Gaussian Process Regression

Kieran Morris

Task 1: Theory

We have a bunch of questions to prove so lets get on with it, firstly we must recall the definition of kernels:

A kernel is a map $k : X \times X \to \mathbb{R}$ such that for any finite set of points $x_1, x_2, \ldots, x_n \in X$, we have that $k(x_i, x_j) = k(x_j, x_i)$ and for any $a = (a_1, ..., a_n) \in \mathbb{R}^n$, we have that

$$\sum_{i,j} a_i a_j k(x_i, x_j) \geq 0$$

.

We attempt to show that the following functions are kernels:

1.1

For any m $in\mathbb{N}$, and the function $g : X \to \mathbb{R}$, we have that $k(x, y) = g(x)g(y)$ is a kernel. Proof:

Let $k(x, y) = g(x)g(y)$, then we have that $k(x, y) = k(y, x)$, and for any $a = (a_1, ..., a_n) \in \mathbb{R}^n$, we have that

$$\sum_{i,j} a_i a_j k(x_i, x_j) = \sum_{i,j} a_i a_j g(x_i) g(x_j) = \left(\sum_i a_i g(x_i) \right)^2 \geq 0$$

.

1.2

For any constant $a \geq 0$, the function $k(x, y) = a$ is a kernel. Proof:

Let $k(x, y) = a$ for all $x, y \in X$, then we have that $k(x, y) = k(y, x)$, and for any $a = (a_1, ..., a_n) \in \mathbb{R}^n$, we have that

$$\sum_{i,j} a_i a_j k(x_i, x_j) = \sum_{i,j} a_i a_j a = a^2 \sum_{i,j} a_i a_j = a^2 \left(\sum i, j a_i a_j \right)^2 \geq 0$$

.

1.3

For any $m \in \mathbb{N}$, kernels $\{k_p\}_{p=1}^m$ and non negative real numbers $\{c_p\}_{p=1}^m$, the function $k(x, y) = \sum_{p=1}^m c_p k_p(x, y)$ is a kernel.

Proof:

Since each k_p is a kernel, we have that $k_p(x, y) = k_p(y, x)$, and so $k(x,y) = k(y,x) too. Additionally for any $a = (a_1, ..., a_n) \in \mathbb{R}^n$, we have that

$$\sum_{i,j} a_i a_j k(x_i, x_j) = \sum_{i,j} a_i a_j \sum_{p=1}^m c_p k_p(x_i, x_j) = \sum_{p=1}^m c_p \sum_{i,j} a_i a_j k_p(x_i, x_j) \geq 0$$

.

1.4

For a kernel k on $\mathbb{R}^p$, and $X \subset \mathbb{R}^p$, the function $k'(x,y) = k(x,y)$ if $x, y \in X$, and $k'(x,y) = 0$ otherwise is a kernel.

Proof:

Trivially $k'(x,y) = k'(y,x)$, and for any $a = (a_1, ..., a_n) \in \mathbb{R}^n$, then we write (without loss of generality) $x_1, ..., x_l \in X$ and $x_{l+1}, ..., x_n \notin X$, then we have that

$$\sum_{i,j=1}^{m} a_i a_j k'(x_i, x_j) = \sum_{i,j=1}^{l} a_i a_j k(x_i, x_j) + \sum_{i,j=l+1}^{m} a_i a_j k(x_i, x_j) = \sum_{i,j=1}^{l} a_i a_j k(x_i, x_j) + 0 \geq 0$$

.

1.5

Task 2: Gaussian Process Regression

Like with the assessed coursework we will use the bone mineral density dataset: `spnbmd.csv` which we have already fitting LASSO and polynomial regression algorithms on.

```
data <- read.csv("spnbmd.csv")
head(data)
```

```
##   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
```

