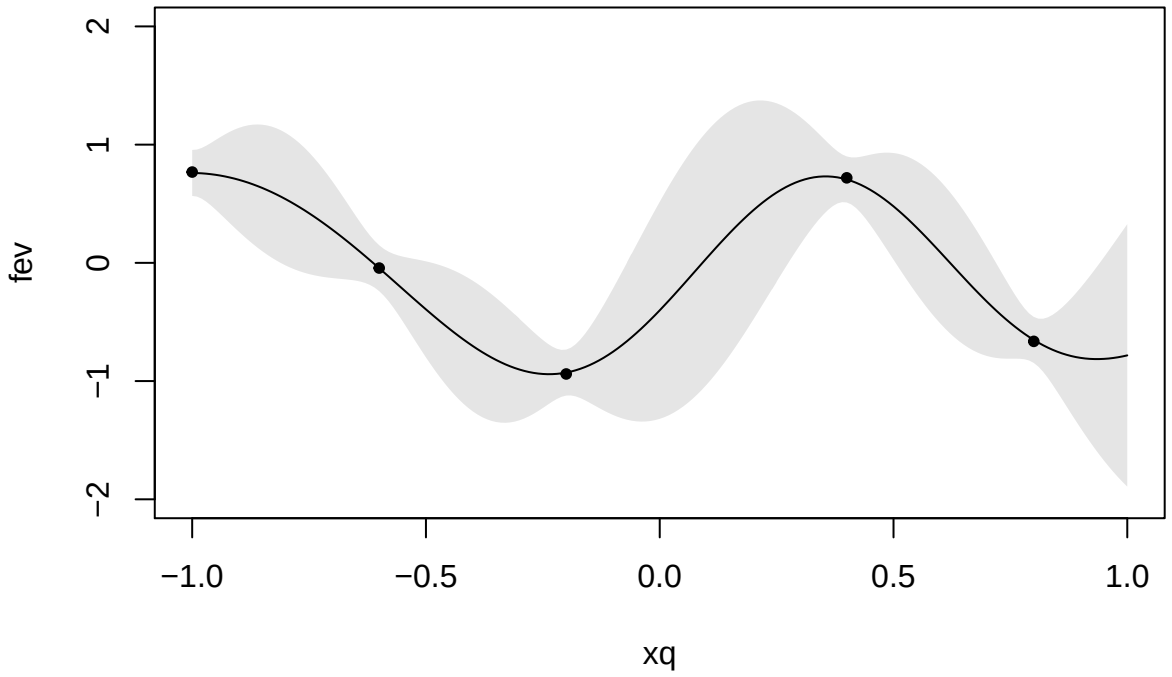
Answer:



(d)

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 $x \in [-1, 1]$. Plot also 95% probability intervals for f.

Answer:

