Statistical Methods 2 Homework

Sam Bowyer

2023-03-17

Task 1

Question 1

Unidentifiable model First we shall show that the constant kernel $k(x, x') = 1 \; \forall x \in \mathcal{X}$ leads to the model being unidentifiable. Importantly, note that the constant kernel is a valid kernel function, as it is positive definite and symmetric.

Now consider the vector space of functions $\mathcal{H}_k = \{f : f \text{ is a constant function}\}$ and the inner product $\langle \cdot, \cdot \rangle_k$ on this space given by $\langle f, g \rangle_k = fg$. Observe that with this inner product, we obtain the reproducing property

$$f(x) = \langle f, k(x, \cdot) \rangle_k \;\; \forall x \in \mathcal{X}, \; \forall f \in \mathcal{H}_k.$$

Hence $(\mathcal{H}_k, \langle \cdot, \cdot \rangle_k)$ is the unique reproducing kernel Hilbert space induced by the kernel k.

Then by the definition of our space $\mathcal{H}_k$ we know that there exist two functions $f_1, f_2 \in \mathcal{H}_k$ such that $a = f_1 \neq f_2 = b$ for some $a, b \in \mathcal{R}$. Finally note that the model parameterised by (f_1, α, ϕ) is the same as the model parameterised by $(f_2, \alpha + a - b, \phi)$ since

$$\alpha + f_1(x) = \alpha + a - b + b = (\alpha + a - b) + f_2(x) = a \;\; \forall x \in \mathcal{X}.$$

Hence the kernel k is leads to the model being unidentifiable.

Identifiable model We shall now show that the Gaussian kernel $k(x, x') = \exp(\sigma \|x - x'\|^2)$ leads to the model being identifiable. To this end, let us write the unique RKHS that this kernel induces as $(\mathcal{H}_k, \langle \cdot, \cdot \rangle_k)$.

Since the parameter ϕ of the model is uniquely defined for given f and α, we know that the model will only be unidentifiable if there exist two distinct functions $f_1, f_2 \in \mathcal{H}_k$ such that $f_1 - f_2$ is a constant function. (For example, similarly to the previous question, if $f_1(x) - f_2(x) = c \in \mathbb{R}$ for all $x \in \mathcal{X}$, then the model parameterised by (f_1, α, ϕ) is the same as the model parameterised by $(f_2, \alpha + c, \phi)$.)

However, we know that since $\mathcal{H}_k$ is a vector space and $f_1, f_2 \in \mathcal{H}$, then $f_1 - f_2 \in \mathcal{H}_k$. Furthermore, a result given in the lecture notes (pg. 200) tells us that $\mathcal{H}_k$ contains no constant functions except for the zero function (and since f_1 and f_2 are distinct, we know that $f_1 - f_2$ is not the zero function). Hence no such functions $f_1, f_2 \in \mathcal{H}_k$ exist meaning the model is identifiable.

Question 2

Set $\lambda > 0$ and suppose that the function

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

is indeed such that

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

As given in the question, we may assume that $\exists f_1 \in \tilde{\mathcal{H}}_n, f_2 \in \tilde{\mathcal{H}}_n^{\perp}$ such that $\hat{f}_\lambda = f_1 + f_2$. Now since $f_1 \in \tilde{\mathcal{H}}_n$ and

$$\tilde{\mathcal{H}}_n = \text{span}\{k(x_1^0, \cdot), ..., k(x_n^0, \cdot)\},$$

then there exists $\hat{\beta}_\lambda \in \mathbb{R}^n$ such that $f_1 = \sum_{i=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, \cdot)$.

Next observe that via the reproducing property of the RKHS $(\mathcal{H}_k, \langle \cdot, \cdot \rangle_k)$, we have that, for each $j \in \{1, ..., n\}$,

$$\begin{aligned}
\hat{f}_\lambda(x_j^0) = \langle \hat{f}_\lambda, k(x_j^0, \cdot) \rangle_k &= \langle f_1 + f_2, k(x_j^0, \cdot) \rangle_k \\
&= \langle f_1, k(x_j^0, \cdot) \rangle_k + \langle f_2, k(x_j^0, \cdot) \rangle_k \\
&= f_1(k(x_j^0) + 0 \\
&= \sum_{i=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, x_j^0)
\end{aligned}$$

Where we have also used the linearity of the inner product and the fact that

$$x_j^0 \in \tilde{\mathcal{H}}_n \implies x_j^0 \notin \tilde{\mathcal{H}}_n^{\perp} \implies \langle f_2, k(x_j^0, \cdot) \rangle_k = 0.$$

Now we have shown that $\hat{f}_\lambda(x_j^0) = \sum_{i=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, x_j^0)$ for all $j \in \{1, ..., n\}$, however, we must still prove that $\hat{f}_\lambda(x) = \sum_{i=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, x)$ for all $x \in \mathbb{R}^p$.

