AML_Lab4_Group8

Hyungyum Kim, Milda Poceviciute, Boxi Zhang, Fanny Karelius

17 october 2018

Question 2.1

1

For the first exercise, the following Gaussian process regression model will be implemented:

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

The function is called posteriorGP and has 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. $X_\star$.
- hyperParam: Vector with two elements, σ_f and ℓ.
- sigmaNoise: Noise standard deviation σ_n.

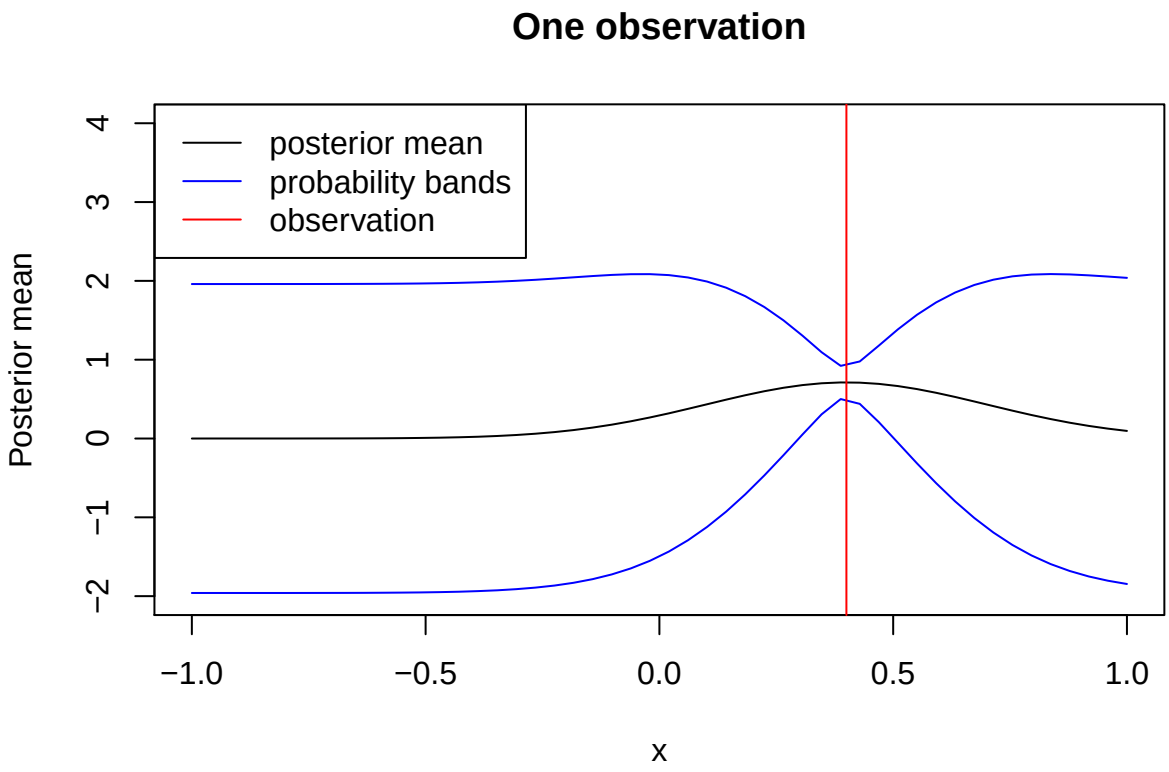
```
SE_kernel <- function(sigmaf, ell){
  val <- function(x1, x2){
    n1 <- length(x1)
    n2 <- length(x2)
    K <- matrix(NA,n1,n2)
    for (i in 1:n2){
      K[,i] <- sigmaf^2*exp(-0.5*( (x1-x2[i])/ell)^2 )
    }
    return(K)
  }
  class(val) <- "kernel"
  return(val)
}

posteriorGP <- function(x, y, xStar, hyperParam, sigmaNoise){
  #### Input:
  #x = train data
  #y = targets
  #xStar = test data
  #hyperParam = list(sigmaF, l)
  #sigmaNoise
  #K covariance for indata, call se_kernel
  y <- as.matrix(y)
  sigmaF <- hyperParam$sigmaF
  l <- hyperParam$l
  kernel <- SE_kernel(sigmaF, l)
  k <- kernel(x, x)
  L <- t(chol(k+sigmaNoise^2*diag(ncol(k))))
  kStar <- kernel(x, xStar)
  alpha <- solve(t(L))%*%(solve(L)%*%y)
  fStar <- t(kStar)%*%alpha #predictive mean
```

```
  v <- solve(L)%*%kStar
  V <- kernel(xStar, xStar)-t(v)%*%v #predictive variance
  #log_marg <- -0.5*t(y)%*%alpha-sum(L) - 0.5*length(x)*log(2*pi)
  #### Output:
  #posterior mean, posterior variance
  return(list("pred_mean"=fStar, "pred_var"=V))
}
```

2

In this part we assume that prior hyper parameters are $\sigma_f = 1$, $\ell = 0.3$, and we update the posterior with observation $(x, y) = (0.4, 0.719)$. Here σ_n is assumed to be 0.1. The plot below shows the observation, the posterior mean, and the 95% probability bands (point wise) plotted over a grid $[-1, 1]$. The probability bands are quite wide for most of the x grid points, except from the location of the observed point (x, y). This observation decreases the uncertainty and the bands become narrower.

