BIOST 527 Homework 3

Benjamin Stan

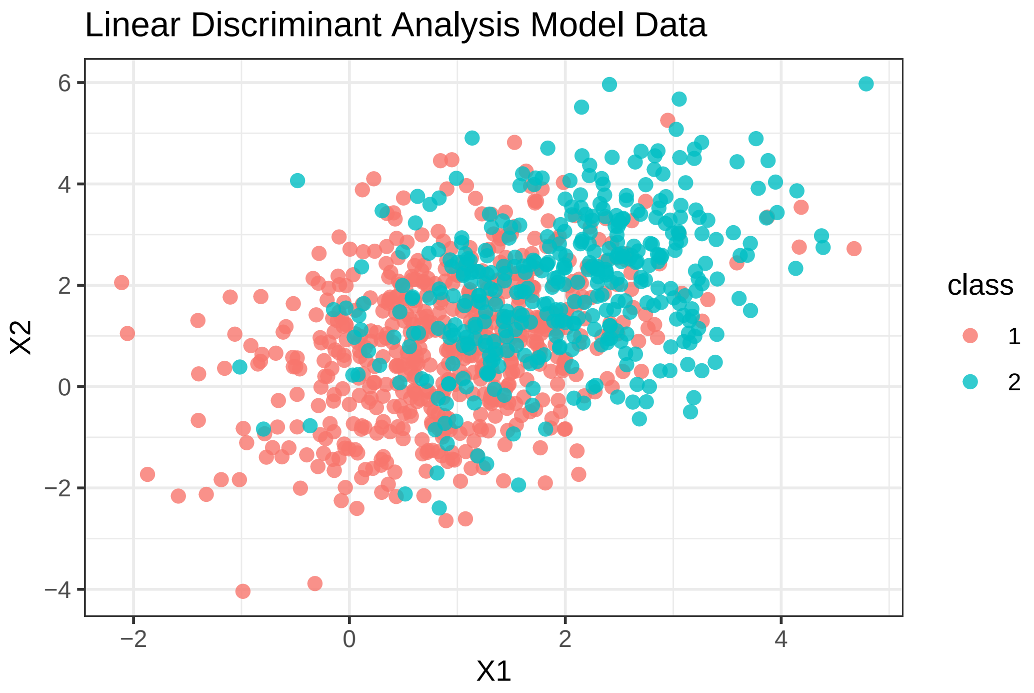
May 10, 2021

Question 1

(a)

In order to generate observations from the LDA model, two classes of data were generated using the `mvnorm` function, which generates data from a multivariate normal distribution. The means for classes 1 and 2 were 1 and 2, respectively. 600 observations were generated for class 1, and 400 were generated for class 2. The covariance matrix for both classes is shown below.

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$$



(b)

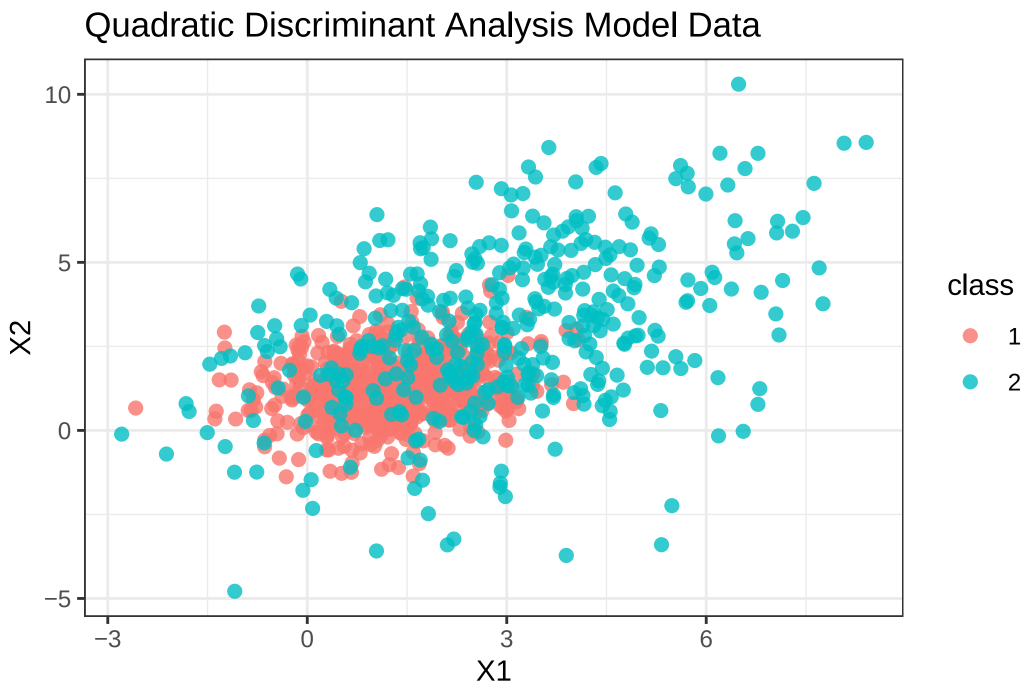
To evaluate the performance of LDA and QDA on the LDA model data, the observations were split into train and test sets with 0.7/0.3 probability. Below is a table with the training and test error for the two methods.