To do this, note that the first term of the optimisation problem's objective function

$$\frac{1}{2n} \sum_{i=1}^n \log \tilde{f}(y_i; g^{-1}(\alpha + f(x_i^0)), \phi)$$

only requires us to evaluate $f = \hat{f}_\lambda$ at $x_1^0, ..., x_n^0$, which is independent of f_2 if we set $f_1(x) = \sum_{i=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, x)$ $\forall x \in \mathbb{R}^p$.

Finally, note that the second term of the objective function is given by

$$\begin{aligned}
\lambda ||f||_{\mathcal{H}_k}^2 = \lambda \langle f, f \rangle_k &= \lambda \langle f_1 + f_2, f_1 + f_2 \rangle_k \\
&= \lambda \langle f_1, f_1 \rangle_k + \lambda \langle f_2, f_2 \rangle_k + 2\lambda \langle f_1, f_2 \rangle_k \\
&= \lambda ||f_1||_{\mathcal{H}_k}^2 + \lambda ||f_2||_{\mathcal{H}_k}^2 + 0.
\end{aligned}$$

This clearly this is minimised when $f_2 = 0$ and hence $||f_2||_{\mathcal{H}_k}^2 = 0$ in which case we arrive at the desired result:

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

Question 3

Since question 2 has showed us that $\hat{f}_\lambda(x_i^0) = \sum_{j=1}^n \hat{\beta}_{\lambda,i} k(x_i^0, x_j^0)$, rewriting the first term of the objective function is trivial. What remains is to show that the norm inside the second term of the objective function can be written as follows:

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

where K is the $n \times n$ matrix with entries $K_{ij} = k(x_i^0, x_j^0)$.

Hence the objective function then becomes

$$\hat{\gamma}_\lambda = (\hat{\alpha}_\lambda, \hat{\phi}_\lambda, \hat{\beta}_\lambda) \in \underset{\alpha\in\mathbb{R},\phi\in(0,\infty),\beta_\lambda\in\mathbb{R}^p}{\operatorname{argmax}} \frac{1}{2n}\sum_{i=1}^{n} \log \tilde{f}(y_i; g^{-1}(\alpha + \sum_{j=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, x_j^0)), \phi) - \lambda \hat{\beta}_\lambda^T K \hat{\beta}_\lambda.$$

Question 4

To reduce the dimension of the optimisation problem from $n+2$ to m we can use the Nyström method. This method is based on the idea that we can approximate the kernel matrix K by a matrix $\tilde{K}^{(d)}$ of rank at most $d := m-2$ such that $\tilde{K}^{(d)} \approx K$. In particular, we let

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

where by $K_{a,b}$ we denote the submatrix of K consisting of the first a rows and b columns. (Note in particular that this means that $K_{n,d} = K_{d,n}^T$.)

Now we may examine how the function $\hat{f}_\lambda$ is affected by the approximation $\tilde{K}^{(d)}$, i.e. using the Nyström kernel

$$\tilde{k}^{(d)}(x, \cdot) = k_d(x)^T (K_{d,d})^{-1} k_d(\cdot)$$

where $k_d(\cdot) = (k(x_1^0, \cdot), ..., k(x_d^0, \cdot)) \in \mathbb{R}^d$. Observe that

$$\begin{aligned}
\hat{f}_\lambda(\cdot) = \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, \cdot) &\approx \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} \tilde{k}^{(d)}(x_i^0, \cdot) \\
&= \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k_d(x_i^0)^T (K_{d,d})^{-1} k_d(\cdot) \\
&= \left(\sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k_d(x_i^0)^T \right) (K_{d,d})^{-1} k_d(\cdot) \\
&= \hat{\beta}_\lambda K_{n,d} (K_{d,d})^{-1} k_d(\cdot) \\
&= \hat{\nu}_\lambda^T k_d(\cdot)
\end{aligned}$$

where $\hat{\nu}_\lambda^T := \hat{\beta}_\lambda^T K_{n,d}(K_{d,d})^{-1} \in \mathbb{R}^d$ (and thus $\hat{\nu}_\lambda = (K_{d,d})^{-1} K_{n,d}^T \hat{\beta}_\lambda$ since $K_{d,d}$ is diagonal means that so is its inverse).

