Advanced Machine Learning - Gaussian Processes

Ali Etminan

10/16/2020

Implementing GP Regression

Function to calculate the posterior distribution using the Gaussian Process regression model:

```
# Covariance function
SquaredExpKernel <- function(x1,x2,sigmaF,l){
  n1 <- length(x1)
  n2 <- length(x2)
  K <- matrix(NA,n1,n2)
  for (i in 1:n2){
    K[,i] <- sigmaF^2*exp(-0.5*( (x1-x2[i])/l)^2 )
  }
  return(K)
}


posteriorGP <- function(X, y, XStar, sigmaNoise, sigmaF, ell) {
  n <- length(XStar)
  k <- SquaredExpKernel(X,X,sigmaF,ell)
  L <- chol(k + (sigmaNoise * diag(ncol(k))))
  solved <- solve(t(L),y)
  alpha <- solve(L,solved)
  kstar <- SquaredExpKernel(X,XStar,sigmaF,ell)
  f_bar <- t(kstar) %*% alpha   #predictive mean
  v <- solve(t(L),kstar)
  kstarstar <- SquaredExpKernel(XStar, XStar,sigmaF,ell)
  V <- kstarstar - t(v) %*% v   #predictive variance
  #log marginal likelihood
  marg_llik <- .5*(t(y)%*%alpha) - sum(log(diag(L))) - (n/2)*log(2*pi)
  return(list(predictive_mean = f_bar, predictive_variance = V, log_marginal=marg_llik))
}
```

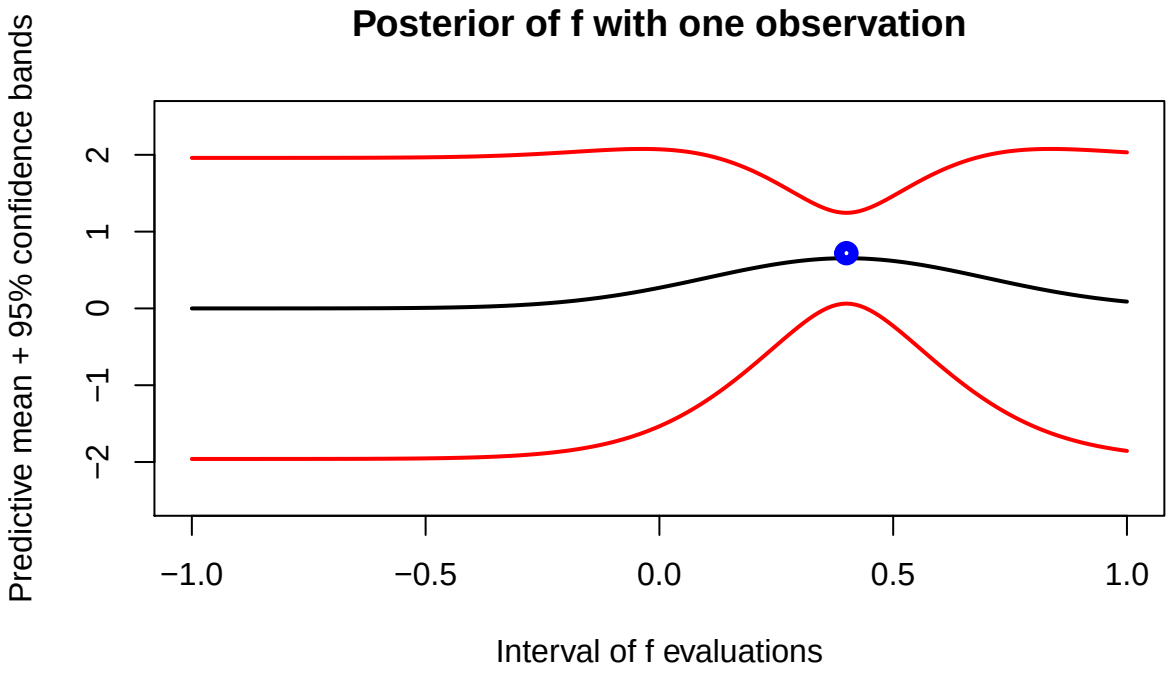
Note that in order to calculate the covariance matrix we have used the available *SquaredExpKernel*. We use $\sigma_f = 1$ and $\ell = 0.3$ for the next three parts.

Updating the prior with one observation

Now we update the prior with the point $(0.4, 0.719)$ and evaluate the kernel function for 201 values of the $[-1, 1]$ interval.

```
xgrid <- seq(-1,1,.01)
sigmaN <- .1
ell <- .3
```

