# portfolio_7

2023-04-20

## Gaussian Process Regression

### Task 1

#### Question 1

We have that the function is a kernel if it satisfies Mercers theorum, let $K_{i,j} = k(x_i, x_j) = g(x_i)g(x_j)$ be the kernel matrix. We have that K is symmetric as:

$$K_{i,j} = g(x_i)g(x_j) = g(x_j)g(x_i) = K_{j,i}$$

We can also show that K is postitive semi-definite, first let $c = (c_1, ..., c_n) \in \mathbb{R}^n$, then,

$$\begin{aligned}
\sum_i \sum_j K_{i,j} c_i c_j &= \sum_i \sum_j c_i c_j g(x_i) g(x_j) \\
&= (\sum_i c_i g(x_i))(\sum_j c_j g(x_j)) \\
&= (\sum_i c_i g(x_i))^2 \\
&\geq 0
\end{aligned}$$

Therefore K is postitive semi-definite, as K is both symmetric and postive semi-definite by Mercers theorum is a kernel.

#### Question 2

Again let $K$ denote the kernel matrix, then,

$$K_{i,j} = k(x_i, x_j) = a$$

K is symmetric as,

$$K_{i,j} = k(x_i, x_j) = a = k(x_j, x_i) = K_{j,i}$$

And we have its postitive semi-definite as for any $c = (c_1, ..., c_n)^T \in \mathbb{R}^n$ we have,

$$\begin{aligned}
\sum_i \sum_j K_{i,j} c_i c_j &= \sum_i \sum_j c_i c_j a \\
&= a \cdot (\sum_i c_i)(\sum_j c_j) \\
&= a \cdot (\sum_i c_i)^2 \\
&\geq 0
\end{aligned}$$

hence by Mercers theorum K is a kernel.

#### Question 3

Again let K denote the kernel matrix, we have,

$$K_{i,j} = \sum_{l=1}^{m} c_l k_l(x_i, x_j)$$

We can also show that K is postitive semidefinite, as each kernel k is symmetric (by Mercers theorum),

$$\begin{aligned} K_{i,j} &= \sum_{l=1}^{m} c_l k_l(x_i, x_j) \\ &= \sum_{l=1}^{m} c_l k_l(x_j, x_i) \\ &= K_{j,i} \end{aligned}$$

We can also show that K is positive semi-definite, as each kernel k is (by Mercers theorum) postiive semidefinite, let $\lambda = (\lambda_1, ..., \lambda_n) \in \mathbb{R}^n$, then,

$$\begin{aligned} \sum_i \sum_j K_{i,j}\lambda_i\lambda_j &= \sum_i \sum_j \lambda_i\lambda_j \sum_l c_l k_l(x_i, x_j) \\ &= \sum_l c_l \sum_i \sum_j \lambda_i\lambda_j k_l(x_i, x_j) \\ &\geq 0 \end{aligned}$$

as $c_l \geq 0$ and $\sum_i \sum_j \lambda_i\lambda_j k_l(x_i, x_j) \geq 0$ for all $l \in \{1, ..., m\}$ therefore K is postitive semi-definite, therefore k is a kernel by Mercers theourm.

#### Question 4

Let K denote the kernel matrix, if we let $x_i, x_j \in \chi$ for all $i, j \in \{1, ..., n\}$, then

$$K_{i,j} = k(x_i, x_j)1_{\chi\times\chi}(x_i, x_j) = k(x_i, x_j)$$

K is symmetric as,

$$K_{i,j} = k(x_i, x_j) = k(x_j, x_i)$$

since $x_i, x_j \in \mathbb{R}^p$ and k is a kernel on $\mathbb{R}^p$, similarly we have for some $c = (c_1, ..., c_n) \in \mathbb{R}^p$ that,

$$\sum_i \sum_j c_i c_j K_{i,j} = \sum_i \sum_j c_i c_j k(x_i, x_j) \geq 0$$

Hence K is symmetric and postitive semi-definite and therefore k a kernel when restricted on $\chi \times \chi$.