Now we have parameters $\hat{\nu}_\lambda \in \mathbb{R}^d$, $\alpha \in \mathbb{R}$ and $\hat{\phi} \in (0, \infty)$, i.e. we have reduced the dimension of the optimisation problem from $n+2$ to $d+2 = m$. However, we need to reformulate the penalty term in the objective function to work with our new parameter $\hat{\nu}_\lambda$ instead of $\hat{\beta}_\lambda$. We can do this by noting that

$$\begin{aligned}
\hat{\beta}_\lambda^T K \hat{\beta}_\lambda \approx \hat{\beta}_\lambda^T \tilde{K}^{(d)} \hat{\beta}_\lambda &= \hat{\beta}_\lambda^T K_{n,d} (K_{d,d})^{-1} K_{n,d}^T \hat{\beta}_\lambda \\
&= \left(\hat{\beta}_\lambda^T K_{n,d} (K_{d,d})^{-1} \right) (K_{d,d}) \left((K_{d,d})^{-1} K_{n,d}^T \hat{\beta}_\lambda \right) \\
&= \hat{\nu}_\lambda^T K_{d,d} \hat{\nu}_\lambda.
\end{aligned}$$

Thus, the objective function of the m-dimensional optimisation problem is:

$$(\hat{\alpha}_\lambda, \hat{\phi}_\lambda, \hat{\nu}_\lambda) \in \underset{\alpha\in\mathbb{R},\phi\in(0,\infty),\nu\_lambda\in\mathbb{R}^p}{\operatorname{argmax}} \frac{1}{2n}\sum_{i=1}^{n} \log \tilde{f}(y_i; g^{-1}(\alpha + \nu_\lambda^T k_d(x_i^0)), \phi) - \lambda \nu_\lambda^T K_{d,d} \nu_\lambda.$$

Question 5

In order to use the `R` package `glmnet` in solving this problem, we need to convert the penalty term into an λ multiplied by an L2 norm (an L1 norm would also work with `glmnet`, however, we'll see that an L2 norm arises fairly easily after some manipulation).

We do this by first obtaining the spectral decomposition of $K_{d,d}$, i.e. $K_{d,d} = UDU^T$ where U is an orthogonal matrix and D is a diagonal matrix. With this, the penalty term easily takes on the desired L2 norm form:

$$\begin{aligned}\lambda\hat{\nu}_\lambda^T K_{d,d}\hat{\nu}_\lambda = \lambda\hat{\nu}_\lambda^T UDU^T\hat{\nu}_\lambda &= \lambda(\hat{\nu}_\lambda^T UD^{1/2})(D^{1/2}U^T\hat{\nu}_\lambda) \\ &= \lambda(D^{1/2}U^T\hat{\nu}_\lambda)^T(D^{1/2}U^T\hat{\nu}_\lambda) \\ &= \lambda||D^{1/2}U^T\hat{\nu}_\lambda||_2^2.\end{aligned}$$

However, `glmnet` actually requires that the norm is taken on the parameter that is being multiplied by the data inside the objective function. Previously this parameter was $\hat{\nu}_\lambda^T$ being multiplied by our (kernelised) data $k_d(x_i^0)$, but now we want to transform the data so that the corresponding parameter is $(D^{1/2}U^T\hat{\nu}_\lambda)^T$. We do this by observing that

$$\begin{aligned}\hat{\nu}_\lambda^T k_d(x_i^0) &= \hat{\nu}_\lambda^T(UD^{1/2}D^{-1/2}U^T)k_d(x_i^0) \\ &= (\hat{\nu}_\lambda^T UD^{1/2})(D^{-1/2}U^T k_d(x_i^0)) \\ &= (D^{1/2}U^T\hat{\nu}_\lambda)^T(D^{-1/2}U^T k_d(x_i^0))\end{aligned}$$

Hence denoting our transformed data as $x_i^{\text{new}} = D^{-1/2}U^T k_d(x_i^0)$ we would call `glmnet` with the command `glmnet(X_new, y, alpha = 0)` (see question 6) where $X_{\text{new}} = (D^{-1/2}U^T K_{d,n})^T = K_{n,d}UD^{-1/2}$. (Note that we have also set `alpha = 0` since we are using an L2 norm penalty, rather than an L1 norm penalty that would require `alpha=1`).