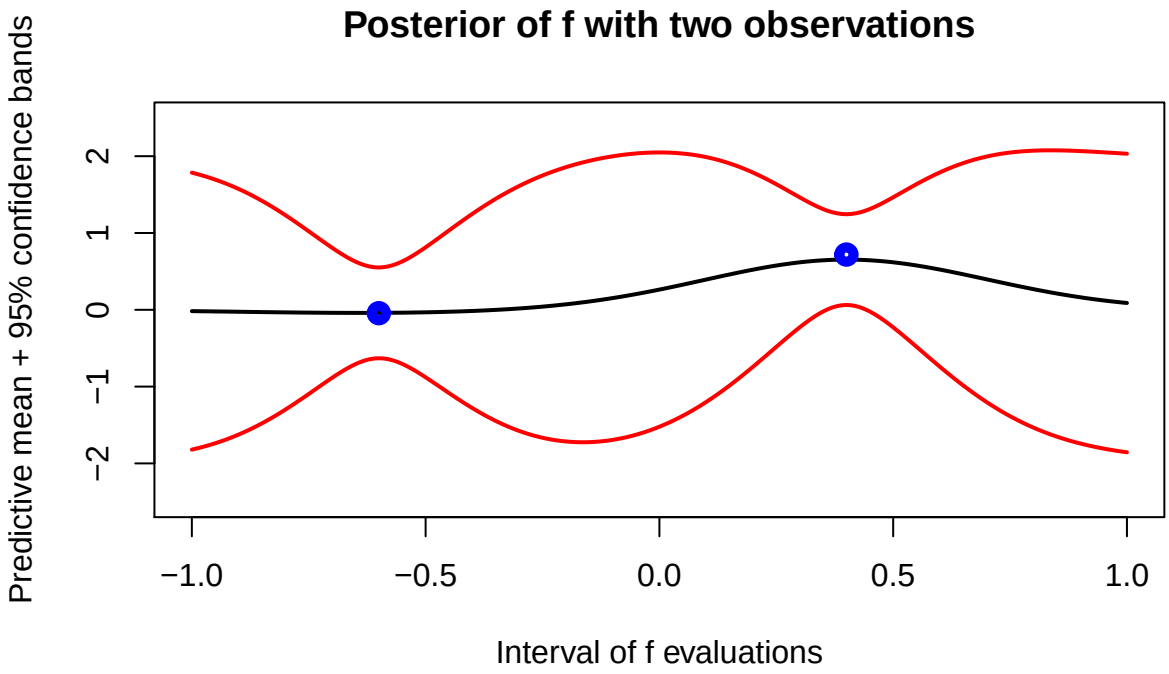
```
sigmaF <- 1
X1 <- .4
y1 <- .719
pst <- posteriorGP(X1,y1,xgrid,sigmaN, sigmaF, ell)
```



Updating the prior with two observations

Now we add another observation to update our prior. For this purpose, the two observations $(0.4, 0.719)$ and $(-0.6, -0.044)$ are passed at once.

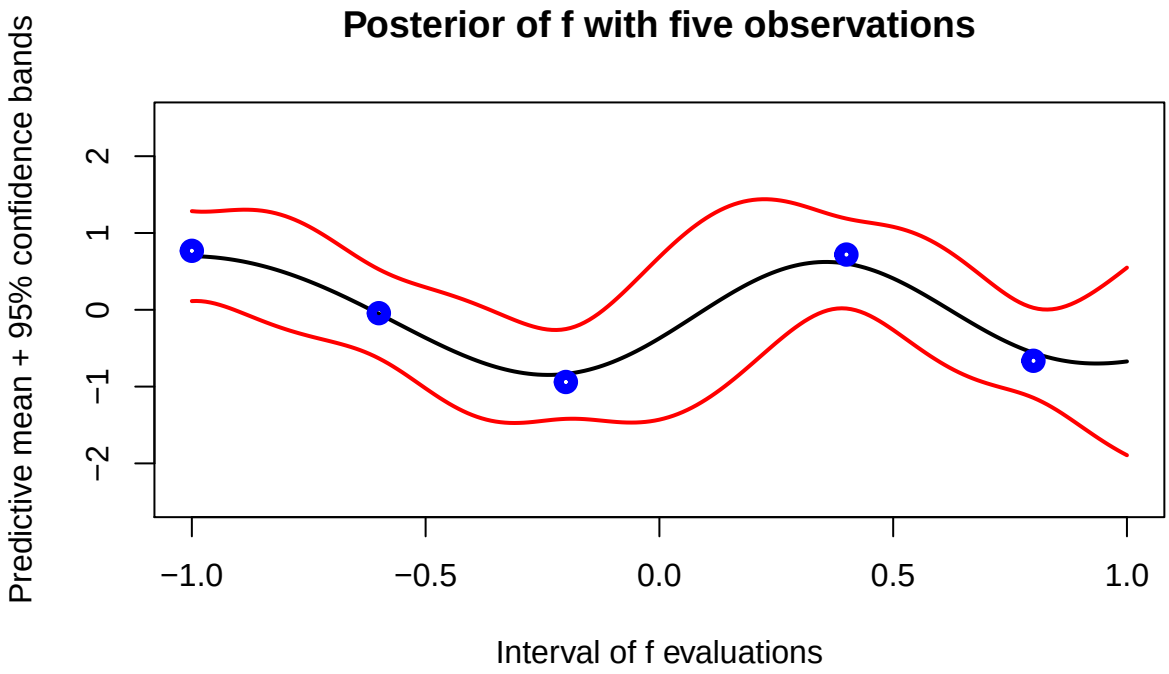
```
X2 <- c(.4,-.6)
y2 <- c(.719,-.044)

pst2 <- posteriorGP(X2, y2, xgrid, sigmaN, sigmaF, ell)
```



Updating the prior with five observations

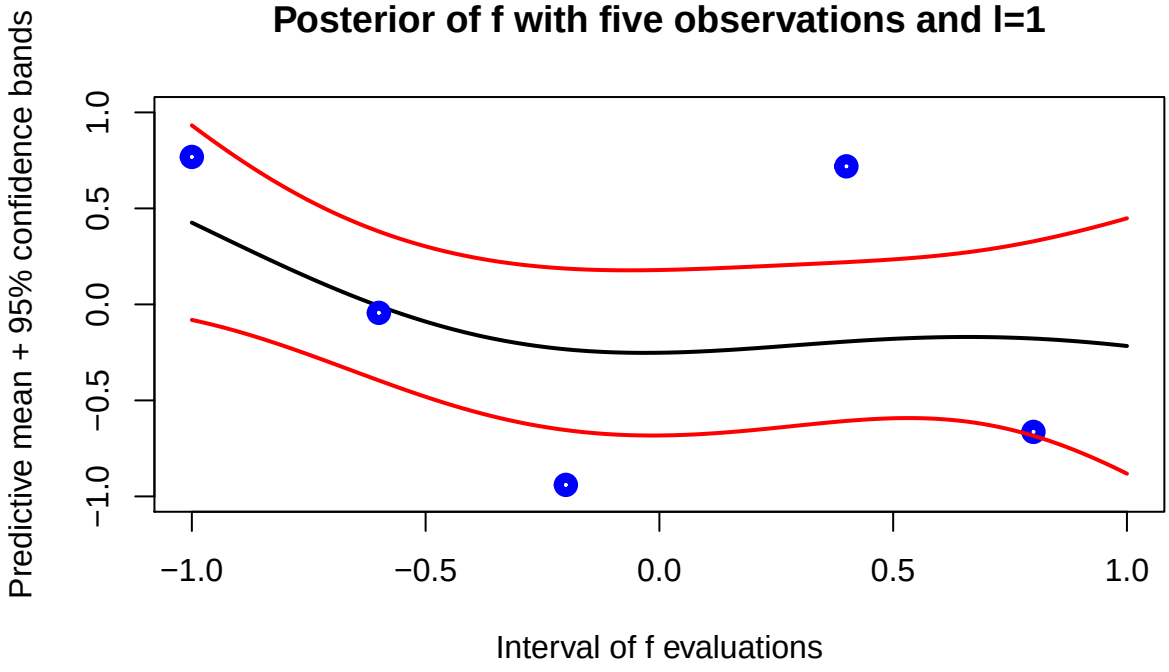
The five observations (including points from part 2) are as follows:

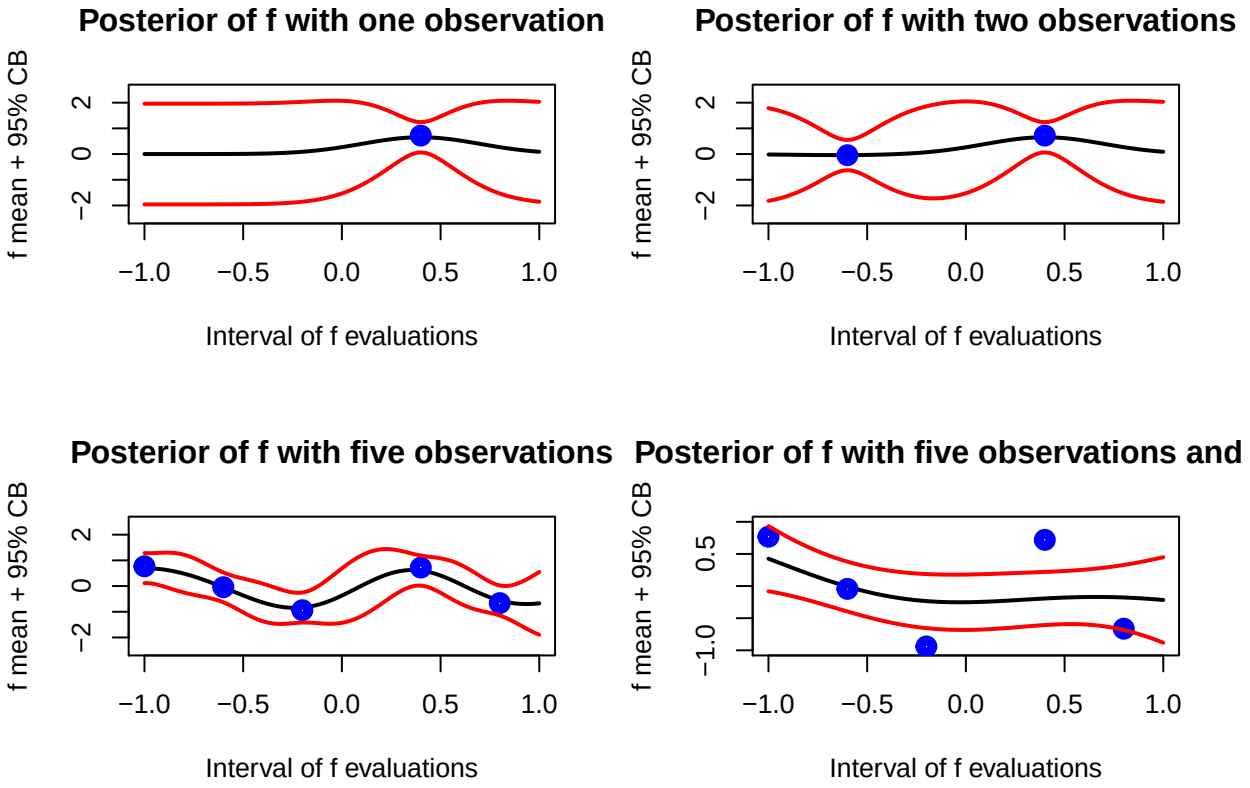
x	y
-1.0	0.768
-0.6	-0.044
-0.2	-0.940
0.4	0.719
0.8	-0.664



Altering hyperparameters

Here we use all the five observations from the previous part and set $\sigma_f = 1$ and $\ell = 1$.





Impact of prior observations with using $\ell = 0.3$

Every time the prior is updated with a new observation, the predictive mean of f is updated accordingly. From the 95% confidence bands, it can be observed that the variance is significantly reduced at observation points. In the setting where we take five observations, the confidence bands completely narrow down as compared to situations where he only have one or two observations.

This implies that with many observations we reduce uncertainty on the whole domain where f is evaluated. However, a model should be **flexible enough** to produce a realistic posterior mean and covariance which is largely controlled by the hyperparameter ℓ.

Impact of $\ell = 1$ on the posterior distribution

ℓ is the smoothing factor in the kernel function. Small ℓ results in a more unsteady covariance and an overfit model. Whereas, large ℓ smooths the covariance between points where in a sense that distant points lose their covariance more slowly.

Comparing the results, we can see $\ell = 1$ creates an underfit model and the confidence bands do not capture 95% of the observations. Also, the predictive mean does not follow the true distribution. This is while $\ell = 0.3$ yields a much more flexible model.