We choose to use a gaussian process regrssion model with known variance $\sigma^2 = \lambda$, and a gaussian kernel $k(x,y) = \exp\left(-\frac{1}{2\psi}\|x-y\|^2\right)$. We begin by computing the posterior distribution, which we do by maximising the marginal likelihood with respect to the hyperparameters λ and ψ - more specifically the negative marginal so we can use `optim` to find the maximum.

```
library(kernlab)

x = as.vector(data$age)
y = as.vector(data$spnbmd)
n = length(x)

neg_marginal <- function(params){
  lambda <- params[1]
  psi <- params[2]
  #Compute K_n and K+lambdaI
  K <- kernelMatrix(rbfdot(sigma = psi), x)
  L <- K + lambda*diag(n)
  if (det(L) == 0){
    return(Inf)
```

```
  }
  #Compute alpha
  y <- as.matrix(y)
  alpha = solve(L,y)
  #Compute neg log marginal likelihood
  neg_marginal_val <- 0.5*(t(y)%*%alpha + sum(log(diag(L))))

  return(neg_marginal_val)
}
```

Like with last week we will use `age` as predictor and `spnbmd` as the response. Since this is not a convex optimisation problem we will simulate a grid of values for λ and ψ to use an inital guesses for the `optim` function.

```
#Simulate grid
lambdas <- seq(1, 100, length.out = 10)
psis <- seq(1, 100, length.out = 10)
neg_marginals <- matrix(0, length(lambdas), length(psis))
```

```
# This takes a while, so we omit running this for knitting purposes and state the output (trust me plea

#Perform optim with inital guesses
for(i in 1:length(lambdas)){
  for(j in 1:length(psis)){

    cat("lambda:",lambdas[i],"\n")
    cat("psi:",psis[j],"\n")
    para = c(lambdas[i], psis[j])
    neg_marginals[i,j] <- optim(para, neg_marginal,control = list(maxit = 10))$value
  }
}

#Find the minimum
min_index <- which(neg_marginals == min(neg_marginals), arr.ind = TRUE)

lambda <- lambdas[min_index[1]]
psi <- psis[min_index[2]]

print(cat("The optimal lambda is", lambda, "and the optimal psi is", psi))
```

We find that optimal values are $(\lambda, \psi) = (1, 78)$, and so we can now compute the posterior mean and variance of the GP. We will use the `kernlab` package to do this.

```
lambda = 1
psi = 78
K <- kernelMatrix(rbfdot(sigma = psi), x)
L <- K + lambda*diag(n)
L_inv <- solve(L)
y <- as.matrix(y)
alpha = solve(L,y)

#We now create the mean funcion and variance function
```