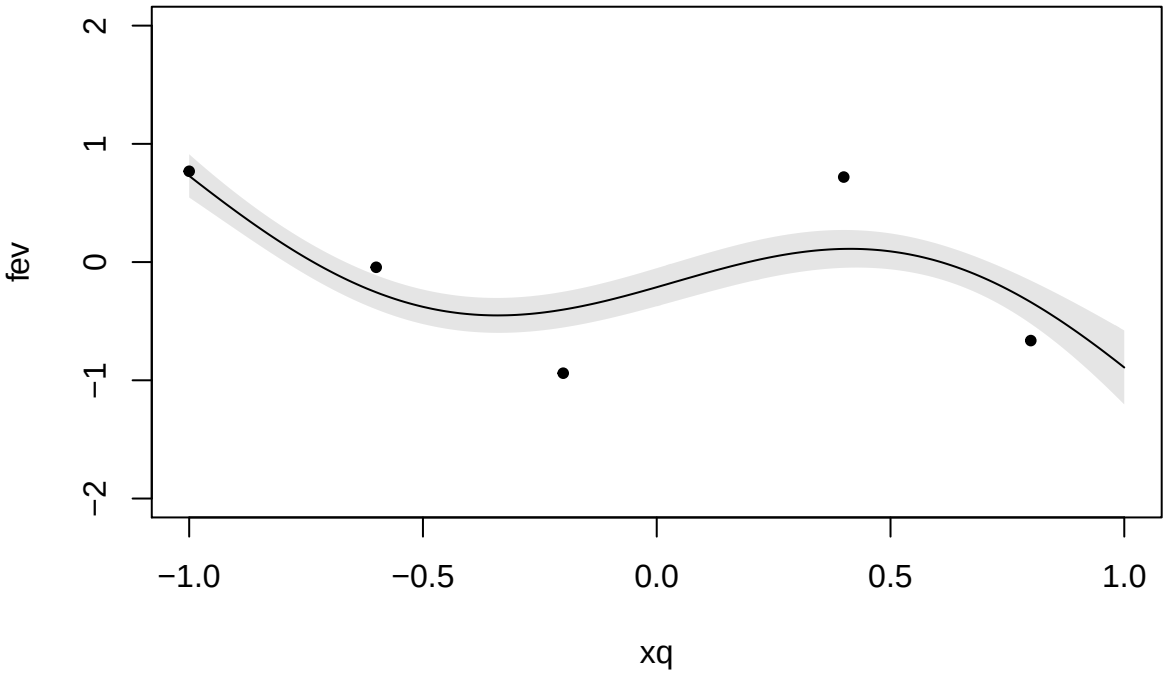


(e)

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

Answer:

If we compare the figure for question (e) with the one provided in question (d) we can observe that the former one has a more smoothed or "generalized" posterior GP. A greater value of the ℓ hyperparamater implies that we place relativly more weight on distant training points compared to when we use a kernel with lower length scale. We can also see that the variance of the function decrease when we make ℓ smaller, which implies that we are more certain (falsely) about the location of the posterior GP.



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

Create the variable **time** which records the day number since the start of the dataset (i.e.time $= 1, 2, ..., 365 \cdot 6 = 2190$). 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's thin out the data by only using every fifth observation. This means that your time variable is now time $= 1, 6, 11, ..., 2186$ and day $= 1, 6, 11, ..., 361, 1, 6, ..., 361$.

(a)

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 my KernLabDemo.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 $K(\mathbf{x}, \mathbf{x}_\star)$ for the input vectors $\mathbf{x} = (1, 3, 4)^T$ and $\mathbf{x}_\star = (2, 3, 4)^T$.

Answer:

```
##             [,1]
## [1,] 0.00386592
```

```
## An object of class "kernelMatrix"
##               [,1]         [,2]        [,3]
## [1,] 3.865920e-03 2.233631e-10 1.92875e-22
## [2,] 3.865920e-03 1.000000e+00 3.86592e-03
## [3,] 2.233631e-10 3.865920e-03 1.00000e+00
```

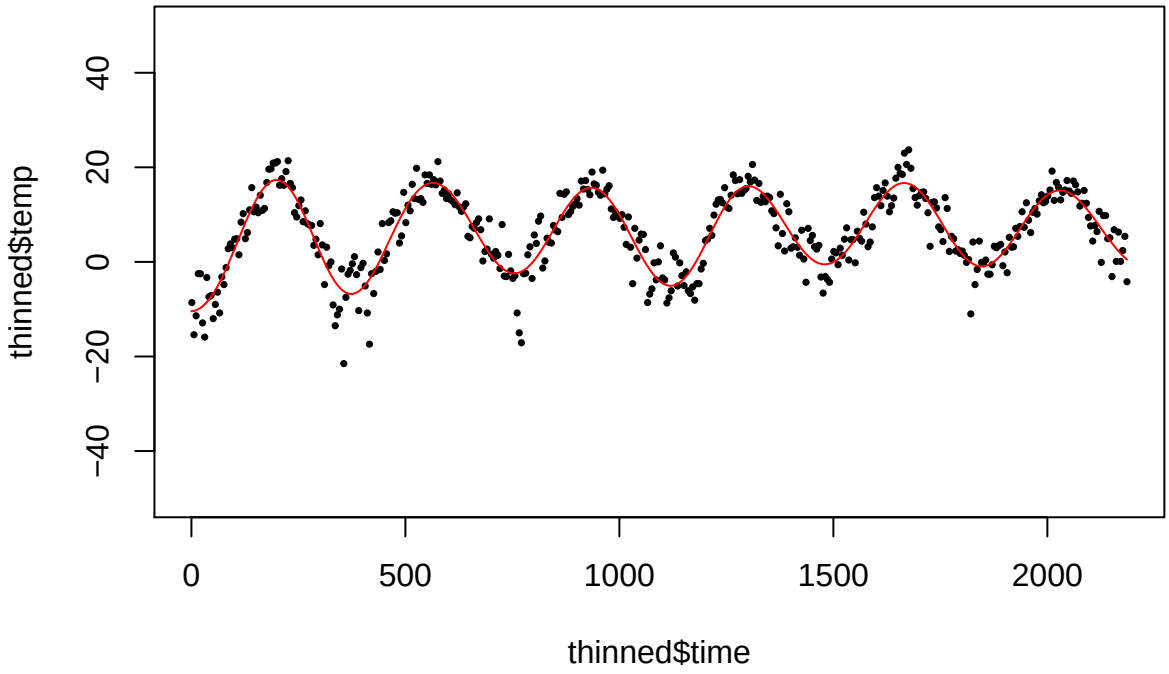
(b)