## Task 2

For this task we will use the bone mineral dataset, let's load it in and have a look at some of it:

```
BMD <- read.csv("portfolio_7_data/spnbmd.csv")
head(BMD)
```

```
##   idnum ethnic  age sex spnbmd
## 1     1  White 11.2 mal  0.719
## 2     1  White 12.2 mal  0.732
## 3     1  White 13.2 mal  0.776
## 4     1  White 14.3 mal  0.781
## 5     2  White 12.7 mal  0.620
## 6     2  White 13.8 mal  0.627
```

Let's now print some summary statistics for the dataset:

```
summary(BMD)
```

```
##      idnum          ethnic               age             sex
##  Min.   :  1.0   Length:1003        Min.   : 8.80   Length:1003
##  1st Qu.: 84.5   Class :character   1st Qu.:12.80   Class :character
##  Median :181.0   Mode  :character   Median :15.70   Mode  :character
##  Mean   :189.9                      Mean   :16.32
##  3rd Qu.:287.5                      3rd Qu.:19.45
##  Max.   :429.0                      Max.   :26.20
##      spnbmd
##  Min.   :0.5360
##  1st Qu.:0.7995
##  Median :0.9650
##  Mean   :0.9476
##  3rd Qu.:1.0705
##  Max.   :1.4430
```

We now are going to add the rate of change as we would like to model how the relative change in spinal Bone Mineral Density (BMD) changes with age:

```
BMD <- BMD[order(BMD$id, BMD$age),]
BMD$sex <- as.factor(BMD$sex)
BMD$rc <- NA

for (id in as.numeric(BMD$idnum)) {
  BMD.dash <- BMD[BMD$idnum == id,]
  if (nrow(BMD.dash) > 1) {
    spnbmd_diff <- diff(BMD.dash$spnbmd)
    rc <- c(NA, spnbmd_diff / BMD.dash$spnbmd[1:(nrow(BMD.dash)-1)])
    BMD[BMD$idnum == id, "rc"] <- rc
  }
}

BMD <- BMD[complete.cases(BMD),]
head(BMD)
```

```
##   idnum ethnic  age sex spnbmd          rc
## 2     1  White 12.2 mal  0.732 0.018080668
## 3     1  White 13.2 mal  0.776 0.060109290
## 4     1  White 14.3 mal  0.781 0.006443299
## 6     2  White 13.8 mal  0.627 0.011290323
## 7     2  White 14.8 mal  0.759 0.210526316
## 8     2  White 15.8 mal  0.790 0.040843215
```

Excellent, let's now get into it, we want to fit a Gaussian process regression model with known variance $\sigma^2 = \lambda$. I will be using the Gaussian kernel as I would like to fit a continuous non-constant function so our parameter vector is simply $\psi = \gamma$ the bandwidth parameter. Now we would like to compute the posterior distribution of f given the observations $y^0_{1:n}$. To do this we will choose $\lambda$ and $\gamma$ using empirical bayes:

```
library(kernlab)

# Define some variables that we are going to use in our function
X <- BMD$age
y <- BMD$rc
n <- nrow(BMD)
```

```r
# Function that calculates the negative marginal log likelihood
nmll <- function(par){
  lambda <- par[1]
  gamma <- par[2]
  rbf <- rbfdot(sigma = gamma)
  gram <- kernelMatrix(rbf, X)
  0.5* log(det(gram + lambda* diag(n) + 1e-6 * diag(n))) + 0.5* t(y) %*% solve(gram + lambda* diag(n) +
}

# Set our search parameters
lower <- c(0.01, 0.01)
upper <- c(3, 4)
n_searches <- 5

# Conduct multiple searches with different initial parameters
set.seed(123)
results <- data.frame(matrix(nrow = n_searches, ncol = 3))
colnames(results) <- c("init_param_1", "init_param_2", "max_nmll")
for (i in 1:n_searches) {
  init_params <- runif(2, lower, upper)
  # Use optim to minimize the negative marginal log likelihood
  result <- optim(init_params, nmll)
  results[i,] <- c(init_params[1], init_params[2], -result$value)
}
```

```
## Warning in log(det(gram + lambda * diag(n) + 1e-06 * diag(n))): NaNs produced
```

```r
print(results)
```

```
##   init_param_1 init_param_2 max_nmll
## 1    0.8698568    3.1553375 369.6367
## 2    1.2328410    3.5332394 369.4803
## 3    2.8219972    0.1917704 369.8159
## 4    1.5890354    3.5707520 369.7303
## 5    1.6587907    1.8318928 369.4598
```