	Training Error	Test Error
LDA	25.3%	28.0%
QDA	25.1%	28.0%

(c)

In order to generate observations from the QDA model, two classes of data were similarly generated using the `mvnorm` function. As before, 600 observations were generated for class 1, and 400 were generated for class 2. The distinction from part (a) is that the two classes were drawn from distributions with different covariance matrices. The means for classes 1 and 2 were 1.2 and 3, respectively. The covariance matrices for the two classes are shown below.

$$\text{Class 1: } \begin{pmatrix} 1 & 0.2 \\ 0.2 & 1 \end{pmatrix}$$
$$\text{Class 2: } \begin{pmatrix} 4 & 2 \\ 2 & 6 \end{pmatrix}$$



(d)

To evaluate the performance of LDA and QDA on the QDA model data, the observations were split into train and test sets with 0.7/0.3 probability. Below is a table with the training and test error for the two methods.

	Training Error	Test Error
LDA	19.6%	20.3%
QDA	14.9%	18.0%

(e)

In both data simulations, the simpler LDA model was compared to the more complex QDA model. One interesting finding was that the QDA performed comparably to the LDA on the data drawn from the LDA model. This can be explained by the idea that it would be possible to fit a quadratic decision boundary to data coming from a linear boundary assumption, but the quadratic term would be very small. In both data-generating models, the difference between train and test error was smaller for LDA than for QDA, despite the comparable or improved performance of QDA in terms of reducing error. This can be explained by the bias-variance trade-off because the simpler LDA model may not be as accurate in terms of test error, but its variance is also lower, leading to more comparable error when considering a test set. By comparison, the QDA model may overfit on the training set, which would lead to a greater discrepancy between training and test error.

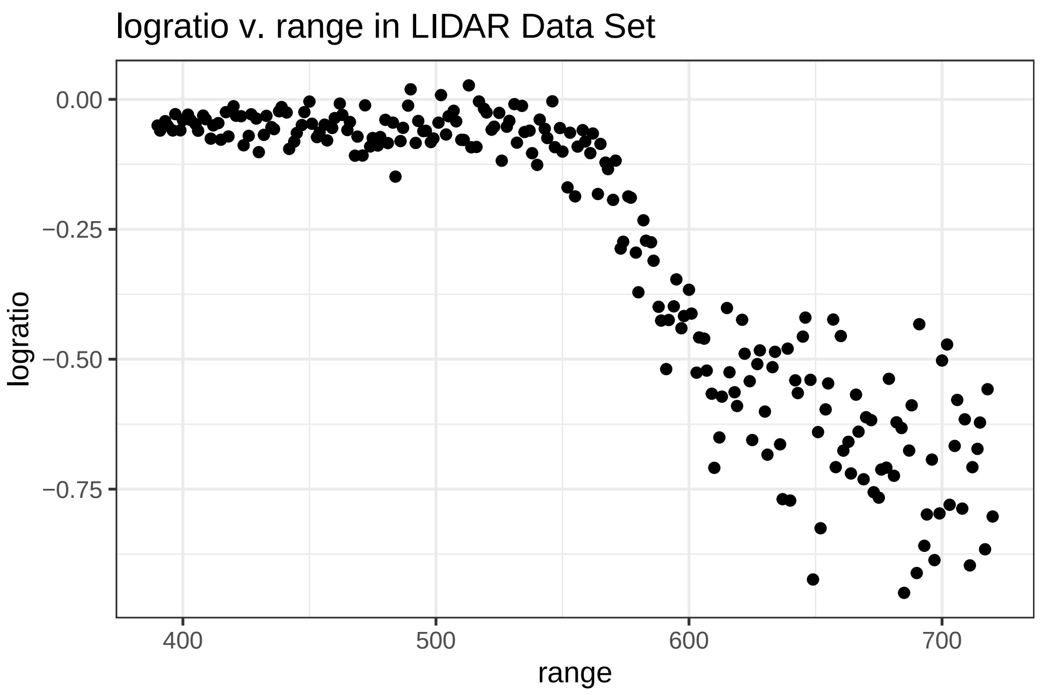
Question 2

(a)