Consider first the model:

$$temp = f(\text{time}) + \varepsilon \quad \varepsilon \sim N(0, \sigma_n^2)$$
$$f \sim GP\left[0, k(\text{time}, \text{time}')\right]$$

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 $\ell = 0.2$. Use the predict function to compute the posterior mean at every data point in the training datasets. 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).

Answer:



(c)

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 $\ell = 0.2$. Superimpose those bands on the figure with the posterior mean in 2b).

Answer:

The figure is presented in question (d) together with other model.

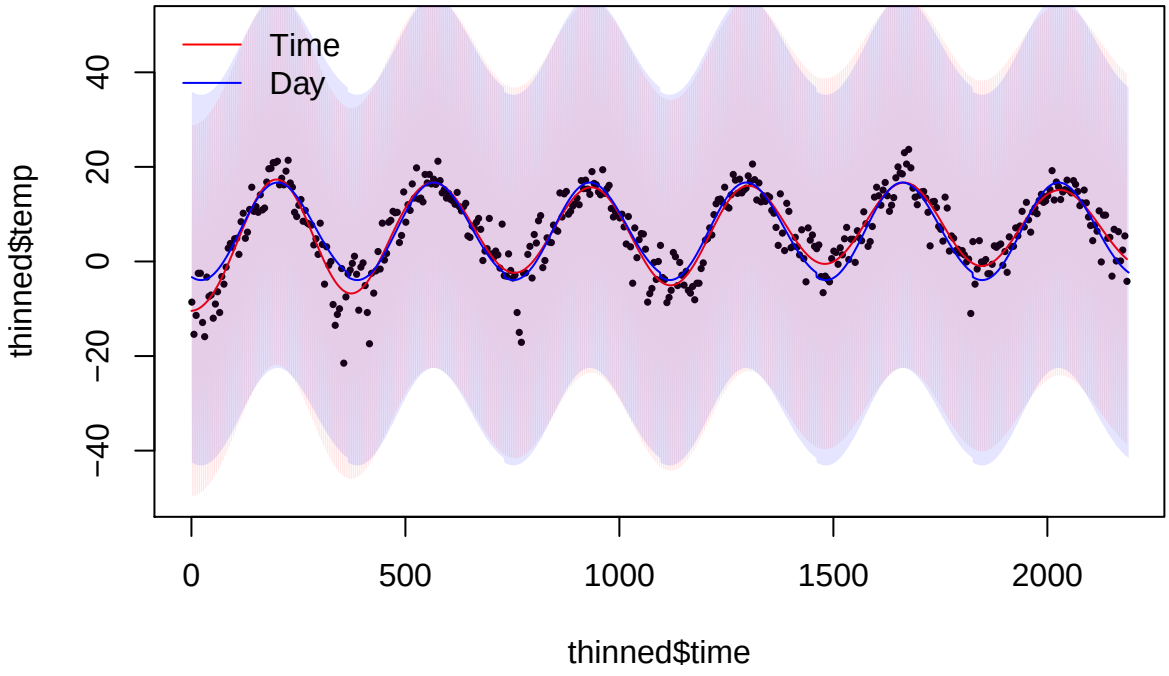
(d)

Estimate the model using the squared exponential function from 2a) with $\sigma_f = 20$ and $\ell = 6 \cdot 0.2 = 1.2$. (I multiplied ℓ 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 $\sigma_f = 20, \ell = 0.2$. Note that this plot should also have the time variable on the horizontal axis. Compare the results from using time to the ones with day. What are the pros and cons of each model?