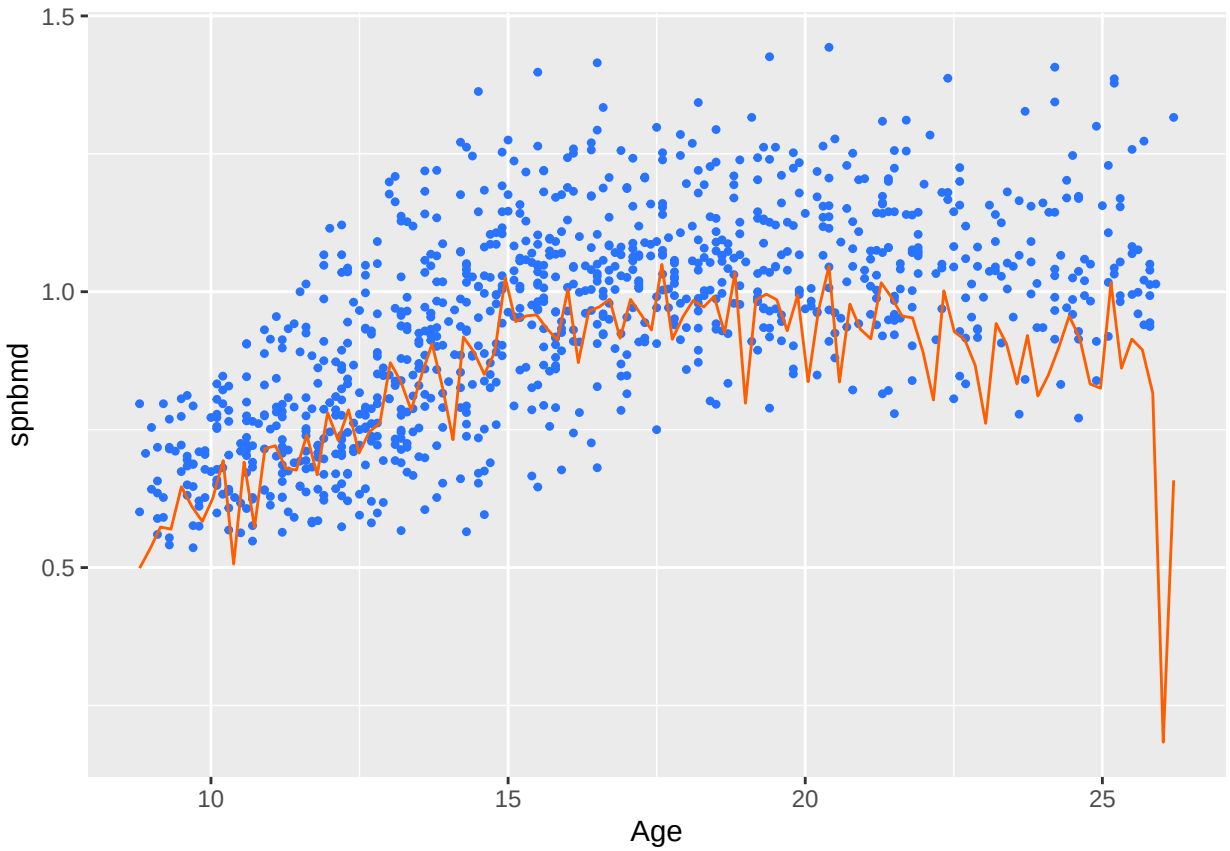
```r
mean_func <- function(x_data){
  k_n <- kernelMatrix(rbfdot(sigma = psi), x, x_data)
  output = t(k_n)%*%alpha
  return(output)
}

variance_func <- function(x_data,x2_data){
  k_n <- as.vector(kernelMatrix(rbfdot(sigma = psi), x2_data, x_data))
  k_star <- as.vector(kernelMatrix(rbfdot(sigma = psi), x_data, x2_data))
  k_1 <- kernelMatrix(rbfdot(sigma = psi), x, x_data)
  k_2 <- kernelMatrix(rbfdot(sigma = psi), x, x2_data)
  print(dim(k_1))
  print(dim(L_inv))
  output <- k_star - t(k_1)%*%solve(L)%*%k_2
  return(output)
}
```

We now plot the posterior mean function over our data with ggplot.

```r
library(ggplot2)
```

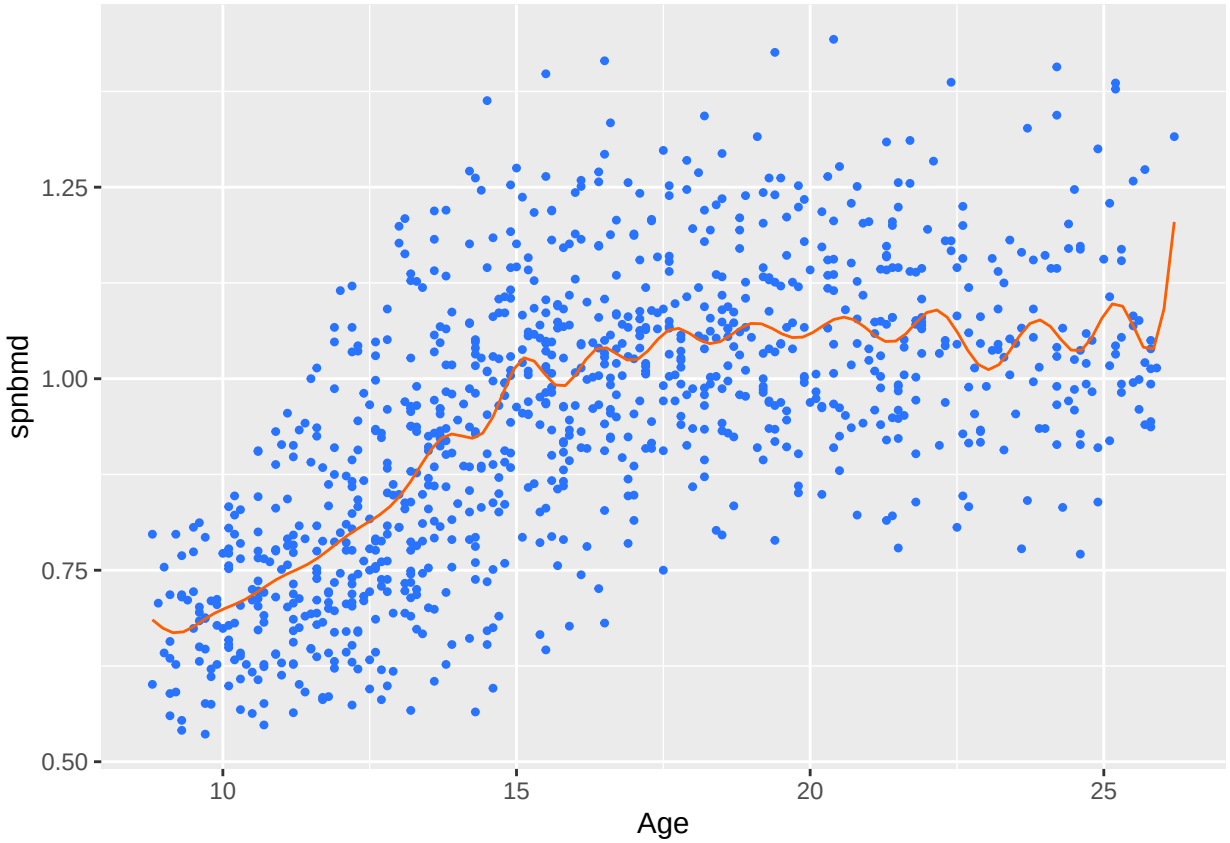
```
##
## Attaching package: 'ggplot2'

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