This is solving the same optimisation problem as in the previous question, however, it might be clearer to now write the objective function as

$$\begin{aligned}(\hat{\alpha}_\lambda, \hat{\phi}_\lambda, \hat{\nu}_\lambda) \in \underset{\alpha\in\mathbb{R},\phi\in(0,\infty),\nu\lambda\in\mathbb{R}^p}{\text{argmax}} \frac{1}{2n}\sum_{i=1}^n \log \tilde{f}(y_i; g^{-1}(\alpha + (D^{1/2}U^T\nu_\lambda)^T x_i^{\text{new}}), \phi) \\ - \lambda||D^{1/2}U^T\nu_\lambda||_2^2.\end{aligned}$$

Task 2

For this task we will be using the `wesdr` dataset (from the `R` package `gss`) from the Wisconsin Epidemiological Study of Diabetic Retinopathy. This dataset contains 669 observations 3 continuous input variables and a single binary output variable `ret` indicating retinopathy progression, hence we will use a binomial link function for our model.

```
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
```

```
dim(wesdr)
```

```
## [1] 669   4
```

We perform an 80/20 train/test split of the data.

```
p = ncol(wesdr) - 1

propTrain = 0.8
trainIdx  = sample(1:nrow(wesdr), nrow(wesdr)*0.8)

XTrain = wesdr[trainIdx,-(p+1)]
yTrain = wesdr[trainIdx, p+1]

n = nrow(XTrain)

XTest = wesdr[-trainIdx, -(p+1)]
yTest = wesdr[-trainIdx, p+1]

dim(XTrain); length(yTrain)
```

```
## [1] 535   3
```

```
## [1] 535
```

```
dim(XTest); length(yTest)
```

```
## [1] 134   3
```

```
## [1] 134
```

We will also be using a collection $\{\tilde{k}_c, c \in \mathcal{C}\}$ of RBF kernels $k(x, x') = \exp(-\sigma||x - x'||||^2)$ with bandwidths given by $\sigma = c \in \mathcal{C} = \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

```
library(kernlab)
bandwidths = c(0.001, 0.01, 0.1, 1, 10, 100, 1000)
```

Question 6

Following the results shown in question 5, we arrive at the following function for finding an approximate solution to the m-dimensional optimization problem for a given λ, c and m.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
question6 <- function(lambda, c, m){
  d = m - 2
  rbf = rbfdot(c)

  K0 = kernelMatrix(rbf, as.matrix(XTrain))


  K_dd = K0[1:d, 1:d]
  K_nd = K0[1:n, 1:d]

  decomp = eigen(K_dd)
```

```
  D = diag(decomp$values)
  U = decomp$vectors

  # X_new = (solve(D)**0.5) %*% t(U) %*% K_nd
  X_new = K_nd %*% U %*% (solve(D)**0.5)

  model = glmnet(X_new, yTrain, family=binomial(link="logit"), alpha = 0)

  return(model)
}
```

Question 7

We do indeed get numerical errors for very small c e.g. with $c = 0.001$.

```
model = question6(1,0.001,100)
```

```
## Error in solve.default(D): system is computationally singular: reciprocal condition number = 2.20464
```

A solution for this is to add a small value ϵ to the diagonal of K^0, which should make inverting the kernel matrix more numerically stable.