GP Regression with kernlab

The existing *Squared Exponential Kernel* function needs to be restructured to be used with kernlab's function:

```
SEK <- function(sigmaF,l){
  val <- function(x1, x2) {
    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)
  }
  class(val) <- "kernel"
  return(val)
}
```

Now we test the kernel function.

```
## Kernel function evaluated at point 1 and 2:
```

```
##          [,1]
## [1,] 2.426123
```

```
## Covariance matrix for c(1,3,4) and c(2,3,4):
```

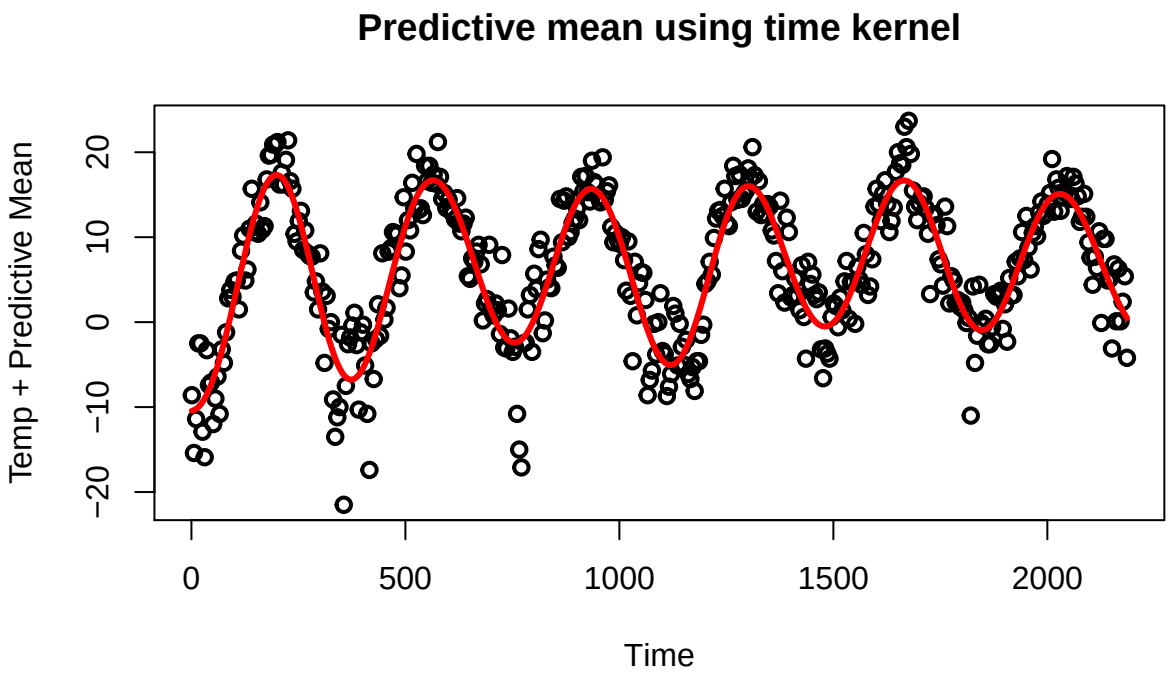
```
## An object of class "kernelMatrix"
##           [,1]      [,2]       [,3]
## [1,] 2.4261226 0.5413411 0.04443599
## [2,] 2.4261226 4.0000000 2.42612264
## [3,] 0.5413411 2.4261226 4.00000000
```

Squared Exponential Kernel with Time

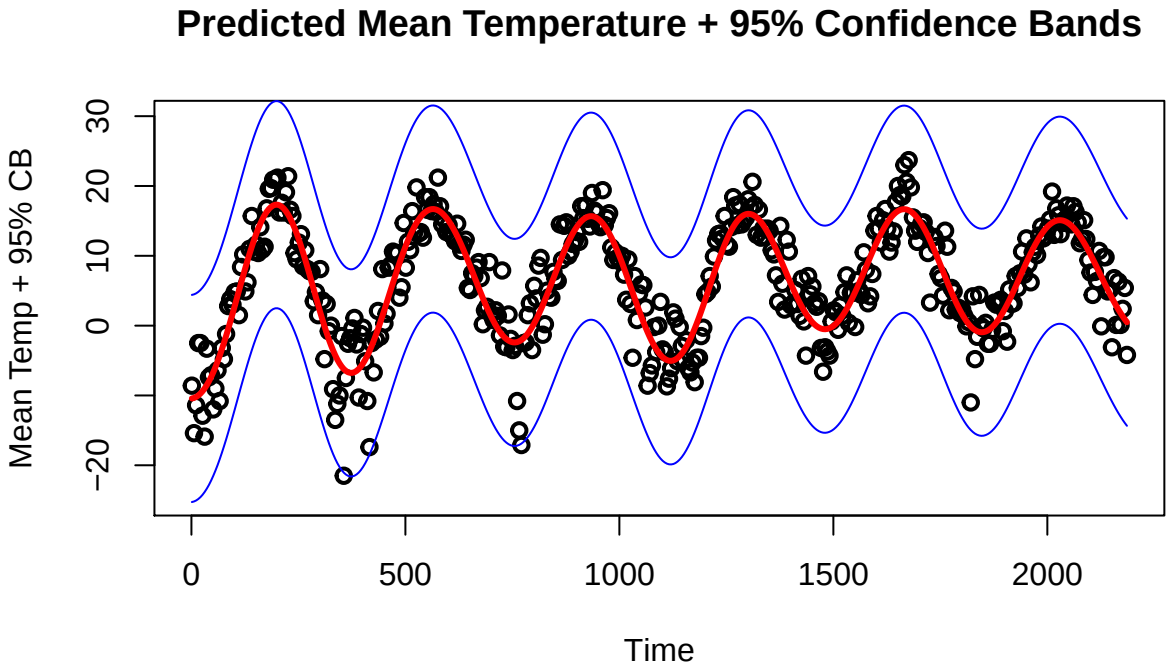
We now attempt to predict the mean temperature using the following parameters:

- Squared Exponential Kernel
- time variable
- $\sigma_f = 20$
- $\ell = 0.2$

```
sigmaF2 <- 20
ell2 <- .2

polynom <- lm(temp~time + I(time^2), data = temps)
sigmaN2 <- sd(polynom$residuals)


GP <- gausspr(temps$time, temps$temp, kernel = SEK,
              kpar = list(sigmaF = sigmaF2, l=ell2),
              var = sigmaN2^2)


meanPred <- predict(GP, temps$time)
```



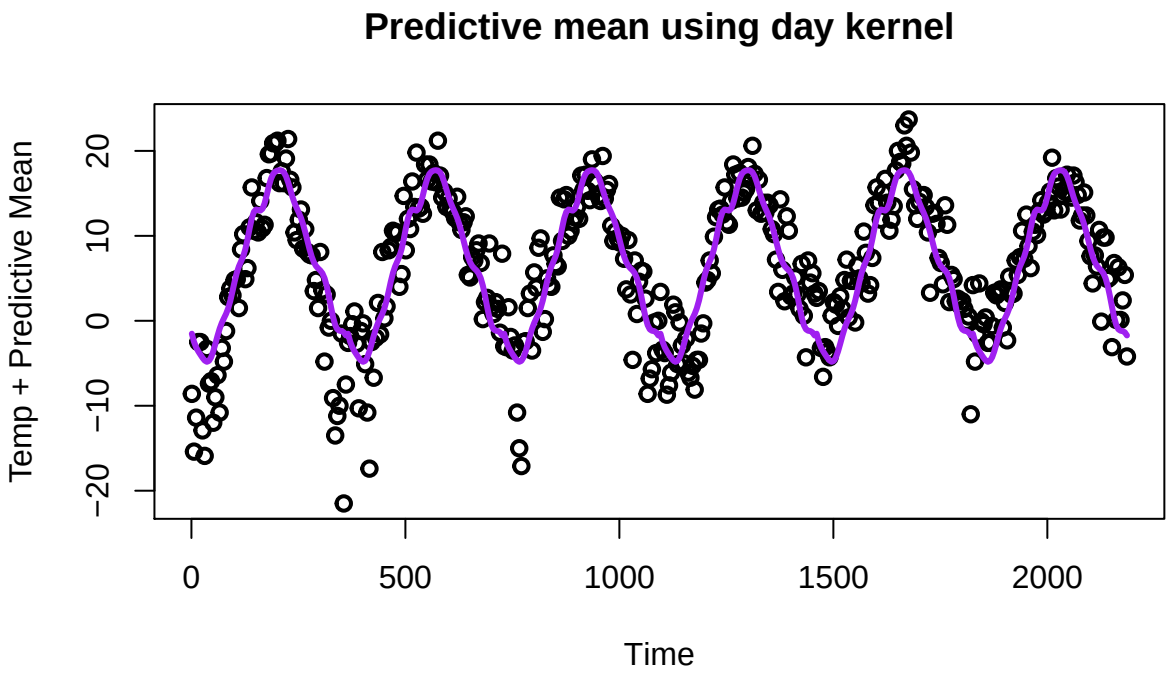
Overlaying 95% Confidence Bands



Squared Exponential Kernel with Day

In this step we predict the mean temperature using the following parameters:

- Squared Exponential Kernel
- day variable
- $\sigma_f = 20$
- $\ell = 0.2$



Periodic Kernel with Time

First step is to define the periodic kernel function.

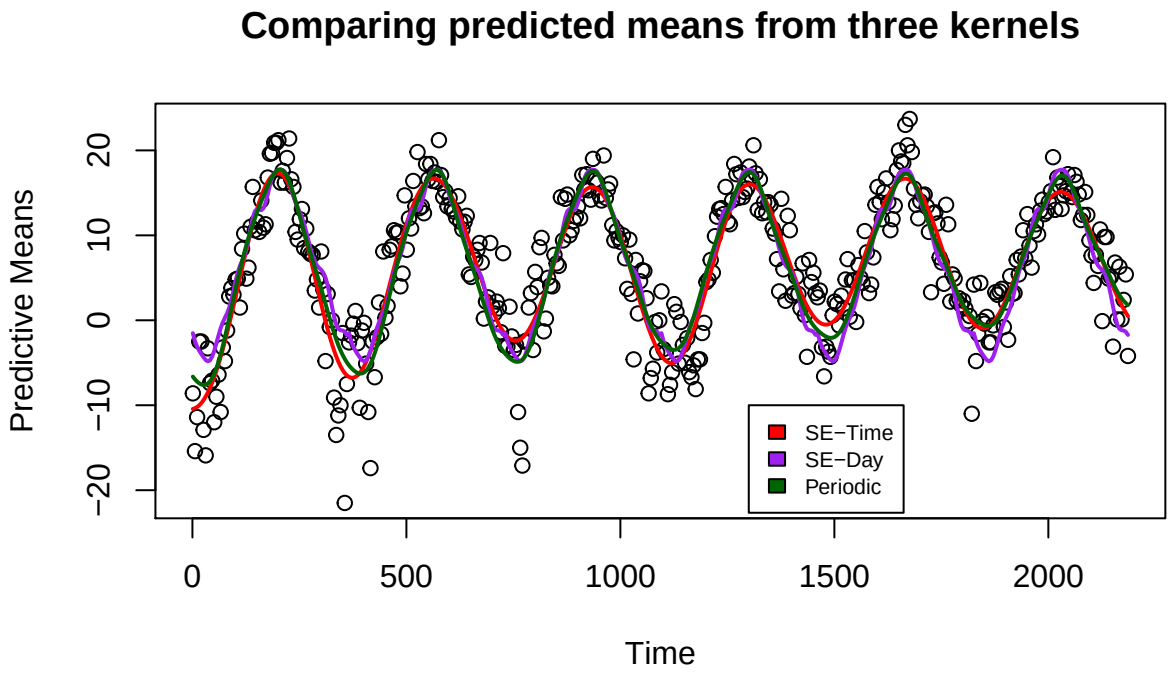
```
PKernel <- function(sigmaF,l1,l2,d) {
  pk <- function(x, xstar) {
    cov <- matrix(NA,length(x), length(xstar))
    for(i in 1:length(xstar)) {
      cov[,i] <- sigmaF^2*exp(-2*(sin(pi*abs(x-xstar[i])/d)/l1)^2)*exp(-.5*(abs(x-xstar[i])/l2)^2)
    }
    return(cov)
  }
  class(pk) <- "kernel"
  return(pk)
}
```

