SM2 Assessed Coursework

Rachel Wood

2023-03-14

This portfolio considers a model for n observations $\{(y_i^0, x_i^0)\} \in \mathbb{R} \times \mathbb{R}^p$ defined by

$$Y_i^0 \sim f(y; \mu_i, \phi)dy, \qquad g(\mu_i) = \alpha + f(x_i^0), \qquad \text{for} i = 1, \ldots, n \tag{1}$$

where $\alpha \in \mathbb{R},\ \phi \in (0, \infty)$ and $f \in \mathcal{F} = \mathcal{H}_k$.

Here $(\mathcal{H}_{\|}, \langle \cdot, \cdot \rangle)$ is a reproducible kernel Hilbert space (RKHS) with positive semi definite kernel k. Then we have that $\mathcal{H}_k$ satisfies the reproducibility property

$$f(x) = \langle f, k(x, \cdot) \rangle \qquad \forall x \in \mathcal{X}, \forall f \in \mathcal{H}_k \tag{2}$$

Part 1

Question 1

Here we consider the identifiability of (1). A kernel will lead to identifiable models if and only if the corresponding RKHS does not contain constant functions.

Unidentifiable kernel: We can take the kernel to be

$$k(x, y) = 1$$

which is positive semi-definite, we can see that taking $\mathcal{H}_k = \{f : f(x) = c\ \forall x\}$ and $\langle f, g \rangle_k = fg$ satisfies the reproducing property:

$$f(x) = \langle f,\ k(x, \cdot) \rangle = \langle f, 1 \rangle = f \qquad \forall x \in \mathcal{X},\ \forall f \in \mathcal{H}_k$$

and thus $(\mathcal{H}_k, \langle \cdot, \cdot \rangle_k)$ is the corresponding RKHS. Since $\mathcal{H}_k$ is made up of constant functions, k does not produce identifiable models.

Identifiable Kernel: An example of an identifiable kernel is the Gaussian kernel:

$$k_\lambda(x, x') = \exp\left(-\frac{||x - x'||^2}{\lambda}\right)$$

Any function in the corresponding RKHS $f \in \mathcal{H}_k$ must satisfy:

$$f(x) = \langle f, k(x, \cdot) \rangle = \frac{1}{\lambda} \left\langle f, -\exp ||x - \cdot||^2 \right\rangle$$

It is clear that there is no constant function f (excepting the zero function) which satisfies this. Hence the Gaussian kernel leads to identifiable models.

Question 2

For this question, we consider the solution $\hat{f}_\lambda$ to the optimisation problem

$$(\hat{\alpha_\lambda}, \hat{\phi_\lambda}, \hat{f_\lambda}) \in \underset{\alpha\in\mathbb{R},\phi\in(0,\infty),f\in\mathcal{H}_k}{\operatorname{argmax}} \frac{1}{2n} \sum_{i=1}^{n} \log f\left(y_i; g^{-1}\left(\alpha + f(x_i^0)\right), \phi\right) - \lambda||f||^2_{\mathcal{H}_k} \tag{3}$$

We assume $\mathcal{H}_k = \tilde{\mathcal{H}_n} \oplus \tilde{\mathcal{H}_n}^\perp$, hence we can write $\hat{f}_\lambda = f_1 + f_2$ where $f_1 \in \tilde{\mathcal{H}_n}$ and $f_2 \in \tilde{\mathcal{H}_n}^\perp$. Thus there exists coefficients $\hat{\beta}_\lambda = \left(\hat{\beta}_{\lambda,1}, \ldots, \hat{\beta}_{\lambda,n}\right) \in \mathbb{R}^n$ such that

$$f_1 = \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, \cdot)$$

and we can write $f = \sum_{i=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, \cdot) + f_2$. Further, by the reproducing property, for any x_j:

$$f(x_j) = \left\langle \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, \cdot) + f_2,\ k(x_j^0, \cdot) \right\rangle = \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} \left\langle k(x_i^0, \cdot), k(x_j^0, \cdot)\right\rangle = \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, x_j^0)$$

and so $f(x_j)$ does not depend on f_2 and as a consequence, the first term in (3) also does not depend on f_2. Hence to choose f_2 we only need to consider minimizing the regularisation term. Using that $\langle f_1, f_2\rangle = 0$, we see

$$||f||^2_{\mathcal{H}_k} = \langle f_1 + f_2, f_1 + f_2\rangle = ||f_1||^2_{\mathcal{H}_k} + ||f_2||^2_{\mathcal{H}_k} \geq ||f_1||^2_{\mathcal{H}_k}$$

with the last inequality becoming an equality when $f_2 = 0$. Hence the minimiser $\hat{f}_\lambda$ must have $f_2 = 0$ and can be written as

$$\hat{f}_\lambda = \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, \cdot) \tag{4}$$