```
getTransformedX <- function(lambda, c, m, epsilon=1e-13){
  d = m - 2
  rbf = rbfdot(c)

  K0 = kernelMatrix(rbf, as.matrix(XTrain))

  # Add a small value to the diagonal of the kernel matrix
  K0 = K0 + epsilon*diag(1, nrow = nrow(K0))

  K_dd = K0[1:d, 1:d]
  K_nd = K0[1:n, 1:d]

  decomp = eigen(K_dd)
  D = diag(decomp$values)
  U = decomp$vectors

  # XTrain_new = (solve(D)**0.5) %*% t(U) %*% t(K_nd)
  XTrain_new = K_nd %*% U %*% (solve(D)**0.5)

  K_nd_Test = matrix(rep(0, nrow(XTest)*d), nrow=nrow(XTest), ncol=d)
  for (i in 1:nrow(XTest)){
    for (j in 1:d){
      diff = as.matrix(XTrain[j,] - XTest[i,])
      K_nd_Test[i,j] = exp(-c * sum(diff**2))
    }
  }
  XTest_new = K_nd_Test %*% U %*% (solve(D)**0.5)

  return(list(train=XTrain_new, test=XTest_new))
}

question7 <- function(lambda, c, m, epsilon=1e-13){
  d = m - 2
```

```
  X_new = getTransformedX(lambda, c, m, epsilon)$train
  model = glmnet(X_new, yTrain, family=binomial(link="logit"), alpha = 0)

  return(model)
}
```

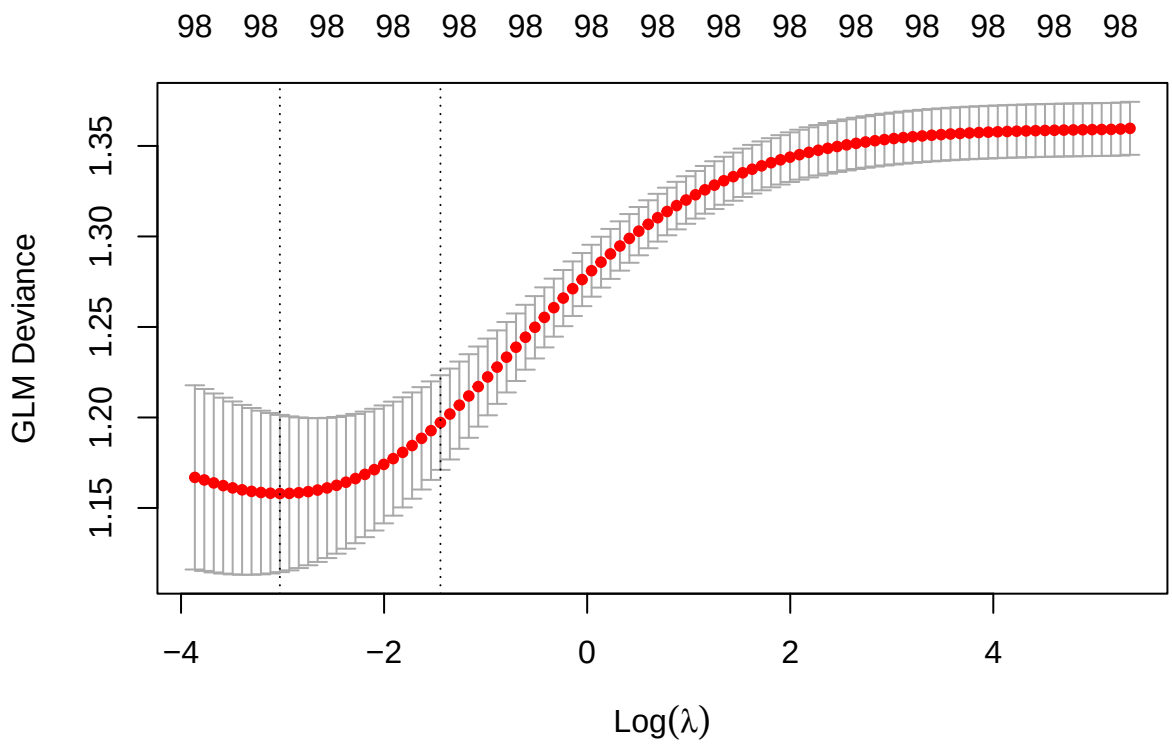
Here we can then see that we no longer get an error with $c = 0.001$.

```
model = question7(1,0.001,100)
```

Question 8

We use `cv.glmnet`'s default $K = 10$-fold cross validation to choose the value of λ in the function below for a given c and m. (We plot this below for $c = 0.001, m = 100$.)