From the LIDAR data documentation, the data set contains 221 observations of the following variables:

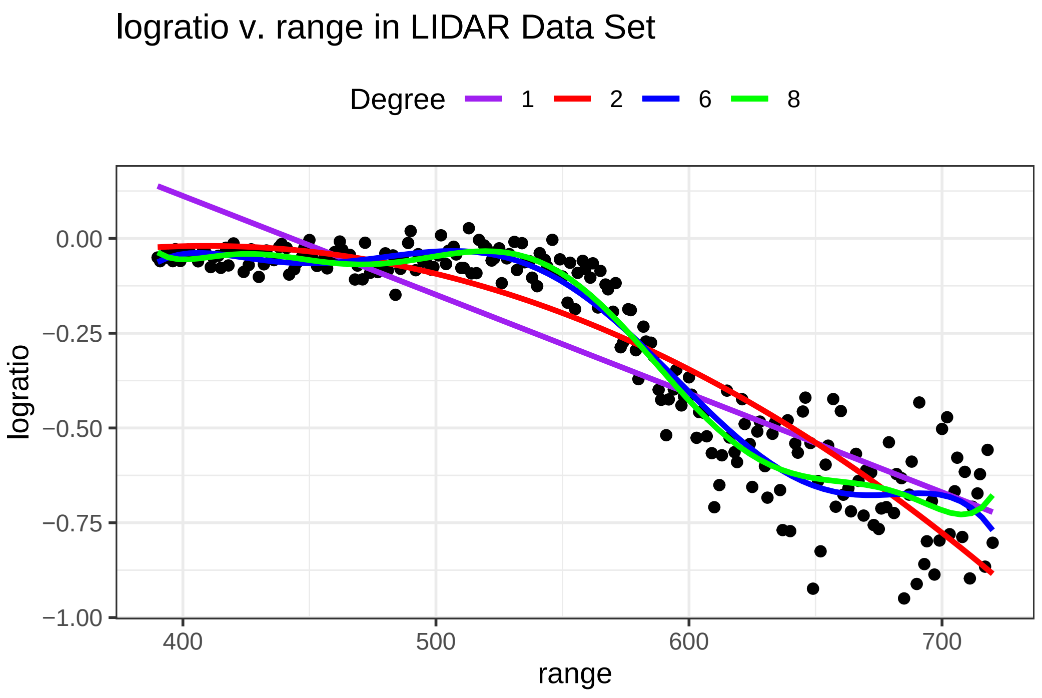
- `range`: Distance traveled before the light is reflected back to its source
- `logratio`: Logarithm of the ratio of received light from two laser sources

Based on this information, the response will be the logratio of received light, and the predictor will be the range. This is due to the relationship of the received light being dependent on the range. The plot below shows `logratio` v. `range`.



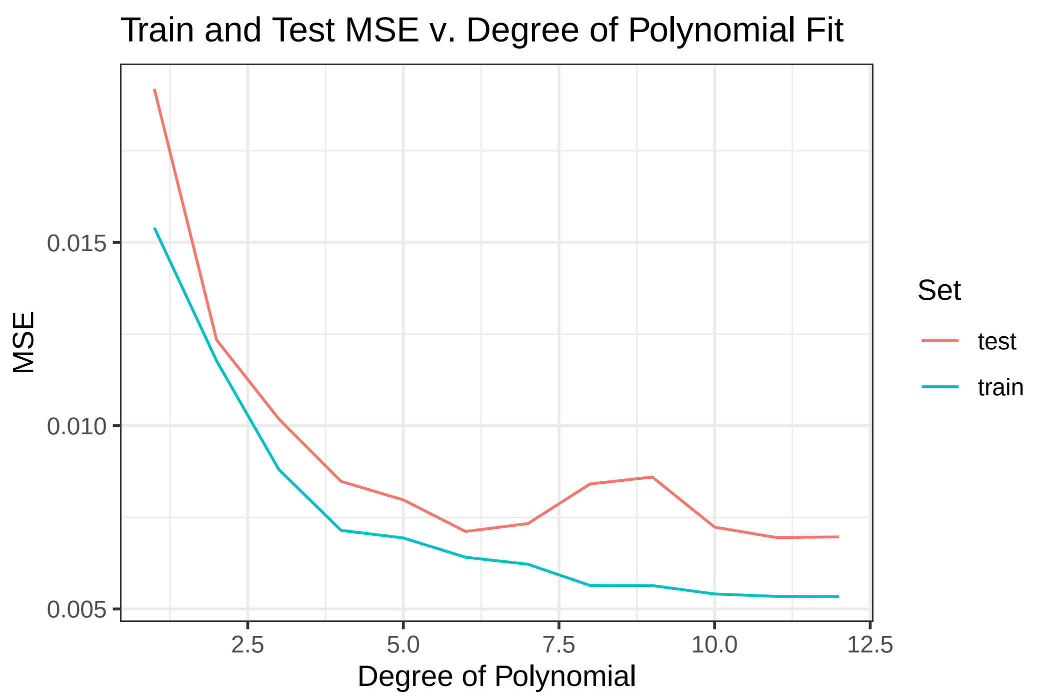
(b)