Question 3

We now want to substitute the results of (4) into (3). The regularisation term becomes:

$$\begin{aligned}
||f||^2_{\mathcal{H}_k} &= \left\langle \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, \cdot), \sum_{i=j}^{n} \hat{\beta}_{\lambda,j} k(x_j^0, \cdot) \right\rangle \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{\beta}_{\lambda,i} \left\langle k(x_i^0, \cdot), k(x_j^0, \cdot)\right\rangle \hat{\beta}_{\lambda,j} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, x_j^0) \hat{\beta}_{\lambda,j} \\
&= \hat{\beta}_\lambda^T K \hat{\beta}_\lambda
\end{aligned}$$

and so (3) becomes

$$(\hat{\alpha_\lambda}, \hat{\phi_\lambda}, \hat{\beta_\lambda}) \in \underset{\alpha\in\mathbb{R},\phi\in(0,\infty),\beta\in\mathbb{R}^n}{\operatorname{argmax}} \frac{1}{2n} \sum_{i=1}^{n} \log f\left(y_i; g^{-1}\left(\alpha + \sum_{j=1}^{n} \beta_i k(x_i^0, x_j^0)\right), \phi\right) - \lambda \beta^T K \beta \tag{5}$$

Question 4

Given $m \leq n+2$, we want to obtain an m-dimensional problem. Then $d = m-2$ is the length of the new $\tilde{\beta}_\lambda$ vector to estimate.

The Nyström method approximates k by $\tilde{f}^{(m)} = k^{(m)}$, given by

$$\tilde{k}^{(d)}(x, x') = k_d(x)^T K_{d,d}^{-1}\ k_d(x')$$

where $K_{d,d} \in \mathbb{R}^{d \times d}$ is the first d rows and columns of the gram matrix K and $k_d(x) = (k(x_1^0, x), \ldots, k(x_d^0, x))$

Then we can write $f(x)$ as

$$f = \sum_{i=1}^{n} \beta_i\ \tilde{k}_d(x_i^0, \cdot) = \sum_{i=1}^{n} \beta_i\ k_d(x_i^0)^T K_{d,d}^{-1}\ k_d(\cdot) = \left(\sum_{i=1}^{n} \beta_i\ k_d(x_i^0)^T K_{d,d}^{-1} \right) k_d(\cdot)$$

and take the new vector of coefficients to be:

$$\tilde{\beta}^T = \sum_{i=1}^{n} \beta_i\ k_d(x_i^0)^T \left(K_d^0\right)^{-1} = \beta^T K_{n,d} K_{d,d}^{-1}$$

where $K_{n,d} \in \mathbb{R}^{n \times d}$ is the first d columns of K We can now rewrite f as:

$$f = \tilde{\beta}^T k_d(\cdot) = \sum_{j=1}^{d} \tilde{\beta}_j k(x_j, \cdot)$$

We now consider the regularisation term, again we substitute $\tilde{k}^{(d)}$:

$$\beta^T \tilde{K}^{(d)} \beta = \beta^T K_{n,d}\ K_{d,d}^{-1}\ K_{n,d}^T\ \beta = \left(\beta^T K_{n,d}\ K_{d,d}^{-1}\right) K_{d,d}^T \left(K_{d,d}^{-T}\ K_{n,d}\ \beta\right) = \tilde{\beta}^T K_{d,d}^T\ \tilde{\beta}$$

Hence we can now write our m-dimension optimisation problem:

$$(\hat{\alpha_\lambda}, \hat{\phi_\lambda}, \tilde{\beta}_\lambda) \in \underset{\alpha \in \mathbb{R}, \phi \in (0,\infty), \tilde{\beta} \in \mathbb{R}^d}{\operatorname{argmax}} \frac{1}{2n} \sum_{i=1}^{n} \log f\left(y_i; g^{-1}\left(\alpha + \sum_{j=1}^{d} \tilde{\beta}_j k(x_j, x_i) \right), \phi \right) - \lambda \tilde{\beta}^T K_{d,d}^T\ \tilde{\beta} \tag{6}$$