We make new predictions for the temperature means with the following parameters:

- Periodic Kernel
- time variable
- $\sigma_f = 20$
- $\ell_1 = 1$
- $\ell_2 = 10$

- $d = 365/sd(time)$

and plot a comparison of the predicted means from all three kernels.



The predicted mean using the *Squared Exponential Kernel* with *day* variable seem to be overfit to the actual data. This can particularly be observed on the peaks of the predictions. On the other hand, the *periodic* and *Squared Exponential Kernel* with *time* variable has resulted in a smoother prediction and follow one another more closely since they both use the time variable.

It is worth noting that the predicted mean using periodic kernel seem to provide an average of the other two. This could be due to the fact that the first part of the periodic kernel function accounts for periodicity within a year (day) and the second part tries to capture point to point similarities for all years (time) and thus yielding a more accurate prediction.

GP Classification with kernlab

Prediction on train set using two covariates

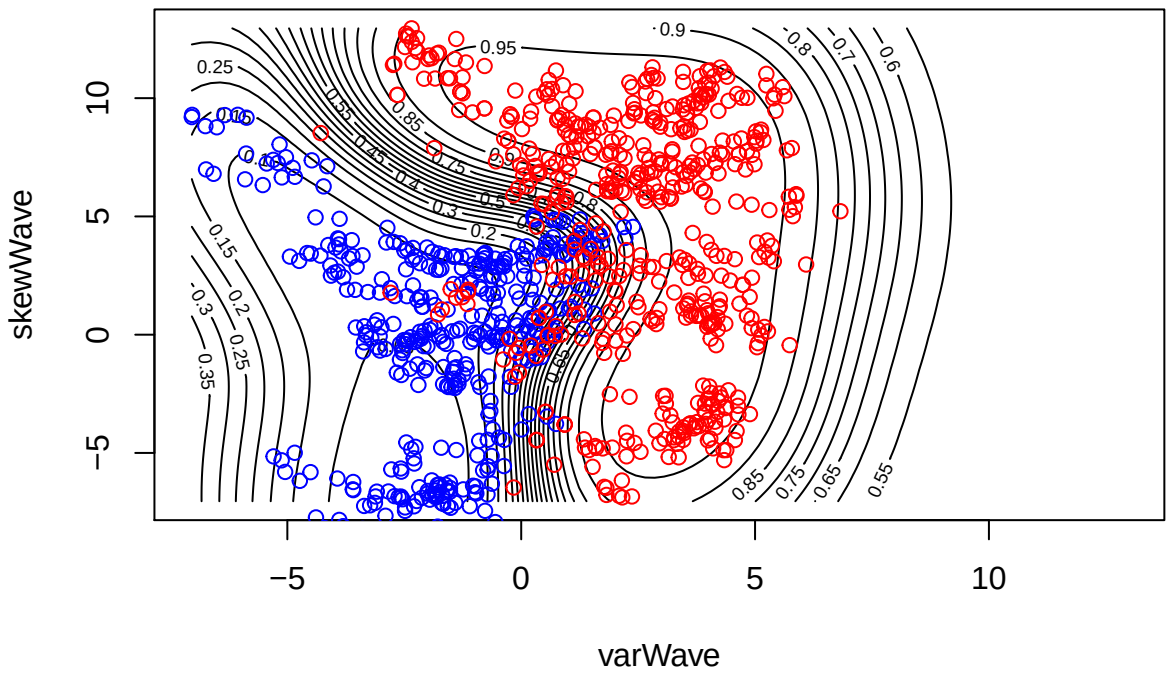
Initially, we divide the data into train and test sets using the provided criterion and fit the model using the train set.

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

```
## Confusion Matrix for train data:
```

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

```
## Accuracy of prediciton on train data:  0.941
```



Prediction on test set using two covariates

```
## Confusion Matrix for test data:

##    predTS
##       0   1
##   0 199  19
##   1   9 145

## Accuracy of prediciton on test data:  0.9247312
```

According to results, the obtained accuracy for train and test are almost identical.

Prediction on test set using all covariates

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

## Confusion Matrix for test data using all covariates:

##    predTS2
##       0   1
##   0 216   2
##   1   0 154

## Accuracy of prediciton on test data using all covariates:  0.9946237
```

The accuracy on the test set increases to 99%, up from 92%, when using all features. Although it is commonly desirable to increase model accuracy, in some case less complex models might be preferred by sacrificing some accuracy. In our case, **varWave** and **skewWave** alone can make an accurate enough classifier while the remaining covariates make significantly less contribution to the model accuracy.

Appendix

```
library(kernlab)
library(AtmRay)
knitr::opts_chunk$set(echo = TRUE)