```
question8 <- function(c, m, epsilon=1e-13, plot=FALSE){
  d = m - 2

  X_new = getTransformedX(lambda, c, m, epsilon)$train

  cv = cv.glmnet(X_new, yTrain, family=binomial(link="logit"), alpha = 0)#, lambdas=c(0.001, 0.01, 0.1,
  optLambda = cv$lambda.min

  if (plot) plot(cv)

  question7(optLambda, c, m, epsilon=1e-13)
}

model = question8(0.001, 100, plot=TRUE)
```



Question 9

Here we use a grid-search to continue the cross-validation from question 8 in order to choose c and λ for a given m: we choose the c and λ with the minimal the cross-validation error as found in the previous question's 10-fold cross validation for choosing λ based on a given c (and m).

```
question9 <- function(m, epsilon=1e-13){
  d = m - 2

  best_CVErr = Inf
  best_c = NA
  best_lambda = NA

  for(c in bandwidths){
    X_new = getTransformedX(lambda, c, m, epsilon)$train

    ridgeCV = cv.glmnet(X_new, yTrain, family=binomial(link="logit"), alpha = 0) #k=10-fold cv by defau
    optCVErr = ridgeCV$cvm
    if (min(optCVErr) < best_CVErr){
      best_CVErr = min(optCVErr)
      best_c = c
      best_lambda = ridgeCV$lambda.min
    }
  }

  return(list(model = question7(best_lambda, best_c, m, epsilon=1e-13), c=best_c, lambda = best_lambda)
}

model = question9(100)
```

Question 10

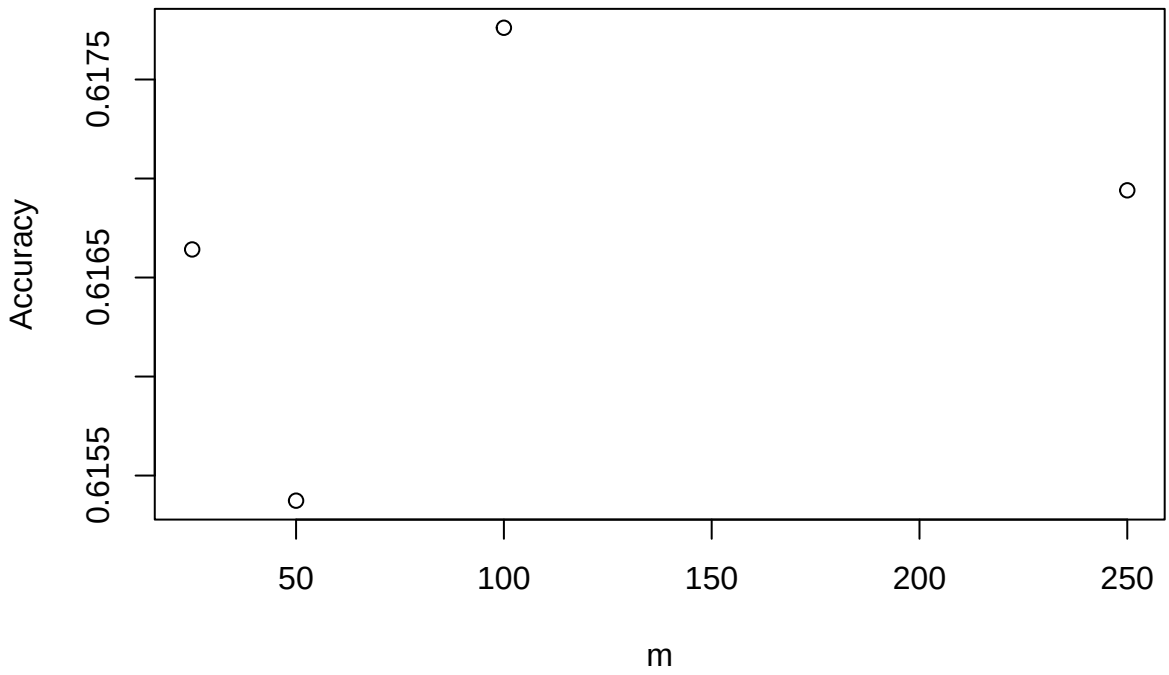
We will fit our model for $m \in \{25, 50, 100, 250\}$, using the function from the previous question to find optimal values for λ and c each time.