Now let's calculate our values for $f_n$ and the credible sets at the 95% level:

```r
lambda <- 1.6587907
gamma <- 1.8318928
rbf <- rbfdot(sigma = gamma) # gamma is the estimated lengthscale parameter
model <- gausspr(X, y, kernel = rbf, lambda = lambda) # lambda is the estimated noise parameter

y_pred <- predict(model, X)
se <- predict(model, X)

lci <- c()
uci <- c()
for(i in 1:length(X)){
  lci[i] <- qnorm(c(0.025), mean = y_pred[i], sd = se[i])
  uci[i] <- qnorm(c(0.975), mean = y_pred[i], sd = se[i])
}
```

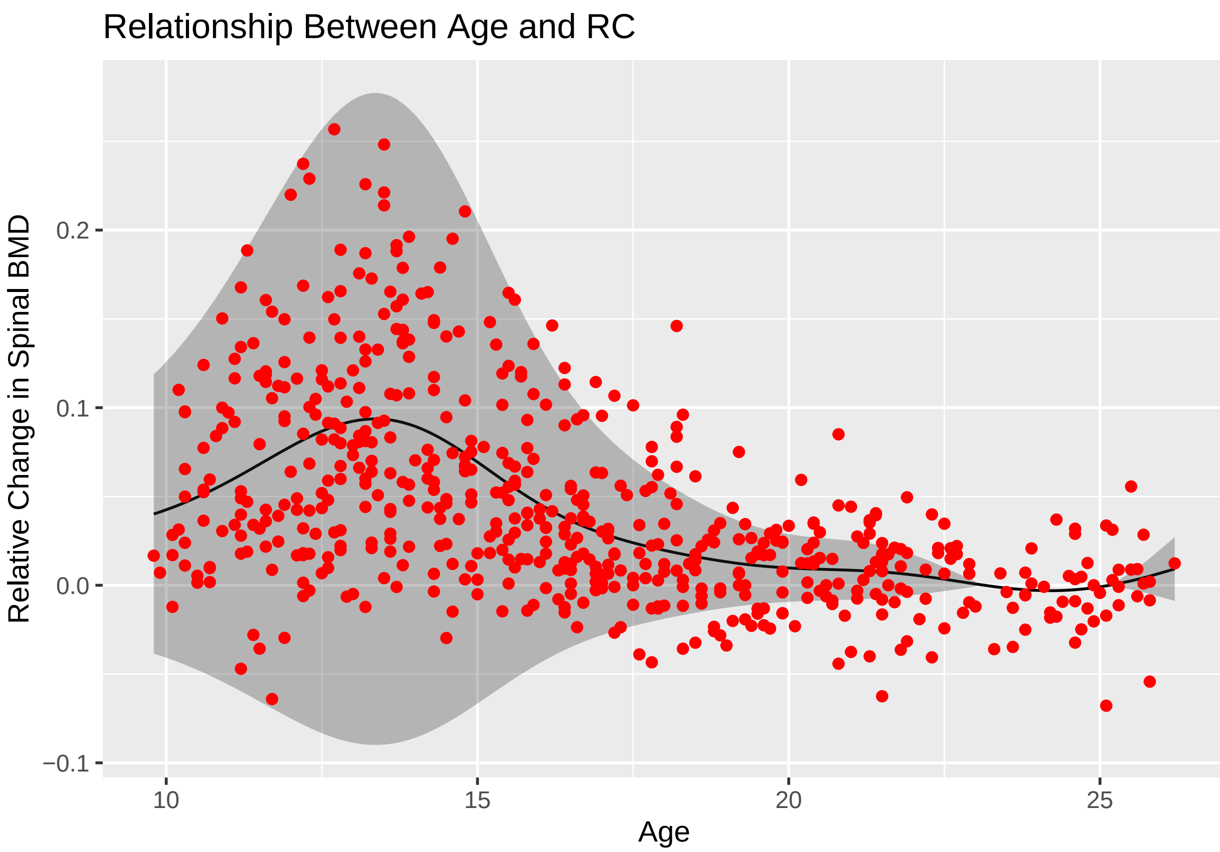
Let's now plot what we found above along with the data:

```r
library(ggplot2)
```