# Covariance function
SquaredExpKernel <- function(x1,x2,sigmaF,l){
  n1 <- length(x1)
  n2 <- length(x2)
  K <- matrix(NA,n1,n2)
  for (i in 1:n2){
    K[,i] <- sigmaF^2*exp(-0.5*( (x1-x2[i])/l)^2 )
  }
  return(K)
}


posteriorGP <- function(X, y, XStar, sigmaNoise, sigmaF, ell) {
  n <- length(XStar)
  k <- SquaredExpKernel(X,X,sigmaF,ell)
  L <- chol(k + (sigmaNoise * diag(ncol(k))))
  solved <- solve(t(L),y)
  alpha <- solve(L,solved)
  kstar <- SquaredExpKernel(X,XStar,sigmaF,ell)
  f_bar <- t(kstar) %*% alpha   #predictive mean
  v <- solve(t(L),kstar)
  kstarstar <- SquaredExpKernel(XStar, XStar,sigmaF,ell)
  V <- kstarstar - t(v) %*% v   #predictive variance
  #log marginal likelihood
  marg_llik <- .5*(t(y)%*%alpha) - sum(log(diag(L))) - (n/2)*log(2*pi)
  return(list(predictive_mean = f_bar, predictive_variance = V, log_marginal=marg_llik))
}



xgrid <- seq(-1,1,.01)
sigmaN <- .1
ell <- .3
sigmaF <- 1
X1 <- .4
y1 <- .719
pst <- posteriorGP(X1,y1,xgrid,sigmaN, sigmaF, ell)


plot(xgrid, pst$predictive_mean,
     ylim=c(-2.5,2.5), type='l', lwd=2,
     main = 'Posterior of f with one observation',
     xlab = 'Interval of f evaluations',
     ylab = 'Predictive mean + 95% confidence bands')
points(X1,y1, lwd=5, col = 'blue')
lines(xgrid, pst$predictive_mean - 1.96*sqrt(diag(pst$predictive_variance)),
      col='red', lwd=2)
```

```
lines(xgrid, pst$predictive_mean + 1.96*sqrt(diag(pst$predictive_variance)),
      col='red', lwd=2)


X2 <- c(.4,-.6)
y2 <- c(.719,-.044)

pst2 <- posteriorGP(X2, y2, xgrid, sigmaN, sigmaF, ell)


plot(xgrid, pst2$predictive_mean,
     ylim=c(-2.5,2.5), type='l', lwd=2,
     main = 'Posterior of f with two observations',
     xlab = 'Interval of f evaluations',
     ylab = 'Predictive mean + 95% confidence bands')
points(X2,y2, lwd=5, col = 'blue')
lines(xgrid, pst2$predictive_mean - 1.96*sqrt(diag(pst2$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst2$predictive_mean + 1.96*sqrt(diag(pst2$predictive_variance)),
      col='red', lwd=2)


df <- data.frame(x=c(-1,-.6,-.2,.4,.8), y=c(.768,-.044,-.94,.719,-.664))
knitr::kable(df)


X3 <- c(-1,-.6,-.2,.4,.8)
y3 <- c(.768,-.044,-.94,.719,-.664)

pst3 <- posteriorGP(X3, y3, xgrid, sigmaN, sigmaF, ell)


plot(xgrid, pst3$predictive_mean,
     ylim=c(-2.5,2.5), type='l', lwd=2,
     main = 'Posterior of f with five observations',
     xlab = 'Interval of f evaluations',
     ylab = 'Predictive mean + 95% confidence bands')
points(X3,y3, lwd=5, col = 'blue')
lines(xgrid, pst3$predictive_mean - 1.96*sqrt(diag(pst3$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst3$predictive_mean + 1.96*sqrt(diag(pst3$predictive_variance)),
      col='red', lwd=2)


ell2 <- 1
pst4 <- posteriorGP(X3, y3, xgrid, sigmaN, sigmaF, ell2)

plot(xgrid, pst4$predictive_mean, type='l',
     ylim = c(-1,1),lwd=2,
     main = 'Posterior of f with five observations and l=1',
     xlab = 'Interval of f evaluations',
     ylab = 'Predictive mean + 95% confidence bands')
points(X3,y3, lwd=5, col = 'blue')
```

```r
lines(xgrid, pst4$predictive_mean - 1.96*sqrt(diag(pst4$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst4$predictive_mean + 1.96*sqrt(diag(pst4$predictive_variance)),
      col='red', lwd=2)


## Comparing results
par(mfrow=c(2,2))

plot(xgrid, pst$predictive_mean,
     ylim=c(-2.5,2.5), type='l', lwd=2,
     main = 'Posterior of f with one observation',
     xlab = 'Interval of f evaluations',
     ylab = 'f mean + 95% CB')
points(X1,y1, lwd=5, col = 'blue')
lines(xgrid, pst$predictive_mean - 1.96*sqrt(diag(pst$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst$predictive_mean + 1.96*sqrt(diag(pst$predictive_variance)),
      col='red', lwd=2)

plot(xgrid, pst2$predictive_mean,
     ylim=c(-2.5,2.5), type='l', lwd=2,
     main = 'Posterior of f with two observations',
     xlab = 'Interval of f evaluations',
     ylab = 'f mean + 95% CB')
points(X2,y2, lwd=5, col = 'blue')
lines(xgrid, pst2$predictive_mean - 1.96*sqrt(diag(pst2$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst2$predictive_mean + 1.96*sqrt(diag(pst2$predictive_variance)),
      col='red', lwd=2)

plot(xgrid, pst3$predictive_mean,
     ylim=c(-2.5,2.5), type='l', lwd=2,
     main = 'Posterior of f with five observations',
     xlab = 'Interval of f evaluations',
     ylab = 'f mean + 95% CB')
points(X3,y3, lwd=5, col = 'blue')
lines(xgrid, pst3$predictive_mean - 1.96*sqrt(diag(pst3$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst3$predictive_mean + 1.96*sqrt(diag(pst3$predictive_variance)),
      col='red', lwd=2)