```
testM <- function(m, epsilon=1e-13){
  output = question9(m, epsilon)
  model = output$model
  lambda = output$lambda
  c = output$c

  XTest_new = getTransformedX(lambda, c, m, epsilon)$test

  yTest_pred = predict(model, XTest_new)

  mean((yTest_pred > 0) == yTest)
}

ms = c(25,50,100,250)
accs = c()

for (m in ms){
  accs = c(accs, testM(m))
}
```

```
data.frame(m=ms, accuracy=accs)
```

```
##     m  accuracy
## 1  25 0.6166418
## 2  50 0.6153731
## 3 100 0.6177612
## 4 250 0.6169403
```

```
plot(ms, accs, xlab="m", ylab="Accuracy")
```



We see that the model obtains a best accuracy of 61.8% with $m = 100$ and a worst accuracy of 61.5% with $m = 50$. This is definitely better than random chance, however, it is far from being very good. It may be the case that a better family of kernels (or even RBF kernels but with a different variety of bandwidths) would improve this result, and we might also want to consider increasing the range of values tested in cross validation for λ.

Question 11

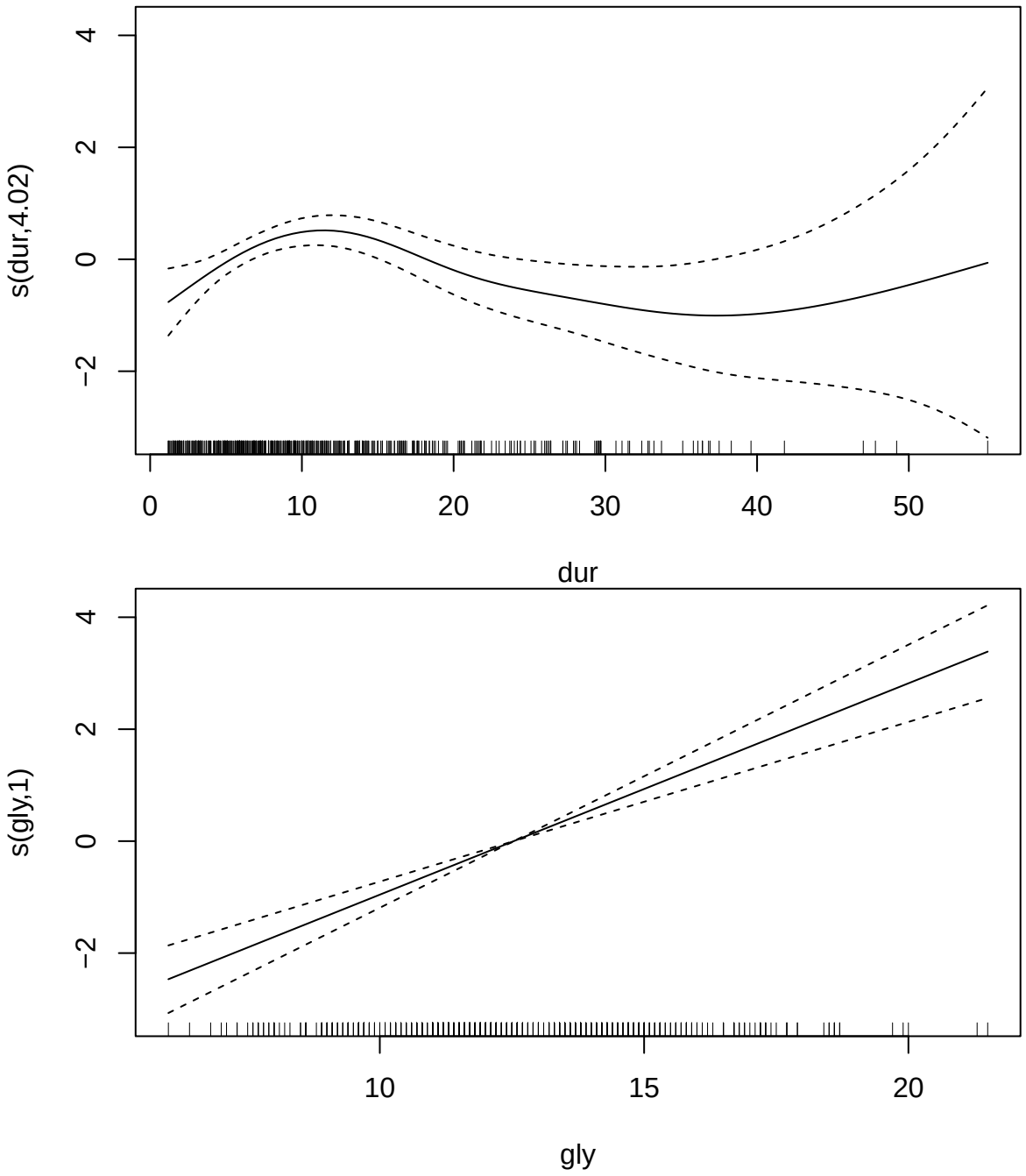