Question 5

For this question we first need to obtain the penalty term as $\omega^T \omega$ for some vector ω. We can first obtain the eigen-decomposition of $K_{d,d}^T$:

$$K_{d,d}^T = V \Lambda V^T$$

Then writing $\Lambda = \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} = \Lambda^{\frac{1}{2}} \left(\Lambda^{\frac{1}{2}}\right)^T$, we get

$$\tilde{\beta}^T K_{d,d}\ \tilde{\beta} = \tilde{\beta}^T V \Lambda^{\frac{1}{2}} \left(\Lambda^{\frac{1}{2}}\right)^T V^T \tilde{\beta} = \omega^T \omega$$

where $\omega^T = \tilde{\beta}^T V \Lambda^{\frac{1}{2}}$, or equivalently $\tilde{\beta}^T = \omega^T \Lambda^{-\frac{1}{2}} V^T$.

Then our objective function becomes

$$(\hat{\alpha_\lambda}, \hat{\phi_\lambda}, \hat{\omega}_\lambda) \in \underset{\alpha\in\mathbb{R},\phi\in(0,\infty),\omega\in\mathbb{R}^d}{\operatorname{argmax}} \frac{1}{2n}\sum_{i=1}^{n} \log f\left(y_i; g^{-1}\left(\alpha + \omega^T \Lambda^{-\frac{1}{2}} V^T K_{d,n}\right), \phi\right) - \lambda\omega^T\omega \tag{7}$$

so we can use $X' = \Lambda^{-\frac{1}{2}} V^T K_{n,d}$, where K is the Gram matrix of the original data, as the input for `glmnet()` and this will give the estimated ω.

Part 2

This section implements the theory from Part 1 on the `wesdr` data from the `gss` package, which has the binary response variable `ret` and explanatory variables `dur`, `gly` and `bmi`

```
library(dplyr)
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
library(gss)
data("wesdr")
head(wesdr)
```

```
##    dur  gly  bmi ret
## 1 10.3 13.7 23.8   0
## 2  9.9 13.5 23.5   0
## 3 15.6 13.8 24.8   0
## 4 26.0 13.0 21.6   1
## 5 13.8 11.1 24.6   1
## 6 31.1 11.3 24.6   1
```

We also separate the explanatory and response variables and construct a testing and training set:

```
X <- as.matrix(wesdr[,1:3])
y <- as.vector(wesdr[,4])

train <- sample(1:nrow(X), 500)
X_train <- X[train,]
y_train <- y[train]

X_test <- X[-train,]
y_test <- y[-train]
```

Question 6

We can now use `glmnet` to write a function implementing Question 4 and 5:

```
library(kernlab)
library(glmnet)
estimate <- function(X, y, lambda, c, m){
  d <- m - 1
  K <- kernelMatrix(kernel = rbfdot(c), x = X)
  K_d <- K[1:d, 1:d]

  K_eigen <- eigen(K_d)
  V <- K_eigen$vectors
  Lambda <- (K_eigen$values)

  K_nd <- K[1:d,]
  X_new <- diag(1/sqrt(Lambda)) %*% t(V) %*% K_nd


  fit <- glmnet(t(X_new), y, family = "binomial", alpha = 0, lambda = lambda)

  alpha <- fit$a0
  omega <- as.vector(fit$beta)

  beta <- omega %*% diag(1/sqrt(Lambda)) %*% t(V)

  values <- list("alpha" = alpha, "beta" = beta, "fit" = fit)
  return(values)
}
```

Question 7

For certain values of m, c and $lambda$ the `X_new` matrix contains some `NaN` values, for example we try the following values:

```
c <- 1e-04
m <- 60
lambda <- 0.1
epsilon = 0.001

estimate <- estimate(X, y, lambda, c, m)
```

```
## Warning in sqrt(Lambda): NaNs produced
```

```
## Error in glmnet(t(X_new), y, family = "binomial", alpha = 0, lambda = lambda): x has missing values;
```

This is because some of the eigenvalues of K are small enough that they are represented as `0` in a floating point number. To remedy this we can replace the kernel matrix with the altered $K'_d = K_d + epsilonI$ for a small ϵ

```
estimate_new <- function(X, y, lambda, c, m, epsilon = 0.001){
  d <- m - 1
  K <- kernelMatrix(kernel = rbfdot(c), x = X)
  K_d <- K[1:d, 1:d] + epsilon* diag(rep(1,d))
```

```
  K_eigen <- eigen(K_d)
  V <- K_eigen$vectors
  Lambda <- (K_eigen$values)

  K_nd <- K[1:d,]
  X_new <- diag(1/sqrt(Lambda)) %*% t(V) %*% K_nd


  fit <- glmnet(t(X_new), y, family = "binomial", alpha = 0, lambda = lambda)

  alpha <- fit$a0
  omega <- as.vector(fit$beta)

  beta <- omega %*% diag(1/sqrt(Lambda)) %*% t(V)

  values <- list("alpha" = alpha, "beta" = beta, "fit" = fit)
  return(values)
}
```

and we can see now that this does not produce any errors

```
estimate <- estimate_new(X_train, y_train, lambda, c, m)
```