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

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

```r
df <- data.frame(x = X, y = y_pred, ci_low = lci, ci_high = uci)
ggplot(df, aes(x = x, y = y)) +
  geom_line() +
  geom_ribbon(aes(ymin = ci_low, ymax = ci_high), alpha = 0.3) +
  geom_point(data = data.frame(x = X, y = y), aes(x = x, y = y), color = "red") +
  labs(x = "Age", y = "Relative Change in Spinal BMD", title = "Relationship Between Age and RC")
```




Now let's repeat what we just did but now using a low rank approximation for 100 and 50 dimensions, we start by finding our hyperparameters using empirical bayes:

```r
# Function that calculates the approximate negative marginal log likelihood
approx_nmll <- function(par, d){
  # Obtain the indices that would sort v
  idx <- order(X)
  # Create a logical vector indicating repeated elements
  repeated <- duplicated(X)[idx]
  # Rearrange v so that repeated elements are pushed to the back
  X_rb <- c(X[idx][!repeated], X[idx][repeated])

  print(par)
```

```
  lambda <- par[1]
  gamma <- par[2]
  rbf <- rbfdot(sigma = gamma)
  gram_d <- kernelMatrix(rbf, X_rb[1:d])
  gram_nd <- kernelMatrix(rbf, X_rb, X_rb[1:d])
  sigma_d <- lambda* gram_d + t(gram_nd) %*% gram_nd

  term_1 <- 0.5* ( log(det(sigma_d + 1e-6 * diag(d))) -log(det(gram_d + 1e-6 * diag(d))) + (n-d)*log(la
  term_2 <- (1/ (2*lambda)) * ( norm(y, type="2") - norm( sqrt(solve( sigma_d + 1e-6 * diag(d) )) %*% t
  term_3 <- (n/2) * log(2*pi)
  term_1 + term_2 + term_3
}
# Set our search parameters
lower <- c(0.01, 0.01)
upper <- c(3, 4)
n_searches <- 5

# Conduct multiple searches with different initial parameters
set.seed(123)
results <- data.frame(matrix(nrow = n_searches, ncol = 3))
colnames(results) <- c("init_param_1", "init_param_2", "max_approx_nmll")
for (i in 1:n_searches) {
  init_params <- runif(2, lower, upper)
  # Use optim to minimize the negative marginal log likelihood
  result <- optim(init_params, approx_nmll, d=100)
  results[i,] <- c(init_params[1], init_params[2], -result$value)
}
print(results)
```

I couldn't get the above to work as I was getting numerical errors when trying to square root the inverse of sigma_d, I tried a wide range of different initial parameters to no avail. If I had got it working I would then have made a plot of $f_n$ with the credible sets similarly as I did before carrying out this low rank approximation.