Below is the plot of polynomial fits. Note that in this approach, the degree of the polynomial was interpreted to be the highest power of the polynomial, i.e. a sixth degree polynomial has a term X^6.



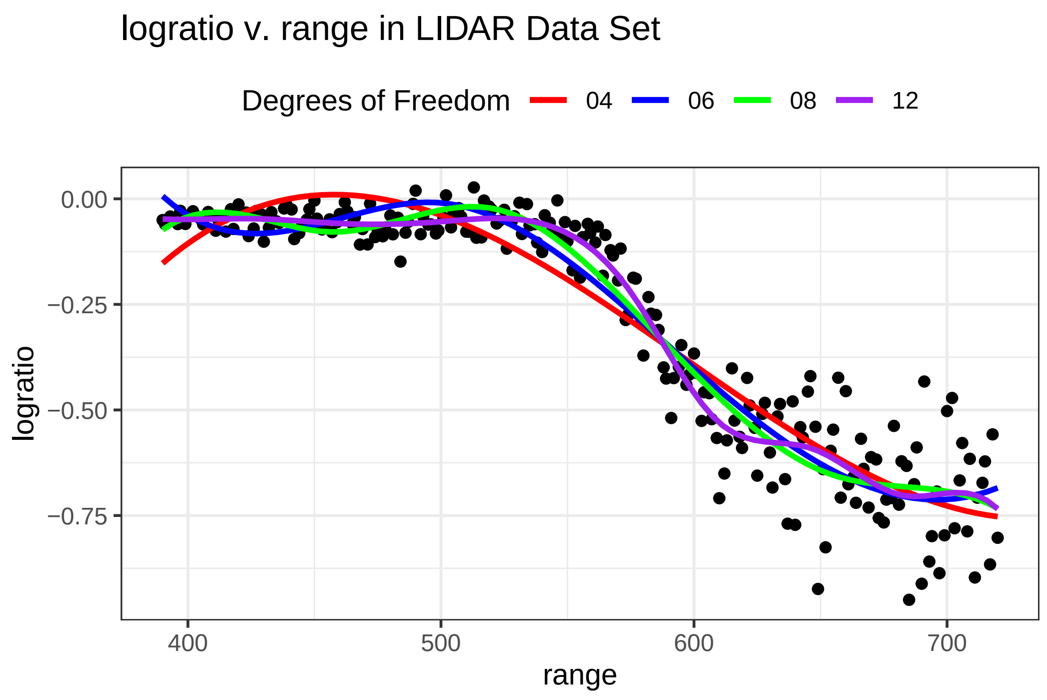
(c)

To compare polynomials of varying degrees, the training and test errors were computed for degrees from 1 to 12. As the degree increased, the training error decreased across all values considered. The test error decreased with increasing polynomial degree up to degree 6, after which point it ceased to monotonically decrease. We conclude that degree 6 provides the best prediction performance in this situation, as higher degrees appear to lead to overfitting and undesirable patterns in test error.