```r
# Generate estimates
x_values <- seq(min(x), max(x), length.out = 100)
y_estimates <- mean_func(x_values)

# Create a dataframe for the estimates
estimate_data <- data.frame(x = x_values, y = y_estimates)

# Plot the original data and the estimates
plt <- ggplot() +
  geom_point(data = data, aes(x = age, y = spnbmd), col = "#2873ff", pch = 20) +
  geom_line(data = estimate_data, aes(x = x, y = y), col = "#ff5e00") +
  xlab("Age") +
  ylab("spnbmd")

plt
```



We can see that our function does follow the data, but it is is very wiggly thanks to the high hyperparameters ψ. We see below that if we arbitraily choose $\lambda = 1$ and $\psi = 0.78$ then the function is much smoother.

```
lambda = 0.01
psi = 0.78
K <- kernelMatrix(rbfdot(sigma = psi), x)
L <- K + lambda*diag(n)
y <- as.matrix(y)
alpha = solve(L,y)


# Generate estimates
x_values <- seq(min(x), max(x), length.out = 100)
y_estimates <- mean_func(x_values)

# Create a dataframe for the estimates
estimate_data <- data.frame(x = x_values, y = y_estimates)

# Plot the original data and the estimates
plt <- ggplot() +
  geom_point(data = data, aes(x = age, y = spnbmd), col = "#2873ff", pch = 20) +
  geom_line(data = estimate_data, aes(x = x, y = y), col = "#ff5e00") +
  xlab("Age") +
  ylab("spnbmd")