Answer:

Before we start discussing the result it's important to note that R recycles the prediction from the "Day" model when we impose the predictions over the time-axis. In other words, the predictions will have the same pattern from year to year.

From the figure, we can observe that the models are quite similar to each other with some minor differences at the yearly minimum and maximum temperatures. The "Day" model has the benefit of being able to capture the seasonal variation within the years, while the "Time" model is beneficial for capturing the longterm variation and trend. It can be assumed that none of these models will be able to capture the interacted variation between the variables independently, at least very poorly. One could theoretically be better off by constructing a kernel that is able to combine the effect of the two variables.



(e)

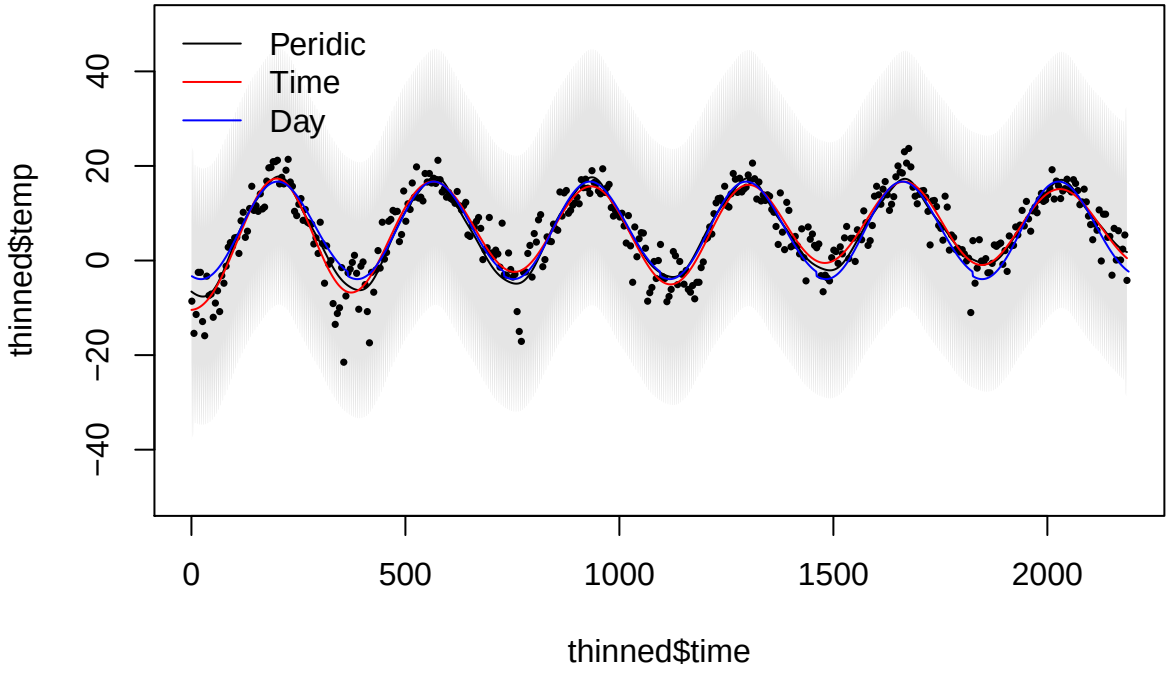
Now implement a generalization of the periodic kernel given in my slides from Lecture 2 of the GP topic (Slide 6)

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

Note that we have two different length scales here, and ℓ_2 controls the correlation between the same day in different years (ℓ_2). So this kernel has four hyperparameters σ_f, ℓ_1, ℓ_2 and the period d. Estimate the GP model using the time variable with this kernel with hyperparameters $\sigma_f = 20$, $\ell_1 = 1$, $\ell_2 = 10$ and $d = 365/$ sd(time). The reason for the rather strange period here is that kernlab standardized inputs to have standard deviation of 1. Compare the fit to the previous two models (with $\sigma_f = 20, \ell = 0.2$). Discuss the results.