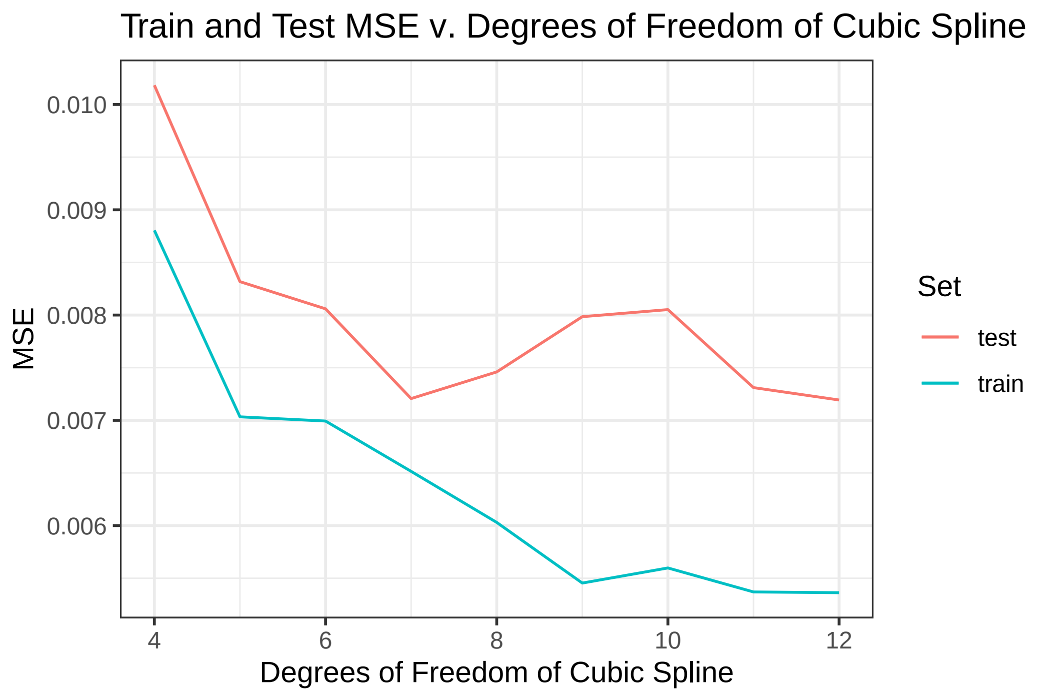
(d)

Below is the plot of cubic regression spline fits.



(e)

To compare cubic regression splines, the training and test errors were computed for degrees of freedom from 4 to 12. As the degrees of freedom increased, the training error tended to decrease, with the exception of one increase from 9 to 10 degrees of freedom. The test error decreased with increasing degrees of freedom up to 7, after which point it ceased to monotonically decrease. We conclude that 7 degrees of freedom provides the best prediction performance in this situation, as higher degrees appear to lead to overfitting and undesirable patterns in test error.

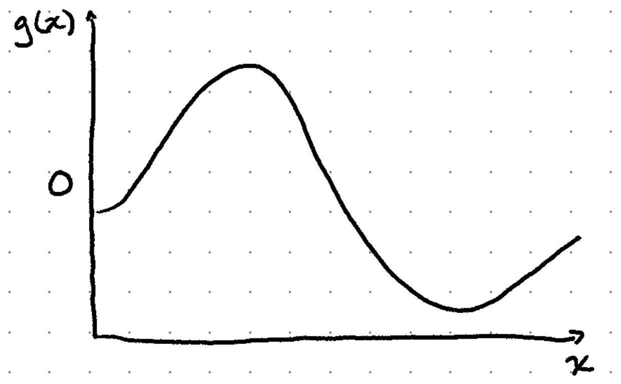


Question 3

Supposing a curve g satisfies

$$\hat{g} = \arg\,\min_g \left(\sum_{i=1}^{N} (y_i - g(x_i))^2 + \lambda \int [g^{(m)}(t)]^2 dt \right)$$

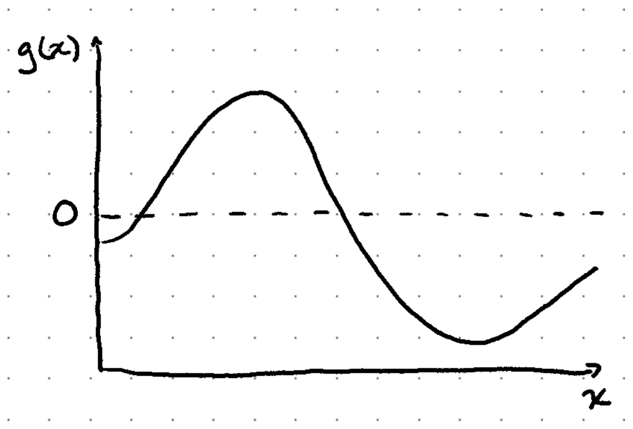
Below is the sketch of an arbitrarily defined data-generating function $g(x)$, which will be used for reference in the different scenarios. This function will be indicated by a solid line, while the fitted function $\hat{g}$ will be indicated by a dashed line.