In comparison, computing a GAM estimate of the model results in an accuracy of 67.9%, a better result than any of our previous models in this investigation.

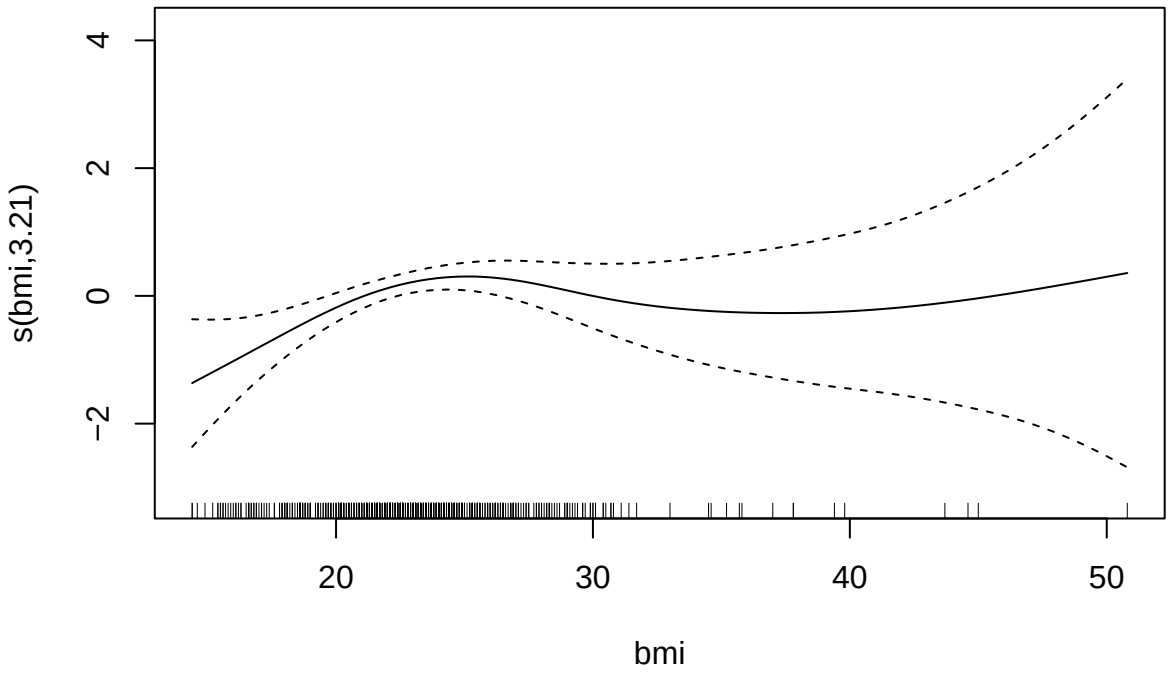
```
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
train = wesdr[trainIdx,]
model = gam(ret ~ s(dur) + s(gly) + s(bmi), family=binomial(link="logit"), data=train)
plot(model)
```


s(dur,4.02)
4
2
0
−2
0
10
20
30
40
50
dur
s(gly,1)
4
2
0
−2
10
15
20
gly




```
summary(model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## ret ~ s(dur) + s(gly) + s(bmi)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.4494     0.1011  -4.447 8.72e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df Chi.sq p-value
## s(dur) 4.024  4.992  20.24 0.00115 **
## s(gly) 1.000  1.000  66.83 < 2e-16 ***
## s(bmi) 3.210  4.051  12.00 0.01801 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.229   Deviance explained = 19.1%
## UBRE = 0.13214  Scale est. = 1         n = 535
```

```
yTestPrediction = predict(model, XTest) > 0
mean(yTest == yTestPrediction)
```

```
## [1] 0.6791045
```