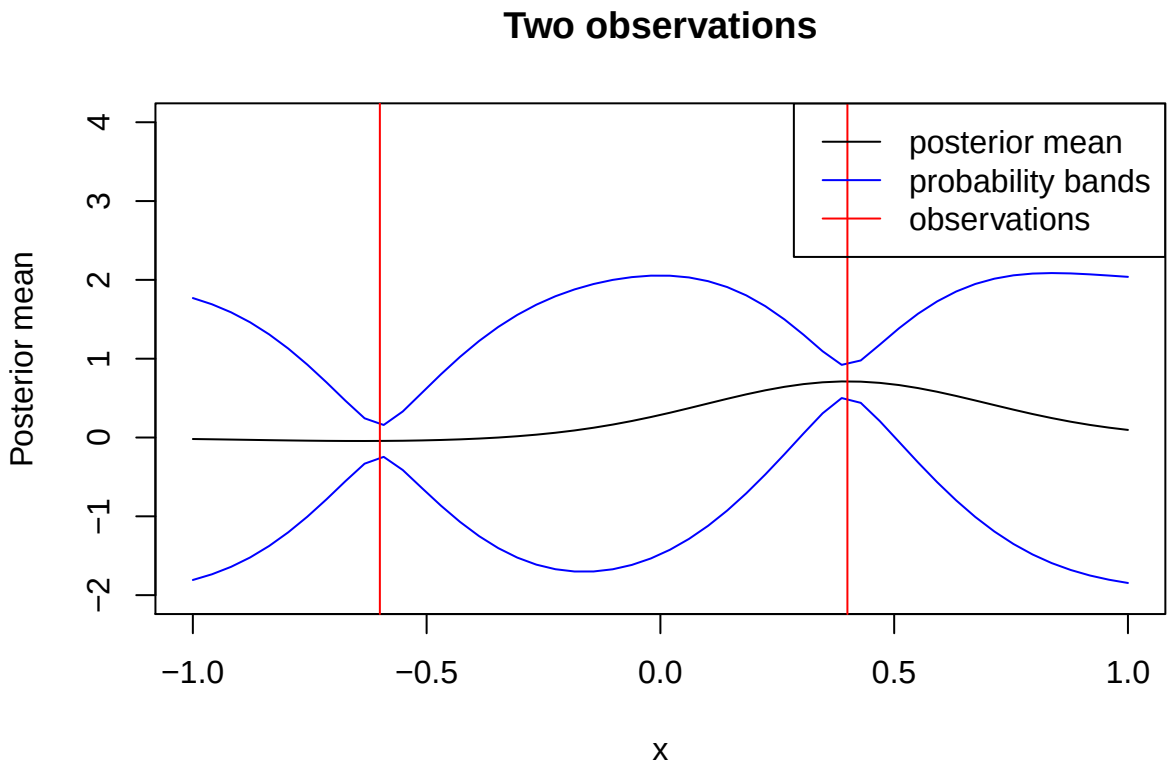
```
noise <- 0.1
hyperParam <- list("sigmaF"=1, "l"=0.3)
xGrid <- seq(-1,1,length=50)

obs1 <- list(x=0.4, y=0.719)
post1<-posteriorGP(obs1$x, obs1$y, xGrid, hyperParam, noise)
confband1 <- conf_band(post1$pred_mean, post1$pred_var)
```



3

The posterior is updated with one more observation $(x, y) = (-0.6, -0.044)$. The plot below shows again the observations, the posterior mean, and the 95% (point wise) probability bands over the grid $[-1, 1]$. This time the uncertainty is smaller for the two observed x values: the probability bands are narrower there.

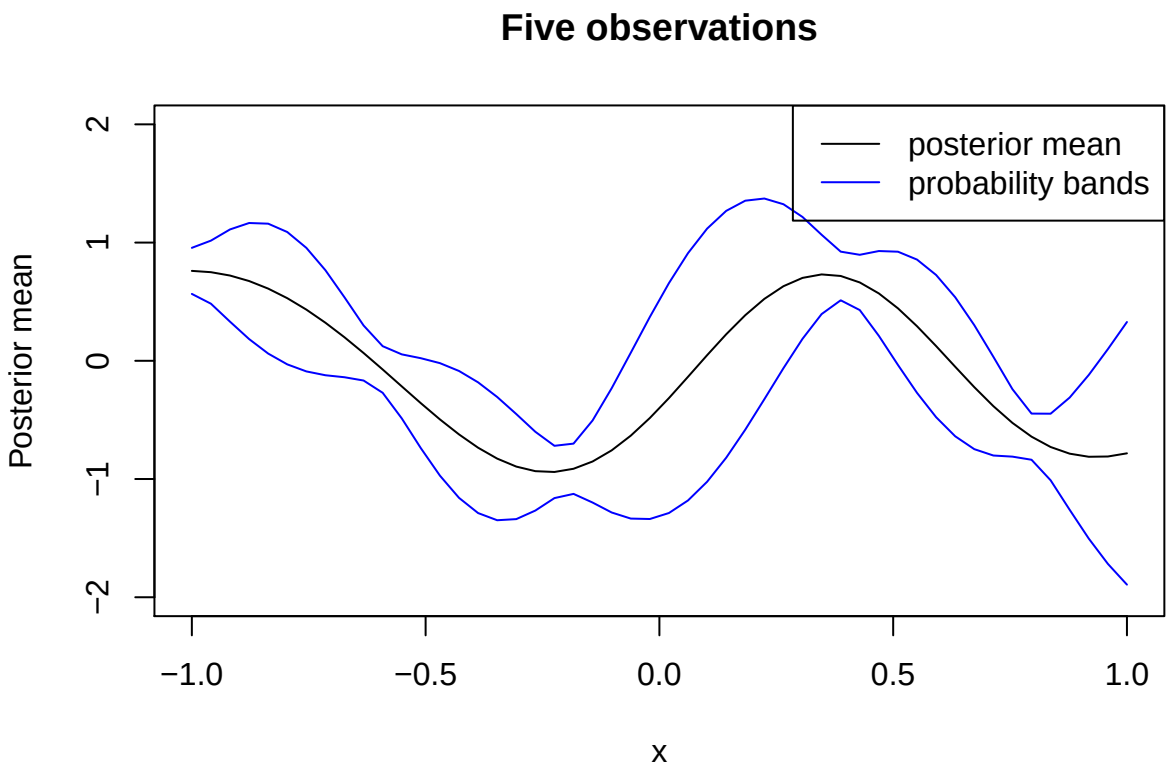
```
obs2 <- list(x=c(obs1$x,-0.6), y=c(obs1$y,-0.044))
post2 <- posteriorGP(obs2$x, obs2$y, xGrid, hyperParam, noise)
confband2 <- conf_band(post2$pred_mean, post2$pred_var)
```



4

The plot shows the posterior mean and the 95% probability bands after 5 observations. The uncertainty has decreased a lot compared to when we only had one and two observations (as indicated by the decreased width of the probability bands).