Answer:

As mentioned in the previous question (and as we can observe in the figure) the periodic model is able to capture yearly trend and at the same time periodic characteristic of function.



Question 3: Gaussian process classification using the kernlab package

(a)

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. 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. Compute the confusion matrix for the classifier and its accuracy.

Answer:

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

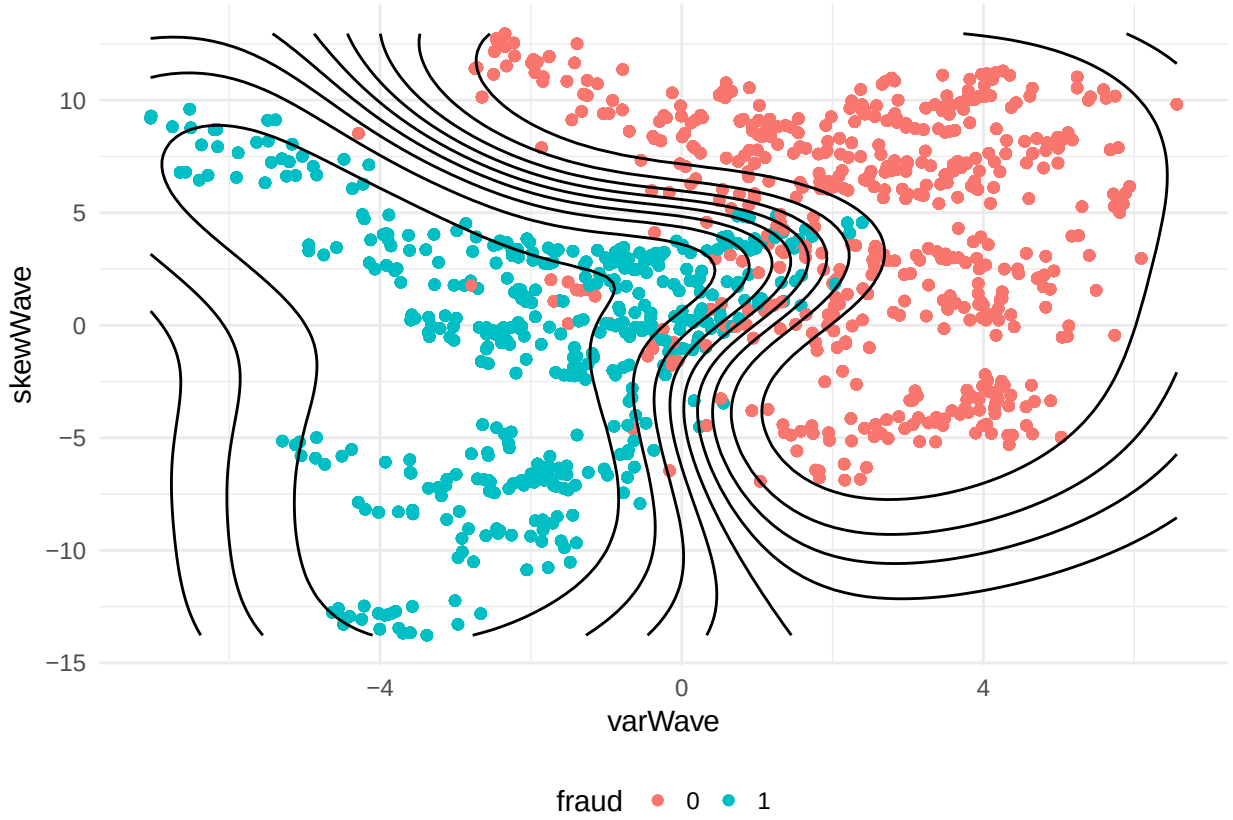


Table 1: Confusion Matrix, Training Data

	0	1
0	512	44
1	24	420

```
## Accuracy, Train data: 0.932000
```

(b)

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

Answer:

Table 2: Confusion Matrix, Test Data

	0	1
0	191	15
1	9	157

```
## Accuracy, Test data: 0.935484
```

(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:

From the result we can observe a better accuracy when including more features.