(a)

For $\lambda \to \infty, m = 0$:

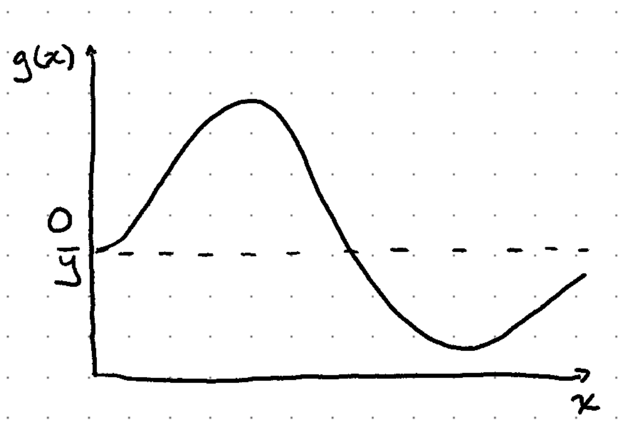
This scenario causes the penalty term on $g(x)$ to increase, forcing the function $g(x)$ and the fitted value $\hat{g}$ to 0; $\hat{g} = 0$.



(b)

For $\lambda \to \infty, m = 1$:

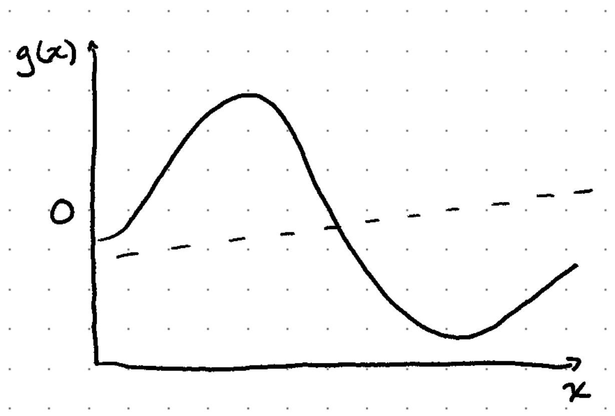
This scenario causes the penalty term on $g'(x)$ to increase, forcing the first derivative to 0. The fitted value is the constant which minimizes the RSS in the first term, which means $\hat{g} = \bar{y}$.



(c)

For $\lambda \to \infty, m = 2$:

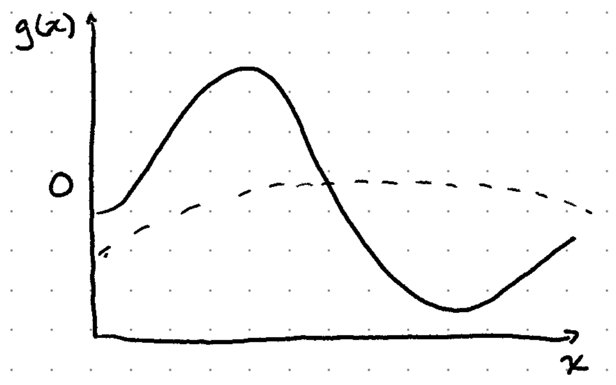
This scenario causes the penalty term on $g''(x)$ to increase, forcing the second derivative to 0. The fitted value is the linear function which minimizes the RSS in the first term, which means $\hat{g}$ is the linear least squares line, $\hat{g} = \beta_0 + \beta_1 x$.



(d)

For $\lambda \to \infty, m = 3$:

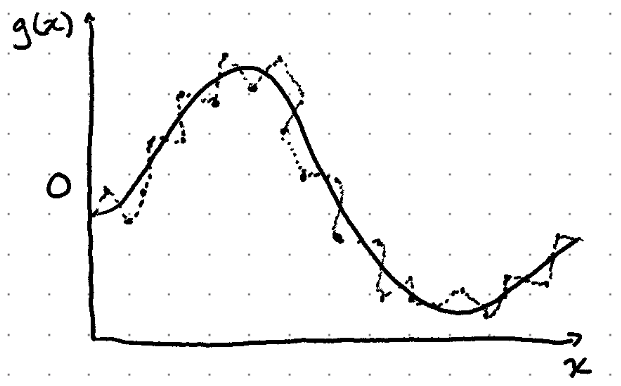
This scenario causes the penalty term on $g'''(x)$ to increase, forcing the third derivative to 0. The fitted value is the quadratic function which minimizes the RSS in the first term, which means $\hat{g}$ is the quadratic least squares line, $\hat{g} = \beta_0 + \beta_1 x + \beta_2 x^2$.