plt
```

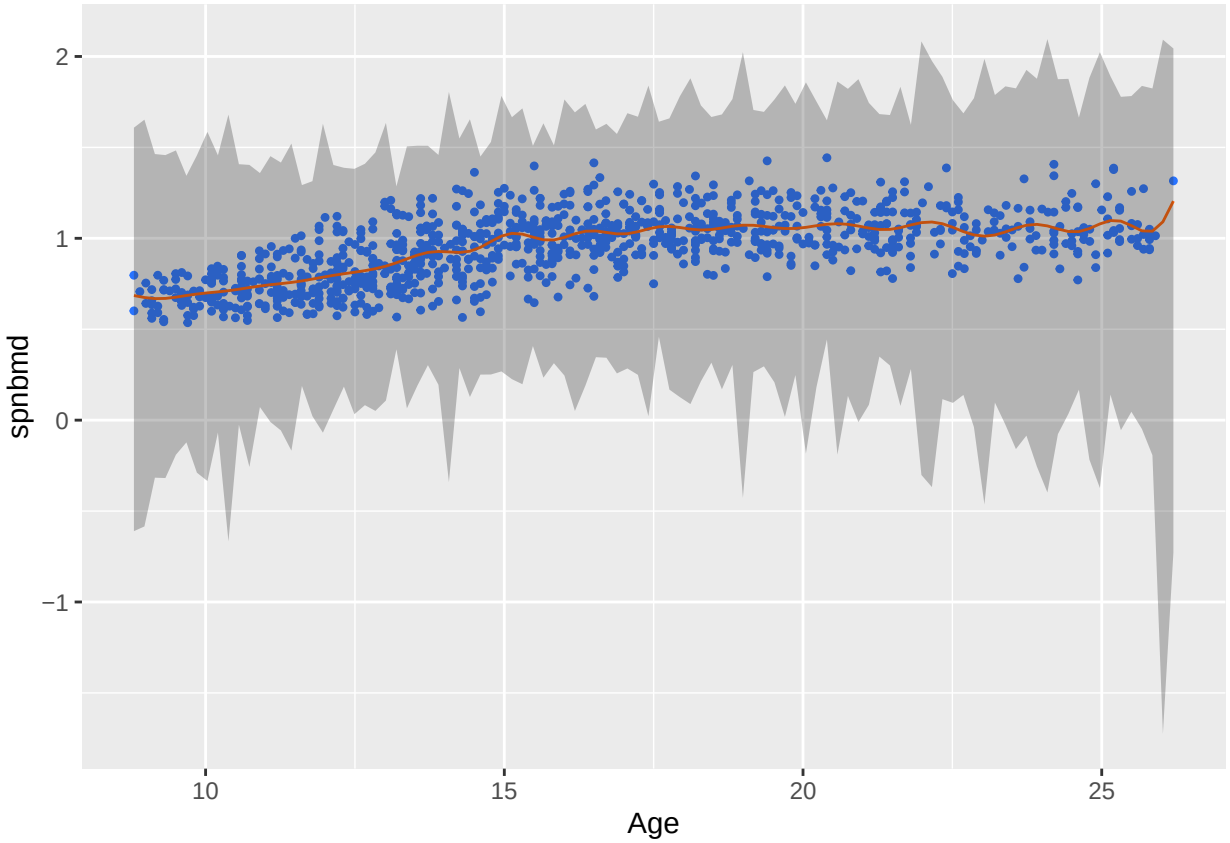


We can now find the credible interval function for the posterior mean function by using the variance function we defined earlier.

```
lambda = 1
psi = 78
K <- kernelMatrix(rbfdot(sigma = psi), x)
L <- K + lambda*diag(n)
y <- as.matrix(y)
alpha = solve(L,y)

#Function for predible interval
credible_int <- function(x_data,alpha_param = 0.05){
  mean <- mean_func(x_data)
  k_xx <- diag(variance_func(x_data, x_data))

  ci_upper <- mean + qnorm(1 - alpha_param/2)*sqrt(k_xx)
  ci_lower <- mean - qnorm(1 - alpha_param/2)*sqrt(k_xx)

  return(list(upper = ci_upper, lower = ci_lower))
}

#Plotting the credible interval with the mean function

x_values <- seq(min(x), max(x), length.out = 100)
ci <- credible_int(x_values)
```

```
## [1] 1003  100
## [1] 1003 1003
```

```r
# Create a dataframe for the estimates
estimate_data <- data.frame(x = x_values, y = y_estimates)

# Plot the original data and the estimates
plt <- ggplot() +
  geom_point(data = data, aes(x = age, y = spnbmd), col = "#2873ff", pch = 20) +
  geom_line(data = estimate_data, aes(x = x, y = y), col = "#ff5e00") +
  geom_ribbon(data = estimate_data, aes(x = x, ymin = ci$lower, ymax = ci$upper), alpha = 0.3) +
  xlab("Age") +
  ylab("spnbmd")