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

Table 3: Confusion Matrix, Test Data (All four Covariates)

	0	1
0	205	1
1	0	166

```
## Accuracy, Test (All four Covariates) data: 0.997312
```

Appendix

```
knitr::opts_chunk$set(echo = FALSE,
                      warning = FALSE,
                      message = FALSE,
                      error = FALSE)

library(ggplot2)
library(kernlab)
library(timeDate)
library(dplyr)
library(grid)
library(gridExtra)
library(knitr)
pathData <- "../Data"

kernfn = function (x1, x2, hyperParam) {
    if (is.null(dim(x1))) x1 = as.matrix(x1)
    if (is.null(dim(x2))) x2 = as.matrix(x2)
    with(hyperParam, {
        res = matrix(-1e-6, ncol(x1), ncol(x2))
        for (p in 1:ncol(x1)) {
            for (q in 1:ncol(x2)) {
                res[p,q] = tausq * exp( - sum(((x1[,p] - x2[,q])/l)^2)/2 )
            }
        }
        res})
}

posteriorGP = function (x, y, xStar, hyperParam, sigmaNoise, kernfn) {
    kxx = kernfn(x, x, hyperParam)
    kxq = kernfn(x, xStar, hyperParam)
    kqq = kernfn(xStar, xStar, hyperParam)
    L = chol(kxx + diag(sigmaNoise^2, ncol(x)))
    alp = solve(L, solve(t(L), y))
    fev = crossprod(kxq, alp)
    v = solve(t(L), kxq)
    vf = kqq - crossprod(v)
    list(fev = fev, vf = vf)
}

plot_gppostr = function (x, y, xq, fev, vf) {
    plot(xq, fev, ylim = c(-2,2), type = 'l')
    points(x, y, pch = 20)
    polygon(x = c(xq, rev(xq)),
            y = c(fev - 1.96*sqrt(diag(vf)), rev(fev + 1.96*sqrt(diag(vf)))),
            col = rgb(0,0,0,0.1),
            border = 0)
}

xq = t(seq(-1, 1, by = 0.01))

x1 = t(matrix(0.4))
```

```r
y1 = matrix(c(0.719))
gppostr1 = posteriorGP(x1, y1, xq, list(l = 0.3, tausq = 1), 0.1, kernfn)
plot_gppostr(x1, y1, xq, gppostr1$fev, gppostr1$vf)

x2 = t(matrix(c(-0.6,0.4)))
y2 = matrix(c(-0.044,0.719))
gppostr2 = posteriorGP(x2, y2, xq, list(l = 0.3, tausq = 1), 0.1, kernfn)
plot_gppostr(x2, y2, xq, gppostr2$fev, gppostr2$vf)

x = t(matrix(c(-1, -0.6, -0.2, 0.4, 0.8)))
y = matrix(c(0.768, -0.044, -0.940, 0.719, -0.664))
gppostr3 = posteriorGP(x, y, xq, list(l = 0.3, tausq = 1), 0.1, kernfn)
plot_gppostr(x, y, xq, gppostr3$fev, gppostr3$vf)

gppostr4 = posteriorGP(x, y, xq, list(l = 1, tausq = 1), 0.1, kernfn)
plot_gppostr(x, y, xq, gppostr4$fev, gppostr4$vf)

library('functional')
library('kernlab')

tullinge = read.csv('https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/TempTullin
                    header=TRUE, sep=';')
tullinge$time = seq_len(nrow(tullinge))
tullinge$day = seq_len(365)
thinned = tullinge[ which((0:(nrow(tullinge)-1)) %% 5 == 0), ]

new_rbfkern = function (ell, sigmaf) {
    f = Curry(kernfn, hyperParam = list(tausq = sigmaf^2, l = ell))
    class(f) = 'kernel'
    f
}

rbf = new_rbfkern(ell = 0.3, sigmaf = 1)
print(rbf(1,2))
print(kernelMatrix( rbf, c(1,3,4), c(2,3,4)))

signoise = summary(lm(temp ~ time + I(time^2), tullinge))$sigma
rbf2 = new_rbfkern(ell = 0.2, sigmaf = 20)

tulfit1 = gausspr(temp ~ time, data = thinned, kernel = new_rbfkern,
                  kpar = list(ell = 0.2, sigmaf = 20), var = signoise^2 )

tulpred1 = predict(tulfit1, data.frame(time = seq(0, max(thinned$time))))


plot(thinned$temp ~ thinned$time, pch = 16, cex = 0.5, ylim = c(-50,50))
lines(tulpred1 ~ seq(0, max(thinned$time)), type = 'l', col = 2)