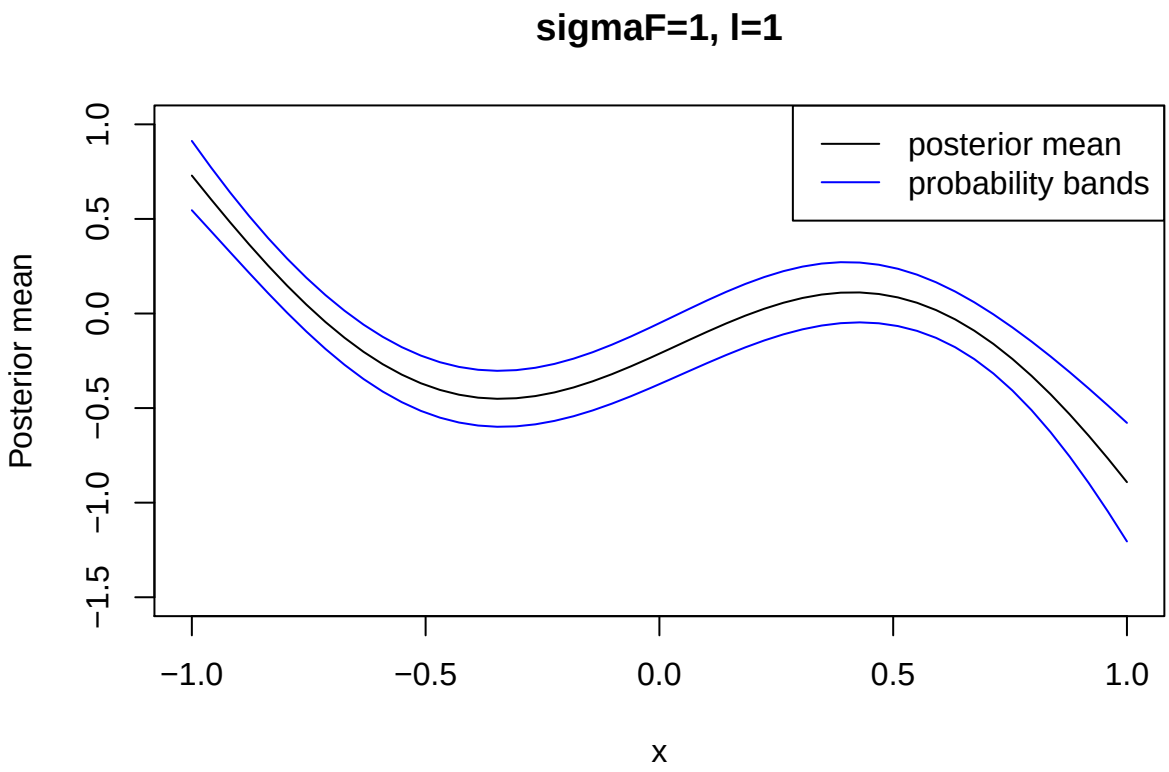
```
obs3 <- list(x=c(obs2$x, -1, -0.2, 0.8), y=c(obs2$y, 0.768, -0.94, -0.664))
post3 <- posteriorGP(obs3$x, obs3$y, xGrid, hyperParam, noise)
confband3 <- conf_band(post3$pred_mean, post3$pred_var)
```



5

The plot below shows the posterior mean and the 95% probability bands after 5 observations as in part 4, but changing the hyper parameters to $\sigma_f = 1$, $\ell = 1$. The higher the ℓ value, the smoother the posterior mean becomes. In the kernel, the distance between the points is divided by ℓ. This means that a higher value of this hyper parameter makes the kernel values converge to σ_f: the distance between the observations becomes less and less important. Hence, the posterior mean becomes more smooth.

```
hyperParam2 <- list(sigmaF=1, l=1)
post4 <- posteriorGP(obs3$x, obs3$y, xGrid, hyperParam2, noise)
confband4 <- conf_band(post4$pred_mean, post4$pred_var)
```



Question 2.2

For this exercise, data of daily mean temperatures in Tullinge are used.

1

```
TempTullinge <- read.csv("TempTullinge.csv", header=TRUE, sep=";")

time <- 1:length(TempTullinge$date)
day <- 1:365
fTime <- seq(from=time[1], to=time[length(time)], by = 5)
fDay <- seq(from=day[1], to=day[length(day)], by = 5)
temp <- TempTullinge$temp[fTime]


kernel_func <- SE_kernel(sigmaf=1, ell=0.3)
point1 <- c(1,2)
kernel_point1 <- kernel_func(point1[1], point1[2])
x_vec <- c(1,3,4)
xStar_vec <- c(2,3,4)
k_cov1<-kernelMatrix(kernel = kernel_func, x = x_vec, y = xStar_vec)
```

For the point $x = 1, x' = 2$, the squared exponential kernel value (with $\sigma_f = 1, \ell = 0.3$) was 0.0038659.

For the vectors $X = (1, 3, 4)^T$ and $X_* = (2, 3, 4)^T$, the covariance matrix (with $\sigma_f = 1, \ell = 0.3$) was

```
## 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
```

2

The following model was used to obtain the posterior mean:

$$temp = f(time) + \epsilon$$

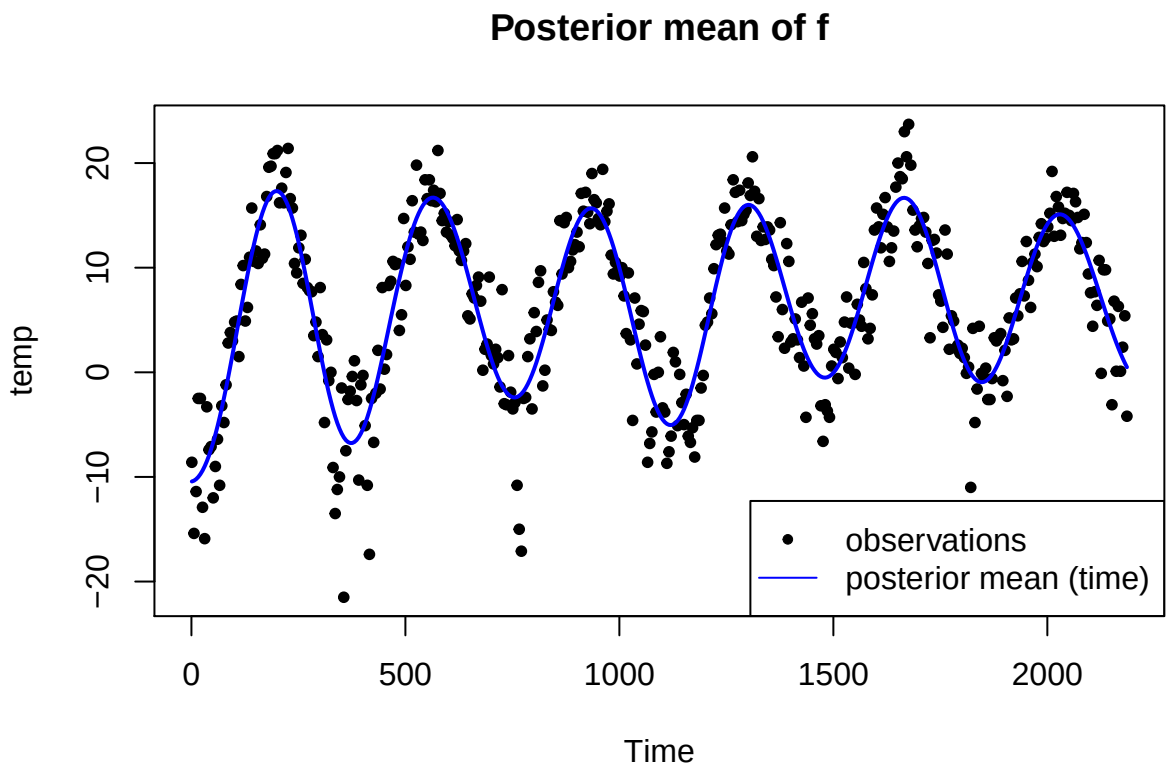
with $\epsilon \sim N(0, \sigma_n^2)$, $f \sim GP(0, k(time, time^{'}))$

σ_n^2 was obtained by taking the variance of the residuals from a quadratic regression of *temp* as a function of *time*.

The hyper parameters used in the squared exponential kernel were $\sigma_f = 20, \ell = 0.2$. A Gaussian process model was fitted on every fifth observation. The posterior mean was calculated by predicting on the training data.

```
lm_fit <- lm(temp~time+time^2, data = TempTullinge)
sigma2n <- var(lm_fit$residuals)
kernel_func2 <- SE_kernel(sigmaf=20, ell=0.2)
gp_model <- gausspr(fTime,temp,data=TempTullinge, kernel=kernel_func2, var=sigma2n)
post_mean <- predict(gp_model, fTime)
```

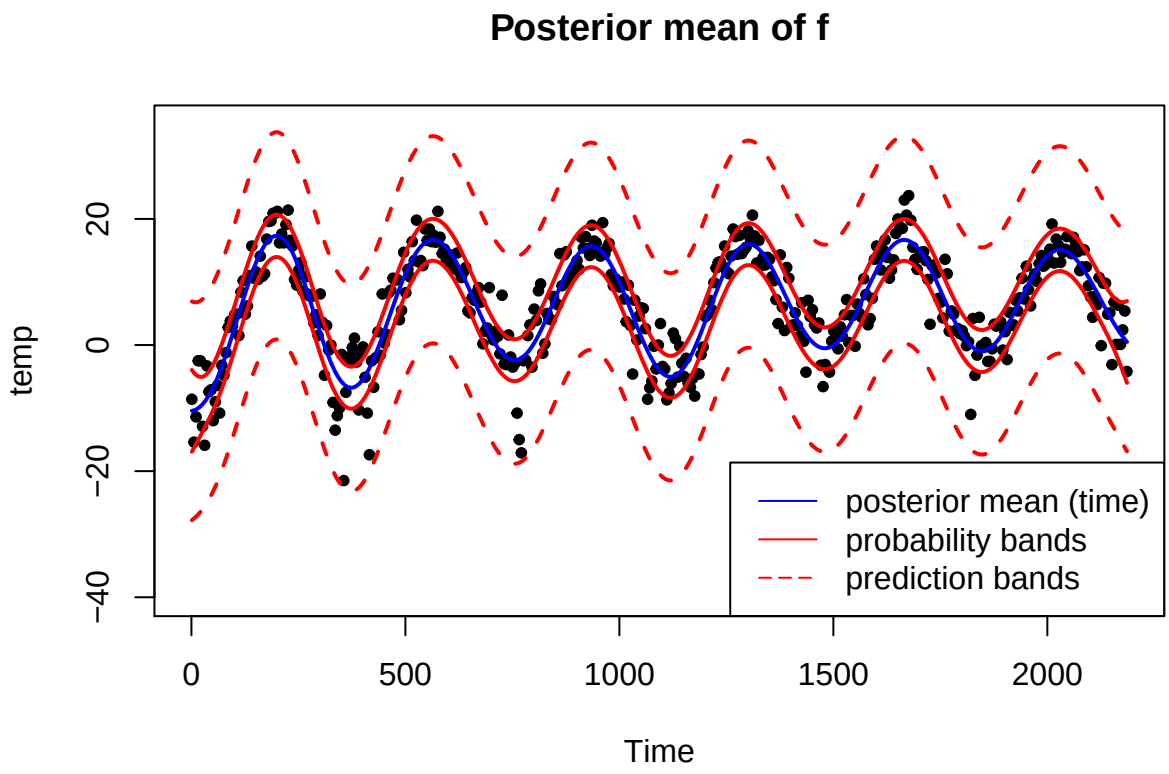
The plot below shows the observations plotted with a fitted posterior mean. The model fits the data quite well as it predicts the underlying trends (periodicity).



3

```
x <- scale(fTime)
xs <- scale(fTime)
n <- length(fTime)
Kss <- kernelMatrix(kernel = kernel_func2, x = xs, y = xs)
Kxx <- kernelMatrix(kernel = kernel_func2, x = x, y = x)
Kxs <- kernelMatrix(kernel = kernel_func2, x = x, y = xs)
Covf = Kss-t(Kxs)%*%solve(Kxx + sigma2n*diag(n), Kxs) # Covariance matrix of fStar
```

From the plot below, we see that the probability and prediction bands fit the data quite well. Though, the prediction bands are quite wide.