plt
```



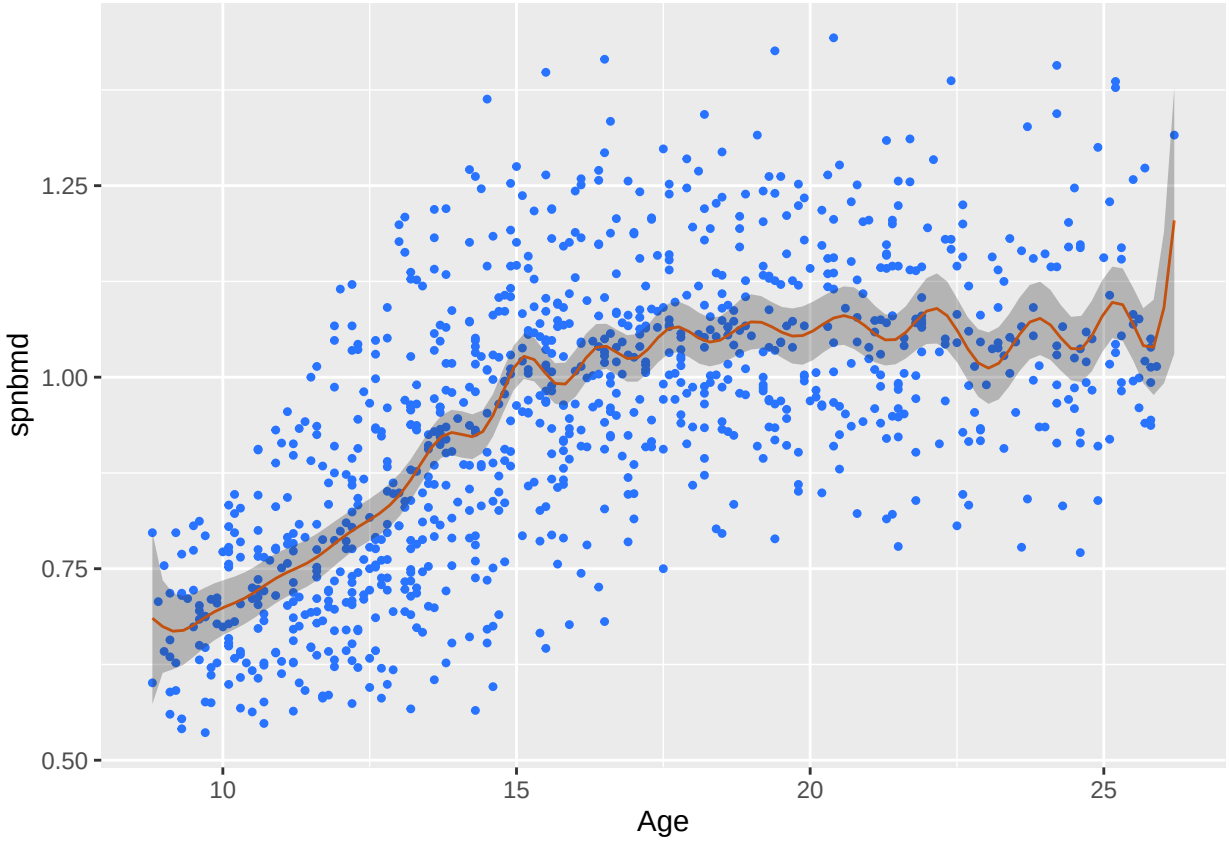
As we can see we get a fairly large confidence interval, which is expected alongside a wiggly line like this, finally lets try it with the smoother function.

```r
lambda = 0.01
psi = 0.78
K <- kernelMatrix(rbfdot(sigma = psi), x)
L <- K + lambda*diag(n)
y <- as.matrix(y)
alpha = solve(L,y)
```

```r
# Generate estimates
x_values <- seq(min(x), max(x), length.out = 100)
y_estimates <- mean_func(x_values)

# Create a dataframe for the estimates

ci <- credible_int(x_values)
```

```
## [1] 1003  100
## [1] 1003 1003
```

```r
# Plot the original data and the estimates

plt <- ggplot() +
  geom_point(data = data, aes(x = age, y = spnbmd), col = "#2873ff", pch = 20) +
  geom_line(data = estimate_data, aes(x = x, y = y), col = "#ff5e00") +
  geom_ribbon(data = estimate_data, aes(x = x, ymin = ci$lower, ymax = ci$upper), alpha = 0.3) +
  xlab("Age") +
  ylab("spnbmd")

plt
```



This is a much nicer interval!