tulpred1_homebrew = posteriorGP(t(thinned$time), thinned$temp,
                                t(seq(0, max(thinned$time))),
                                list(l = 0.2, tausq = 20^2),
                                signoise, kernfn)
```

```r
tulfit2 = gausspr(temp ~ day, data = thinned, kernel = new_rbfkern,
                  kpar = list(ell = 1.2, sigmaf = 20), var = signoise^2)

tulpred2 = predict(tulfit2, tullinge)

tulpred2_homebrew = posteriorGP(t(thinned$day), thinned$temp,
                                tullinge$day,
                                list(l = 0.2, tausq = 20^2),
                                signoise, kernfn)

plot(thinned$temp ~ thinned$time, pch = 16, cex = 0.5, ylim = c(-50,50))
lines(tulpred1 ~ seq(0, max(thinned$time)), type = 'l', col = 2)
polygon(x = c(seq(0, max(thinned$time)), rev(seq(0, max(thinned$time)))),
        y = c(tulpred1 - 1.96*sqrt(diag(tulpred1_homebrew$vf)),
              rev(tulpred1 + 1.96*sqrt(diag(tulpred1_homebrew$vf)))),
        col = rgb(1,0,0,0.1), border = 0)
lines(tulpred2 ~ tullinge$time, type = 'l', col = 4)
polygon(x = c(tullinge$time, rev(tullinge$time)),
        y = c(tulpred2 - 1.96*sqrt(diag(tulpred2_homebrew$vf)),
              rev(tulpred2 + 1.96*sqrt(diag(tulpred2_homebrew$vf)))),
        col = rgb(0,0,1,0.1), border = 0)
legend("topleft",
       legend = c("Time", "Day"),
       lty = c(1,1),
       col = c("red", "blue"),
       bty = "n")

new_periodickern = function (d, sigmaf, ell1, ell2) {
    f = function (x1, x2) {
        ## I am assuming x1 and x2 are both single row vectors
        res = matrix(-1e-6, length(x1), length(x2))
        for (p in 1:length(x1)) {
            for (q in 1:length(x2)) {
                res[p,q] = sigmaf^2 * exp(- 2 * sin( pi * abs(x1[p] - x2[q]) / d )^2 / ell1^2 ) *
                    exp( - sum(((x1[p] - x2[q])/ell2)^2)/2 )
            }
        }
        res
    }

    class(f) = 'kernel'
    f
}

tulfit3 = gausspr(
    temp ~ time,
    data = thinned,
    kernel = new_periodickern,
    kpar = list(d = 365/sd(thinned$time), ell2 = 10, ell1 = 1, sigmaf = 20),
    var = signoise^2)
```

```
tulpred3 = predict(tulfit3, data.frame(time = seq(0, max(thinned$time))))

tulpred3_homebrew = posteriorGP(
    t(thinned$time), thinned$temp,
    seq(0, max(thinned$time)),
    list(d = 365/sd(thinned$time), ell2 = 10, ell1 = 1, sigmaf = 20),
    signoise, function(x1, x2, hyperparams) {
        do.call(new_periodickern, hyperparams)(x1, x2)
    })

plot(thinned$temp ~ thinned$time, pch = 16, cex = 0.5, ylim = c(-50, 50))
lines(tulpred3 ~ seq(0, max(thinned$time)), type = 'l', col = 1)
polygon(x = c(seq(0, max(thinned$time)), rev(seq(0, max(thinned$time)))),
        y = c(tulpred3 - 1.96*sqrt(diag(tulpred3_homebrew$vf)),
              rev(tulpred3 + 1.96*sqrt(diag(tulpred3_homebrew$vf)))),
        col = rgb(0,0,0,0.1), border = 0)
lines(tulpred1 ~ seq(0, max(thinned$time)), type = 'l', col = 2)
lines(tulpred2 ~ tullinge$time, type = 'l', col = 4)
legend("topleft",
       legend = c("Peridic","Time", "Day"),
       lty = c(1,1,1),
       col = c("black","red", "blue"),
       bty = "n")



# Question 3

dataBank <- read.csv(paste(pathData,"banknoteFraud.csv", sep = "/"),
                     header=FALSE, sep=",")
names(dataBank) <- c("varWave","skewWave","kurtWave","entropyWave","fraud")
dataBank[,5] <- as.factor(dataBank[,5])