4

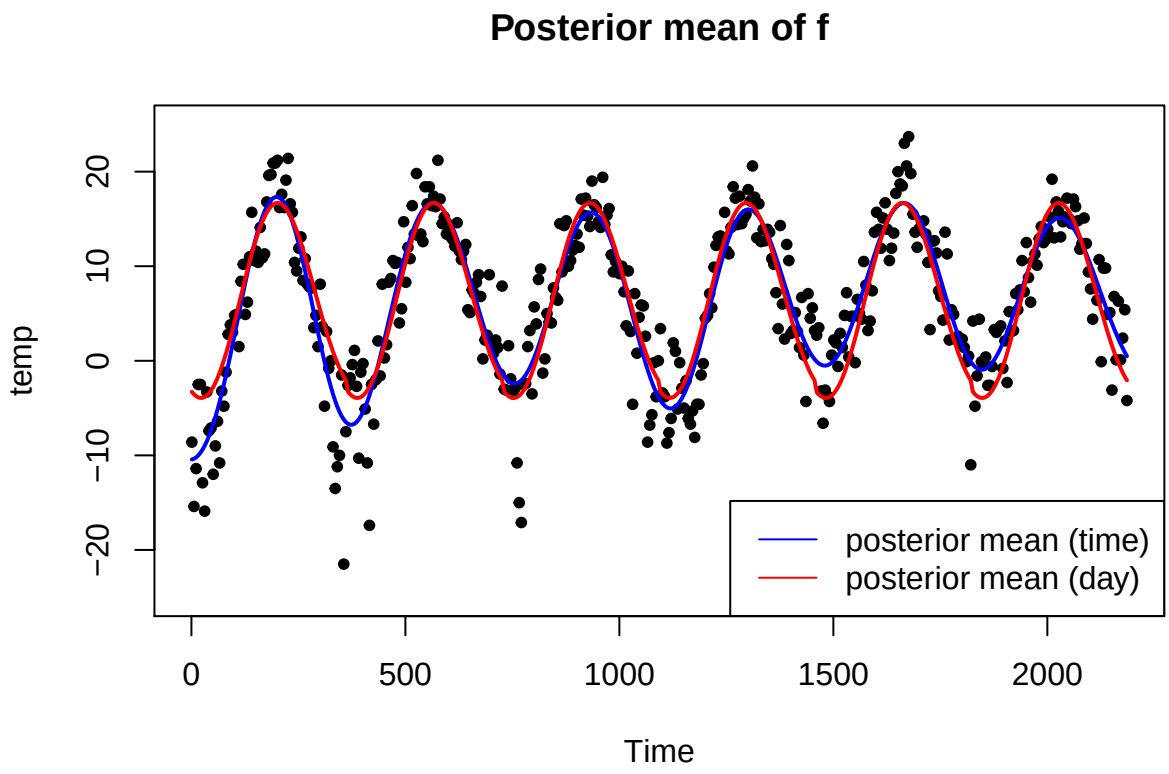
The following model was used to obtain the posterior mean:

$$temp = f(day) + \epsilon, \quad \epsilon \sim N(0, \sigma_n^2), \quad f \sim GP(0, k(day, day'))$$

σ_n^2 was obtained by taking the variance of the residuals from a quadratic regression of *temp* as a function of *day*.

The hyper parameters used in the squared exponential kernel were $\sigma_f = 20, \ell = 0.2$. A Gaussian process model was fitted on every fifth observation. The posterior mean was calculated by predicting on the training data.

```
fDay_rep <- rep(fDay,6)
lm_fit2 <- lm(temp~fDay_rep+fDay_rep^2)
sigma2n_day <- var(lm_fit2$residuals)
kernel_func3 <- SE_kernel(sigmaf=20, ell=1.2)
gp_model2 <- gausspr(fDay_rep,temp,data=TempTullinge, kernel=kernel_func3, var=sigma2n_day)
post_mean2 <- predict(gp_model2, fDay_rep)
```



The model with prediction variable *time* shows a more smooth posterior mean curve than the model with prediction variable *day*. The posterior mean for *day* captures the seasonal trend of the temperature (as the $day = 1, 2, ..., 365$ variable is replicated 6 times), but the posterior mean for *time* captures the overall trend of temperature. Briefly, it is better to use the *time* model to see changes overtime and better to use the *day* model to see general seasonal trends within a year.

5

The GP with a generalization of the periodic kernel is used in this part to model the temperature against time.

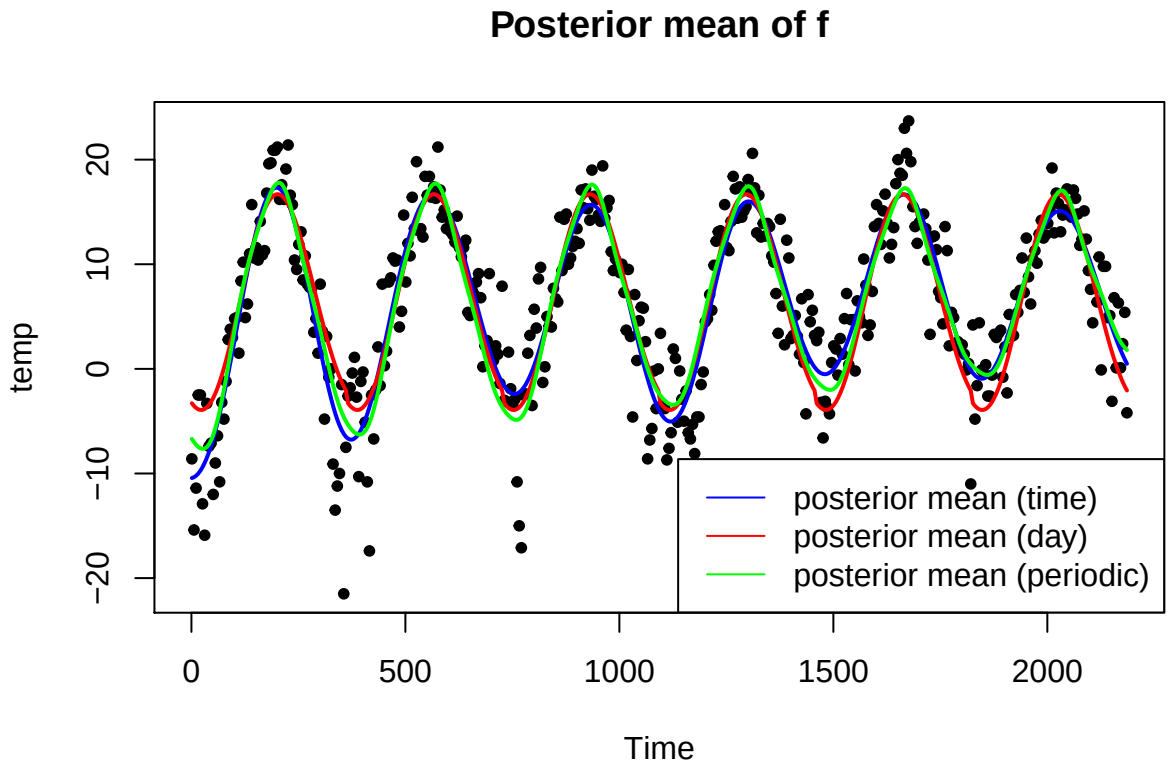
The periodic kernel is given by:

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

The hyper parameters used in the kernel were $\sigma_f = 20, \ell_1 = 1, \ell_2 = 10, d = 365/\text{sd}(time)$.

```
periodic_kernel <- function(sigmaf, ell1, ell2, d){
  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(-2*sin(pi*abs(x1-x2[i])/d)^2/ell1^2)*exp(-0.5*abs(x1-x2[i])^2/ell2^2)
    }
```

```
    return(K)
  }
  class(val) <- "kernel"
  return(val)
}

periodic_k <- periodic_kernel(20, 1, 10, 365/sd(time))
gp_model3 <- gausspr(fTime,temp,data=TempTullinge, kernel=periodic_k, var=sigma2n_day)
post_mean3 <- predict(gp_model3, fTime)
```



From the plot, it seems that the model with periodic kernel captures more within year seasonality than the model with squared exponential kernel with variable *time* and also shows more overall trend of temperature than the model with squared exponential kernel with variable *day*. Since the periodic kernel has one more length scale, which controls the correlation between the same day in different years, this result looks reasonable.

Question 2.3

Banknote fraud data was used to fit a Gaussian process classification model. 1000 observations were used as training data and the rest as test data.

1

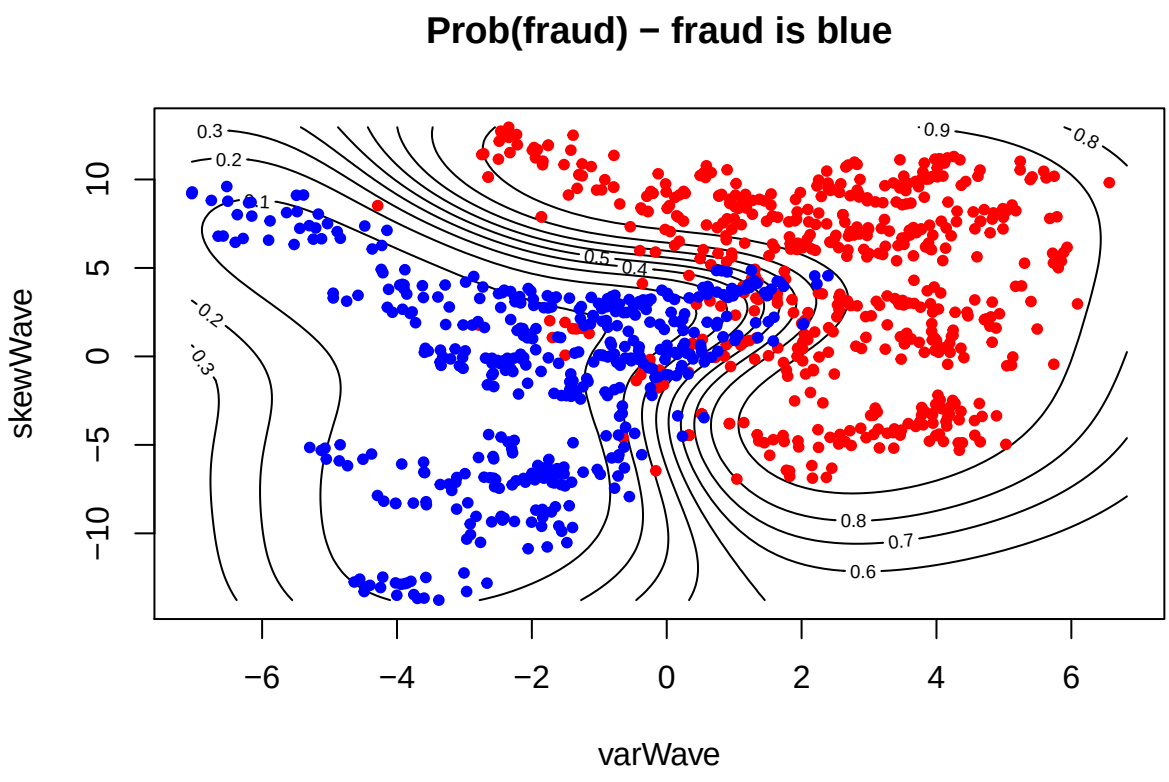
Using the `kernlab` library, a GP classification model was trained on the training data using the covariates $varWave$ and $skewWave$.

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

set.seed(111)
SelectTraining <- sample(1:dim(fraud_data)[1], size = 1000, replace = FALSE)
fraud_train <- fraud_data[SelectTraining,]
fraud_test <- fraud_data[-SelectTraining,]
gp_fraud1 <- gausspr(fraud~varWave+skewWave, data=fraud_train)
```

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

```
x1 <- seq(min(fraud_data$varWave),max(fraud_data$varWave),length=100)
x2 <- seq(min(fraud_data$skewWave),max(fraud_data$skewWave),length=100)
fraud_grid <- meshgrid(x1,x2)
grid_points <- data.frame(cbind(c(fraud_grid$x), c(fraud_grid$y)))
names(grid_points) <- c("varWave","skewWave")
probPreds1 <- predict(gp_fraud1, grid_points, type="probabilities")
```

The contours of prediction probability were plotted over a grid of $varWave$ and $skewWave$ values. The training data was overlayed.



The confusion matrix and accuracy rate are given by

```
##            True
## Prediction   0   1
##          0 512  24
##          1  44 420
```

```
## [1] 0.932
```

2

Using the model obtained, predictions were made on the test data. The confusion matrix and accuracy rate was

```
conf_mat2<-table("Prediction"=predict(gp_fraud1, fraud_test[,1:2]), "True"=fraud_test$fraud)
accuracy2<-sum(diag(conf_mat2))/sum(conf_mat2)
conf_mat2
```

```
##            True
## Prediction   0   1
##          0 191   9
##          1  15 157
```

```
accuracy2
```

```
## [1] 0.9354839
```

The accuracy is 0.9354839, which is pretty close to the accuracy rate obtained on the training data. This is a good indication that the classification works well and that the model is not over fitting the training data.

3

Using all four covariates, a GP classification model was trained on the training data and then predictions were made using the test data.