(e)

For $\lambda \to 0, m = 3$:

This scenario will remove the penalty term, leading to the solution that minimizes the RSS, which would be interpolation of the observations. The observations from the sample are shown in the plot below along with the data-generating function, $g(x)$ and the fitted values $\hat{g}$.



Question 4

The goal is to prove that any linear combination of the functions

$$
\begin{aligned}
h_1(X) &= 1 \\
h_2(X) &= X \\
h_3(X) &= X^2 \\
h_4(X) &= X^3 \\
h_5(X) &= (X - \psi_1)^3_+ \\
h_6(X) &= (X - \psi_2)^3_+
\end{aligned}
$$

is a cubic spline with knots at ψ_1 and ψ_2. In order to prove this, we will prove that $f(X) = \sum_{m=1}^{6} \beta_m h_m(x)$ satisfies:

- The function $f(X)$ is continuous and has continuous first and second derivatives

- The function $f(X)$ is cubic in all regions

First, we will prove that the function is continuous. For $h > 0$, consider the limits at the first knot, ψ_1, beginning with the left limit:

$$\begin{aligned}
f(\psi_1 - h) &= \beta_1 + \beta_2(\psi_1 - h) + \beta_3(\psi_1 - h)^2 + \beta_4(\psi_1 - h)^3 + \beta_5(\psi_1 - h - \psi_1)^3_+ + \beta_6(\psi_1 - h - \psi_2)^3_+ \\
&= \beta_1 + \beta_2(\psi_1 - h) + \beta_3(\psi_1 - h)^2 + \beta_4(\psi_1 - h)^3 \\
lim_{h\to 0} f(\psi_1 - h) &= \beta_1 + \beta_2(\psi_1) + \beta_3(\psi_1)^2 + \beta_4(\psi_1)^3
\end{aligned}$$

Compare this to the right limit:

$$\begin{aligned}
f(\psi_1 + h) &= \beta_1 + \beta_2(\psi_1 + h) + \beta_3(\psi_1 + h)^2 + \beta_4(\psi_1 + h)^3 + \beta_5(\psi_1 + h - \psi_1)^3_+ + \beta_6(\psi_1 + h - \psi_2)^3_+ \\
&= \beta_1 + \beta_2(\psi_1 + h) + \beta_3(\psi_1 + h)^2 + \beta_4(\psi_1 + h)^3 + \beta_5(h)^3_+ \\
lim_{h\to 0} f(\psi_1 + h) &= \beta_1 + \beta_2(\psi_1) + \beta_3(\psi_1)^2 + \beta_4(\psi_1)^3
\end{aligned}$$

Since the limits from the left and right are equivalent, the function is continuous at ψ_1. Similar logic can be used to prove continuity at the second knot, ψ_2, and thus prove that the function is continuous overall.

Next, we will prove, $f(X)$ is cubic in all regions. In order to prove this, we will consider three regions:

- $X \le \psi_1$
- $\psi_1 \le X \le \psi_2$
- $X \ge \psi_2$

We have already proven that the function is continuous across the three regions. Thus, for a cubic function, the desired form of function $f(X)$ can be defined

$$f(X) = \begin{cases} a_1 + b_1X + c_1X^2 + d_1X^3 \text{ for } X \le \psi_1 \\ a_2 + b_2X + c_2X^2 + d_2X^3 \text{ for } \psi_1 \le X \le \psi_2 \\ a_3 + b_3X + c_3X^2 + d_3X^3 \text{ for } X \ge \psi_2 \end{cases}$$

with constraints

$$\begin{aligned}
a_1 + b_1\psi_1 + c_1\psi_1^2 + d_1\psi_1^3 &= a_2 + b_2\psi_1 + c_2\psi_1^2 + d_2\psi_1^3 \\
a_2 + b_2\psi_2 + c_2\psi_2^2 + d_2\psi_2^3 &= a_3 + b_3\psi_2 + c_3\psi_2^2 + d_3\psi_2^3
\end{aligned}$$