plot(xgrid, pst4$predictive_mean, type='l',
     ylim = c(-1,1),lwd=2,
     main = 'Posterior of f with five observations and l=1',
     xlab = 'Interval of f evaluations',
     ylab = 'f mean + 95% CB')
points(X3,y3, lwd=5, col = 'blue')
lines(xgrid, pst4$predictive_mean - 1.96*sqrt(diag(pst4$predictive_variance)),
      col='red', lwd=2)
lines(xgrid, pst4$predictive_mean + 1.96*sqrt(diag(pst4$predictive_variance)),
      col='red', lwd=2)
```

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

temps$time <- seq(1,nrow(temps), 1)
temps$day <- rep(1:365, 6)

temps <- temps[temps$time[seq(1,length(temps$time),5)],]


SEK <- function(sigmaF,l){
  val <- function(x1, x2) {
    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)
  }
  class(val) <- "kernel"
  return(val)
}


ell <- 1
sigmaF <- 2

SEKfunc <- SEK(sigmaF = sigmaF, l = ell)
cat('Kernel function evaluated at point 1 and 2: \n')
SEKfunc(1,2)

X5 <- c(1,3,4)
XStar5 <- c(2,3,4)
K <- kernelMatrix(kernel = SEKfunc, x = X5, y = XStar5)
cat('Covariance matrix for c(1,3,4) and c(2,3,4): \n')
K


sigmaF2 <- 20
ell2 <- .2

polynom <- lm(temp~time + I(time^2), data = temps)
sigmaN2 <- sd(polynom$residuals)


GP <- gausspr(temps$time, temps$temp, kernel = SEK,
              kpar = list(sigmaF = sigmaF2, l=ell2),
              var = sigmaN2^2)


meanPred <- predict(GP, temps$time)
```

```r
plot(temps$time, temps$temp,lwd=2,
     main = 'Predictive mean using time kernel',
     xlab = 'Time', ylab = 'Temp + Predictive Mean')
lines(temps$time, meanPred, col='red', lwd=3)


scaledTemp <- scale(temps$temp)
pst6 <- posteriorGP(temps$time, scaledTemp, temps$time, sigmaN2^2, sigmaF2, ell2)

std <- pst6$predictive_variance

plot(temps$time, temps$temp,lwd=2, ylim = c(-25,30),
     main = 'Predicted Mean Temperature + 95% Confidence Bands',
     xlab = 'Time', ylab = 'Mean Temp + 95% CB')
lines(temps$time, meanPred,type='l',
      col='red', lwd=3)
lines(temps$time, meanPred - 1.96*sqrt(diag(std)),
      col='blue')
lines(temps$time, meanPred + 1.96*sqrt(diag(std)),
      col='blue')


polynom2 <- lm(temp~day + I(day^2), data = temps)
sigmaN3 <- sd(polynom2$residuals)


GP2 <- gausspr(temps$day, temps$temp, kernel = SEK,
               kpar = list(sigmaF = sigmaF2, l=ell2),
               var = sigmaN3^2)


meanPred2 <- predict(GP2, temps$day)

plot(temps$time, temps$temp,lwd=2,
     main = 'Predictive mean using day kernel',
     xlab = 'Time', ylab = 'Temp + Predictive Mean')
lines(temps$time, meanPred2, col='purple', lwd=3)


PKernel <- function(sigmaF,l1,l2,d) {
  pk <- function(x, xstar) {
    cov <- matrix(NA,length(x), length(xstar))
    for(i in 1:length(xstar)) {
      cov[,i] <- sigmaF^2*exp(-2*(sin(pi*abs(x-xstar[i])/d)/l1)^2)*exp(-.5*(abs(x-xstar[i])/l2)^2)
    }
    return(cov)
  }
  class(pk) <- "kernel"
  return(pk)
}


sigmaF3 <- 20
```

```
ell1 <- 1
ell2 <- 10
d <- 365/sd(temps$time)


polynom <- lm(temp~time + I(time^2), data = temps)
sigmaN2 <- sd(polynom$residuals)


GP3 <- gausspr(temps$time, temps$temp, kernel = PKernel,
               kpar = list(sigmaF = sigmaF2, l1=ell1, l2=ell2,d=d),
               var = sigmaN2^2)


meanPred3 <- predict(GP3, temps$time)

plot(temps$time, temps$temp,
     main = 'Comparing predicted means from three kernels',
     xlab = 'Time', ylab = 'Predictive Means')
lines(temps$time, meanPred, col='red', lwd=2)
lines(temps$time, meanPred2, col='purple', lwd=2)
lines(temps$time, meanPred3, col='dark green', lwd=2)
legend(1300,-10, legend = c('SE-Time', 'SE-Day', 'Periodic'),
       fill = c('red','purple','dark green'), cex = .7)


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)

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

GPClass <- gausspr(fraud~varWave+skewWave, data = train)
predTR <- predict(GPClass, train[,1:2])

confTR <- table(train$fraud, predTR)
cat('Confusion Matrix for train data: \n')
confTR

accTR <- sum(diag(confTR))/sum(confTR)
cat('Accuracy of prediciton on train data: ', accTR)


x1 <- seq(min(train[,1]),max(train[,2]),length=100)
x2 <- seq(min(train[,1]),max(train[,2]),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(GPClass, gridPoints, type="probabilities")

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


predTS <- predict(GPClass, test[,1:2])
confTS <- table(test$fraud, predTS)
cat('Confusion Matrix for test data: \n')
confTS

accTS <- sum(diag(confTS))/sum(confTS)
cat('Accuracy of prediciton on test data: ', accTS)


GPClass2 <- gausspr(fraud~., data = train)
predTS2 <- predict(GPClass2, test)

confTS2 <- table(test$fraud, predTS2)
cat('Confusion Matrix for test data using all covariates: \n')
confTS2

accTS2 <- sum(diag(confTS2))/sum(confTS2)
cat('Accuracy of prediciton on test data using all covariates: ', accTS2)
```