Question 8

Here we want to choose λ using cross-validation, the `glmnet` package enables us to do this using the `cv.glmnet()` function, which uses the deviance to perform k-fold cross-validation. We use the following function to obtain the solution to (6) for the cross-validated value of λ:

```
lambda_cv_estimate <- function(X, y, c, m,  epsilon = 0.01){
  d <- m - 1
  K <- kernelMatrix(kernel = rbfdot(c), x = X)
  K_d <- K[1:d, 1:d]

  K_eigen <- eigen(K_d)

  V <- K_eigen$vectors
  Lambda <- (K_eigen$values) + epsilon * (rep(1,d))

  K_nd <- K[1:d,]
  X_new <- diag(1/sqrt(Lambda)) %*% t(V) %*% K_nd

  cv_fit <- cv.glmnet(t(X_new), y, family = "binomial", alpha = 0)

  lambda <- cv_fit$lambda.min
  lambda_ind <- cv_fit$index[1]

  fits <- cv_fit$glmnet.fit

  alpha <- fits$a0[lambda_ind]
  omega <- as.vector(fits$beta[,lambda_ind])
```

```
  beta <- omega %*% diag(1/sqrt(Lambda)) %*% t(V)

  values <- list("alpha" = alpha, "beta" = beta, "lambda" = lambda, "cvm" = cv_fit$cvm[lambda_ind], "fi
  return(values)
}
```

```
fit <- lambda_cv_estimate(X, y, 1, 100)
```

Question 9

We can now use the function from **Question 8** to write a function which performs a grid search to choose the optimal value for `c`:

```
c_cv_estimate <- function(X, y, m){
  c = c(0.0001, 0.0005, 0.0008, 0.001, 0.003, 0.005, 0.01, 0.05, seq(0.1, 1, 0.1), 1.5,2, 2.5)

  l <- length(c)

  alphas <- numeric(l)
  betas <- matrix(nrow = l, ncol = m - 1)

  lambdas <- numeric(l)
  cvms <- numeric(l)

  fits <- vector(mode = "list", length = l)

  Ks <- vector(mode = "list", length = l)

  for (i in 1:l) {
    est <- lambda_cv_estimate(X, y, c[i], m)

    alphas[i] <- est$alpha
    betas[i,] <- est$beta

    lambdas[i] <- est$lambda
    cvms[i] <- est$cvm

    fits[[i]] <- est$fit

    Ks[[i]] <- est$K
  }

  min_ind <- which.min(cvms)

  values <- list("alpha" = alphas[min_ind], "beta" = betas[min_ind, ], "lambda" = lambdas[min_ind], "c"
  return(values)
}
```

We can now use the training dataset to obtain the solution when `c` and `lambda` are chosen with cross-validation:

```
parameters <- c_cv_estimate(X_train, y_train, m =150)
```

Question 10

We can now use the output of the function above, the training set and test set to produce the estimated response vector for the test set: for various a range of `m` values

```
set.seed(1)
m <- c(10,20,30,50,70,100,150,200,300)
n <- length(m)

responses <- matrix(nrow = length(y_test), ncol = n)
for (i in 1:n){
  model <-  c_cv_estimate(X_train, y_train, m[i])

  fit <- model$fit

  c <- model$c

  K <- model$K
  d <- m[i] -1
  K_d <- K[1:d, 1:d]

  K_eigen <- eigen(K_d)

  V <- K_eigen$vectors
  Lambda <- (K_eigen$values) + epsilon * rep(1,d)

  K_prime <- kernelMatrix(kernel = rbfdot(c), x = X_test, y = X_train)
  K_prime_ld <- K_prime[,1:d]


  X_test_new <-  diag(1/sqrt(Lambda)) %*% t(V) %*% t(K_prime_ld)

  omega <- model$beta %*% V %*% diag(sqrt(Lambda))
  eta <- model$alpha + t(X_test_new) %*% t(omega)

  pred_response <-round(1 / (1+ exp(-eta)))

  responses[,i] <- pred_response
}
```