set.seed(111)
trainOBS <- sample(1:dim(dataBank)[1], size = 1000, replace = FALSE)



# Question 3 (a)

fraudGPclass <- gausspr(fraud ~ varWave + skewWave,
                        data = dataBank[trainOBS,])
trainPredProbabilities <- predict(fraudGPclass,
                              newdata = dataBank[trainOBS,],
                              type = "probabilities")
trainPredLabel <- predict(fraudGPclass,
                      newdata = dataBank[trainOBS,],
                      type = "response")

dataGrid <- expand.grid(seq(min(dataBank[trainOBS,]$varWave),
                            max(dataBank[trainOBS,]$varWave),
                            length.out = 100),
                        seq(min(dataBank[trainOBS,]$skewWave),
                            max(dataBank[trainOBS,]$skewWave),
```

```
                            length.out = 100))
colnames(dataGrid) <- c("varWave", "skewWave")
predPrGrid <- predict(fraudGPclass,
                      newdata = dataGrid,
                      type = "probabilities")

dataGrid <- cbind(dataGrid, predPrGrid)
colnames(dataGrid) <- c(colnames(dataGrid)[1:2], "Pr0", "Pr1")

#  Plot
cbind(dataBank[trainOBS,],
      data.frame(Pr = ifelse(dataBank[trainOBS,]$fraud == 1,
                             trainPredProbabilities[,2],
                             trainPredProbabilities[,1]),
                 Pr2 = predPrGrid[,2])) %>%
  ggplot() +
  geom_point(aes(x = varWave,
                 y = skewWave,
                 col = fraud)) +
  geom_contour(data = dataGrid,
               aes(x = varWave,
                   y = skewWave,
                   z =  Pr1),
               col = "black") +
  theme_minimal() +
  theme(legend.position = "bottom")



confTableTrainPred <- table("True" = dataBank[trainOBS,]$fraud,
                            "Predicted" = trainPredLabel)


kable(confTableTrainPred, caption = "Confusion Matrix, Training Data")

accurTrain <- sum(diag(confTableTrainPred))/sum(confTableTrainPred)

printString <- sprintf("Accuracy, %s data: %f", "Train", accurTrain)
cat(printString)


# Question 3 (b)

testPredLabel <- predict(fraudGPclass,
                         newdata = dataBank[-trainOBS,],
                         type = "response")

confTableTestPred <- table("True" = dataBank[-trainOBS,]$fraud,
                           "Predicted" = testPredLabel)


kable(confTableTestPred, caption = "Confusion Matrix, Test Data")
```

```r
accurTest <- sum(diag(confTableTestPred))/sum(confTableTestPred)


printString <- sprintf("Accuracy, %s data: %f", "Test", accurTest)
cat(printString)


# Question 3 (c)

fraudGPclassAllcov <- gausspr(fraud ~ varWave + skewWave + kurtWave + entropyWave,
                              data = dataBank[trainOBS,])


testPredLabelAllcov <- predict(fraudGPclassAllcov,
                               newdata = dataBank[-trainOBS,],
                               type = "response")


confTableTestPredAllcov <- table("True" = dataBank[-trainOBS,]$fraud,
                            "Predicted" = testPredLabelAllcov)


kable(confTableTestPredAllcov, caption = "Confusion Matrix, Test Data (All four Covariates)")

accurTestAllcov <- sum(diag(confTableTestPredAllcov))/sum(confTableTestPredAllcov)


printString <- sprintf("Accuracy, %s data: %f", "Test (All four Covariates)", accurTestAllcov)
cat(printString)
```