```
gp_fraud2 <- gausspr(fraud~., data=fraud_train)
conf_mat3<-table("Prediction"=predict(gp_fraud2, fraud_test), "True"=fraud_test$fraud)
accuracy3<-sum(diag(conf_mat3))/sum(conf_mat3)
```

The confusion matrix and accuracy rate was

```
##            True
## Prediction   0   1
##          0 205   0
##          1   1 166
```

```
## [1] 0.9973118
```

Here, the accuracy is very close to 1, and there was only one misclassification. The result with only two covariates was not bad since the accuracy was 0.9354839, but obviously the result is better when all covariates are considered.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(987654321)
library(readr)
library(kernlab)
library(AtmRay)
conf_band <- function(mean, var){
  upp_band <- mean+qnorm(0.975)*sqrt(diag(var))
  low_band <- mean-qnorm(0.975)*sqrt(diag(var))
  return(list(upper=upp_band, lower=low_band))
}
SE_kernel <- function(sigmaf, ell){
  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])/ell)^2 )
    }
    return(K)
  }
  class(val) <- "kernel"
  return(val)
}

posteriorGP <- function(x, y, xStar, hyperParam, sigmaNoise){
  #### Input:
  #x = train data
  #y = targets
  #xStar = test data
  #hyperParam = list(sigmaF, l)
  #sigmaNoise
  #K covariance for indata, call se_kernel
  y <- as.matrix(y)
  sigmaF <- hyperParam$sigmaF
  l <- hyperParam$l
  kernel <- SE_kernel(sigmaF, l)
  k <- kernel(x, x)
  L <- t(chol(k+sigmaNoise^2*diag(ncol(k))))
  kStar <- kernel(x, xStar)
  alpha <- solve(t(L))%*%(solve(L)%*%y)
  fStar <- t(kStar)%*%alpha #predictive mean
  v <- solve(L)%*%kStar
  V <- kernel(xStar, xStar)-t(v)%*%v #predictive variance
  #log_marg <- -0.5*t(y)%*%alpha-sum(L) - 0.5*length(x)*log(2*pi)
  #### Output:
  #posterior mean, posterior variance
  return(list("pred_mean"=fStar, "pred_var"=V))
}

noise <- 0.1
hyperParam <- list("sigmaF"=1, "l"=0.3)
xGrid <- seq(-1,1,length=50)
```

```
obs1 <- list(x=0.4, y=0.719)
post1<-posteriorGP(obs1$x, obs1$y, xGrid, hyperParam, noise)
confband1 <- conf_band(post1$pred_mean, post1$pred_var)

plot(xGrid, post1$pred_mean, type="l", ylim=c(-2,4), ylab="Posterior mean", xlab = "x", main="One obser
lines(xGrid,confband1$upper, col="blue")
lines(xGrid,confband1$lower, col="blue")
abline(v=0.4,col="red")
legend("topleft",c("posterior mean", "probability bands", "observation"),
       col = c("black","blue", "red"), lty = c(1, 1, 1), lwd=c(1, 1, 1))

obs2 <- list(x=c(obs1$x,-0.6), y=c(obs1$y,-0.044))
post2 <- posteriorGP(obs2$x, obs2$y, xGrid, hyperParam, noise)
confband2 <- conf_band(post2$pred_mean, post2$pred_var)

plot(xGrid, post2$pred_mean, type="l", ylim=c(-2,4), ylab="Posterior mean", xlab = "x", main="Two obser
lines(xGrid,confband2$upper, col="blue")
lines(xGrid,confband2$lower, col="blue")
abline(v=0.4,col="red")
abline(v=-0.6,col="red")
legend("topright",c("posterior mean", "probability bands", "observations"),
       col = c("black","blue","red"), lty = c(1, 1, 1), lwd=c(1, 1, 1))

obs3 <- list(x=c(obs2$x, -1, -0.2, 0.8), y=c(obs2$y, 0.768, -0.94, -0.664))
post3 <- posteriorGP(obs3$x, obs3$y, xGrid, hyperParam, noise)
confband3 <- conf_band(post3$pred_mean, post3$pred_var)

plot(xGrid, post3$pred_mean, type="l", ylim=c(-2,2), ylab="Posterior mean", xlab = "x", main="Five obse
lines(xGrid,confband3$upper, col="blue")
lines(xGrid,confband3$lower, col="blue")
legend("topright",c("posterior mean", "probability bands"),
       col = c("black","blue"), lty = c(1, 1), lwd=c(1, 1))

hyperParam2 <- list(sigmaF=1, l=1)
post4 <- posteriorGP(obs3$x, obs3$y, xGrid, hyperParam2, noise)
confband4 <- conf_band(post4$pred_mean, post4$pred_var)

plot(xGrid, post4$pred_mean, type="l", ylim=c(-1.5,1), ylab="Posterior mean", xlab = "x", main="sigmaF=
lines(xGrid,confband4$upper, col="blue")
lines(xGrid,confband4$lower, col="blue")
legend("topright",c("posterior mean", "probability bands"),
       col = c("black","blue"), lty = c(1, 1), lwd=c(1, 1))

TempTullinge <- read.csv("TempTullinge.csv", header=TRUE, sep=";")

time <- 1:length(TempTullinge$date)
day <- 1:365
fTime <- seq(from=time[1], to=time[length(time)], by = 5)
fDay <- seq(from=day[1], to=day[length(day)], by = 5)
temp <- TempTullinge$temp[fTime]


kernel_func <- SE_kernel(sigmaf=1, ell=0.3)
```

```r
point1 <- c(1,2)
kernel_point1 <- kernel_func(point1[1], point1[2])
x_vec <- c(1,3,4)
xStar_vec <- c(2,3,4)
k_cov1<-kernelMatrix(kernel = kernel_func, x = x_vec, y = xStar_vec)
k_cov1

lm_fit <- lm(temp~time+time^2, data = TempTullinge)