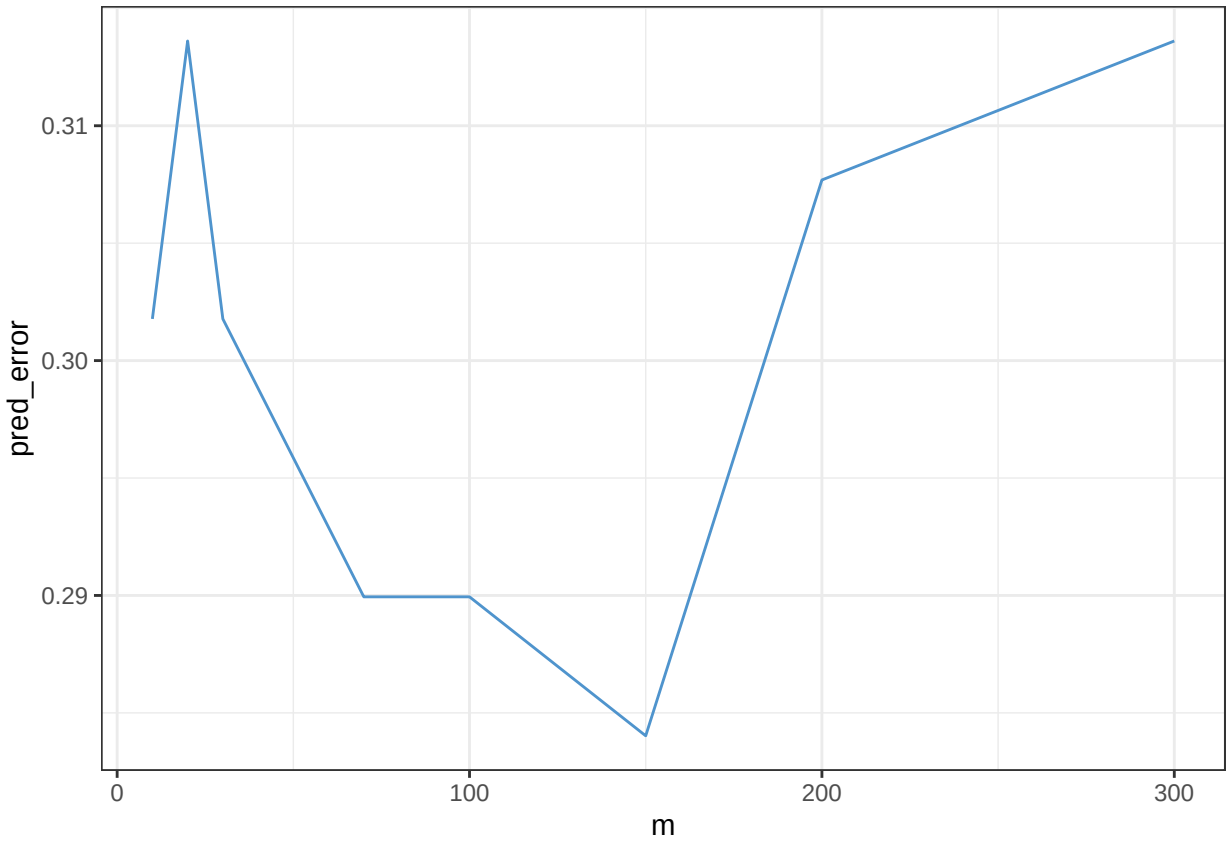
We can now plot the predicted errors for different values of `m`:

```
library(dplyr)
pred_error <- colMeans(abs(responses - y_test))

plot_dat <- cbind("m" = m, "Prediction Error" = pred_error) %>%
  as_tibble()
```

```
library(ggplot2)

ggplot(plot_dat) +
  geom_line(aes(x = m, y = pred_error), color = "steelblue3")
```



We can see the lowest prediction error is at `m = 150`, with a value of ≈ 0.28, hence we can get a prediction that performs better than a guess but is still not excellent.

Question 11

We can use a GAM to approximate the model given in (1):

```
library(mgcv)

fit_gam <- gam(ret ~ s(dur, bs = "cr") + s(gly, bs = "cr") + s(bmi, bs = "cr"), data = wesdr[train,], fa

test_pred <- predict(fit_gam, wesdr[-train,], type = "response")

gam_error <- mean(abs(round(test_pred) - y_test))
gam_error
```

```
## [1] 0.295858
```

We can see here the prediction error is very similar and there doesn't seem to be much different within the models