sigma2n <- var(lm_fit$residuals)
kernel_func2 <- SE_kernel(sigmaf=20, ell=0.2)
gp_model <- gausspr(fTime,temp,data=TempTullinge, kernel=kernel_func2, var=sigma2n)
post_mean <- predict(gp_model, fTime)

plot(fTime, temp, pch=20, main="Posterior mean of f", xlab="Time")
lines(fTime,post_mean, col="blue", lwd=2)
legend("bottomright",c("observations", "posterior mean (time)"),
       col = c("black","blue"), pch=c(20,-1), lty = c(0, 1), lwd=c(0, 1))

x <- scale(fTime)
xs <- scale(fTime)
n <- length(fTime)
Kss <- kernelMatrix(kernel = kernel_func2, x = xs, y = xs)
Kxx <- kernelMatrix(kernel = kernel_func2, x = x, y = x)
Kxs <- kernelMatrix(kernel = kernel_func2, x = x, y = xs)
Covf = Kss-t(Kxs)%*%solve(Kxx + sigma2n*diag(n), Kxs) # Covariance matrix of fStar

# Probability intervals for fStar
plot(fTime, temp, ylim=c(-40,35), pch=20, main="Posterior mean of f", xlab="Time")
lines(fTime,post_mean, col="blue", lwd=2)
lines(fTime, post_mean - 1.96*sqrt(diag(Covf)), col = "red", lwd=2)
lines(fTime, post_mean + 1.96*sqrt(diag(Covf)), col = "red", lwd=2)
# Prediction intervals for yStar
lines(fTime, post_mean - 1.96*sqrt((diag(Covf) + sigma2n)), col = "red", lwd=2, lty=2)
lines(fTime, post_mean + 1.96*sqrt((diag(Covf) + sigma2n)), col = "red", lwd=2, lty=2)
legend("bottomright",c("posterior mean (time)", "probability bands","prediction bands"),
       col = c("blue","red", "red"), lty = c(1, 1, 5), lwd=c(1, 1, 1))

fDay_rep <- rep(fDay,6)
lm_fit2 <- lm(temp~fDay_rep+fDay_rep^2)
sigma2n_day <- var(lm_fit2$residuals)
kernel_func3 <- SE_kernel(sigmaf=20, ell=1.2)
gp_model2 <- gausspr(fDay_rep,temp,data=TempTullinge, kernel=kernel_func3, var=sigma2n_day)
post_mean2 <- predict(gp_model2, fDay_rep)

plot(fTime, temp, ylim=c(-25,25), pch=20, main="Posterior mean of f", xlab="Time")
lines(fTime,post_mean, col="blue", lwd=2)
lines(fTime,post_mean2, col="red", lwd=2)
legend("bottomright",c("posterior mean (time)","posterior mean (day)"),
       col = c("blue","red"), lty = c(1,1), lwd=c(1,1))

periodic_kernel <- function(sigmaf, ell1, ell2, d){
  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(-2*sin(pi*abs(x1-x2[i])/d)^2/ell1^2)*exp(-0.5*abs(x1-x2[i])^2/ell2^2)
    }
    return(K)
  }
  class(val) <- "kernel"
  return(val)
}

periodic_k <- periodic_kernel(20, 1, 10, 365/sd(time))
gp_model3 <- gausspr(fTime,temp,data=TempTullinge, kernel=periodic_k, var=sigma2n_day)
post_mean3 <- predict(gp_model3, fTime)

plot(fTime, temp, pch=20, main="Posterior mean of f", xlab="Time")
lines(fTime,post_mean, col="blue", lwd=2)
lines(fTime,post_mean2, col="red", lwd=2)
lines(fTime,post_mean3, col="green", lwd=2)
legend("bottomright",c("posterior mean (time)","posterior mean (day)","posterior mean (periodic)"),
       col = c("blue","red","green"), lty = c(1, 1, 1), lwd=c(1, 1, 1))

fraud_data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/Code/banknot
names(fraud_data) <- c("varWave","skewWave","kurtWave","entropyWave","fraud")
fraud_data[,5] <- as.factor(fraud_data[,5])

set.seed(111)
SelectTraining <- sample(1:dim(fraud_data)[1], size = 1000, replace = FALSE)
fraud_train <- fraud_data[SelectTraining,]
fraud_test <- fraud_data[-SelectTraining,]
gp_fraud1 <- gausspr(fraud~varWave+skewWave, data=fraud_train)
x1 <- seq(min(fraud_data$varWave),max(fraud_data$varWave),length=100)
x2 <- seq(min(fraud_data$skewWave),max(fraud_data$skewWave),length=100)
fraud_grid <- meshgrid(x1,x2)
grid_points <- data.frame(cbind(c(fraud_grid$x), c(fraud_grid$y)))
names(grid_points) <- c("varWave","skewWave")
probPreds1 <- predict(gp_fraud1, grid_points, type="probabilities")

contour(x1,x2,matrix(probPreds1[,1], 100, byrow = TRUE), xlab = "varWave", ylab = "skewWave", main = 'Pr
points(fraud_train[fraud_train$fraud==0,"varWave"],fraud_train[fraud_train$fraud==0,"skewWave"],col="re
points(fraud_train[fraud_train$fraud==1,"varWave"],fraud_train[fraud_train$fraud==1,"skewWave"],col="blu

conf_mat1<-table("Prediction"=predict(gp_fraud1, fraud_train[,1:2]), "True"=fraud_train$fraud)
accuracy1<-sum(diag(conf_mat1))/sum(conf_mat1)

conf_mat1
accuracy1

conf_mat2<-table("Prediction"=predict(gp_fraud1, fraud_test[,1:2]), "True"=fraud_test$fraud)
accuracy2<-sum(diag(conf_mat2))/sum(conf_mat2)
conf_mat2
accuracy2
gp_fraud2 <- gausspr(fraud~., data=fraud_train)
```

```
conf_mat3<-table("Prediction"=predict(gp_fraud2, fraud_test), "True"=fraud_test$fraud)
accuracy3<-sum(diag(conf_mat3))/sum(conf_mat3)

conf_mat3
accuracy3
```