Note that new coefficients are defined to distinguish from the $\beta_1, \dots, \beta_6$ coefficients above. It is evident that the criteria of being cubic is satisfied by the piecewise function in the first region ($X \le \psi_1$). To prove that any linear combination of the basis functions is cubic in the second region, consider a coefficient γ on $h_5(X)$. Thus, in this region, the function $f(X)$ would take the form

$$\begin{aligned}
f(X) &= a_1 + b_1X + c_1X^2 + d_1X^3 + \gamma(X - \psi_1)^3 \\
&= a_1 + b_1X + c_1X^2 + d_1X^3 + \gamma(X^3 - 3\psi_1X^2 + 3X\psi_1^2 - \psi_1^3) \\
&= (a_1 - \gamma\psi_1^3) + (b_1 + 3\gamma\psi_1^2)X + (c_1 - 3\gamma\psi_1)X^2 + (d_1 + \gamma)X^3
\end{aligned}$$

Equating this with the desired form, $a_2 + b_2X + c_2X^2 + d_2X^3$, to define the new coefficients

$$\begin{aligned}
a_2 &= a_1 - \gamma\psi_1^3 \\
b_2 &= b_1 + 3\gamma\psi_1^2 \\
c_2 &= c_1 - 3\gamma\psi_1 \\
d_2 &= d_1 + \gamma
\end{aligned}$$

Therefore, in the second region, the function $f(X)$ is a cubic function. To prove that any linear combination of the basis functions is cubic in the third region, consider a coefficient η on $h_6(X)$. Thus, in this region, the function $f(X)$ would take the form

$$\begin{aligned}
f(X) &= a_2 + b_2X + c_2X^2 + d_2X^3 + \eta(X - \psi_2)^3 \\
&= (a_2 - \eta\psi_2^3) + (b_2 + 3\eta\psi_2^2)X + (c_2 - 3\eta\psi_2)X^2 + (d_2 + \eta)X^3
\end{aligned}$$

Equating this with the desired form, $a_3 + b_3X + c_3X^2 + d_3X^3$, to define the new coefficients

$$\begin{aligned}
a_3 &= a_2 - \gamma\psi_2^3 \\
b_3 &= b_2 + 3\gamma\psi_2^2 \\
c_3 &= c_2 - 3\gamma\psi_2 \\
d_3 &= d_2 + \gamma
\end{aligned}$$

Thus, in the third region, the function $f(X)$ is a cubic function.

Lastly, we will use these cubic forms to prove that the function $f(X)$ has continuous first and second derivatives. Consider the first knot, ψ_1, and the derivative of the function from the left:

$$\begin{aligned}
f'_-(\psi_1) &= \frac{d}{dX}\big(a_1 + b_1X + c_1X^2 + d_1X^3\big)\Big|_{X=\psi_1} \\
&= \big(b_1 + 2c_1X + 3d_1X^2\big)\Big|_{X=\psi_1} \\
&= b_1 + 2c_1\psi_1 + 3d_1\psi_1^2
\end{aligned}$$

Compare this to the derivative from the right:

$$\begin{aligned}
f'_+(\psi_1) &= \frac{d}{dX}\big(a_2 + b_2X + c_2X^2 + d_2X^3\big)\Big|_{X=\psi_1} \\
&= \big(b_2 + 2c_2X + 3d_2X^2\big)\Big|_{X=\psi_1} \\
&= b_2 + 2c_2\psi_1 + 3d_2\psi_1^2 \\
&= (b_1 + 3\gamma\psi_1^2) + 2(c_1 - 3\gamma\psi_1)\psi_1 + 3(d_1 + \gamma)\psi_1^2 \\
&= b_1 + 3\gamma\psi_1^2 + 2c_1\psi_1 - 6\gamma\psi_1^2 + 3d_1\psi_1^2 + 3\gamma\psi_1^2 \\
&= b_1 + 2c_1\psi_1 + 3d_1\psi_1^2
\end{aligned}$$

As the derivative from the left and the derivative from the right are equivalent, the function $f(X)$ has a continuous derivative at ψ_1. Similar logic can be used to show that the derivative is continuous at ψ_2, and therefore, the function has a continuous first derivative.

For the second derivative continuity, consider the first knot, ψ_1, and the second derivative of the function from the left:

$$\begin{aligned}
f''_{-}(\psi_1) &= \frac{d^2}{dX^2}\left(a_1 + b_1X + c_1X^2 + d_1X^3\right)\Big|_{X=\psi_1} \\
&= \left(2c_1 + 6d_1X\right)\Big|_{X=\psi_1} \\
&= 2c_1 + 6d_1\psi_1
\end{aligned}$$

Compare this to the second derivative from the right:

$$\begin{aligned}
f''_{+}(\psi_1) &= \frac{d^2}{dX^2}\left(a_2 + b_2X + c_2X^2 + d_2X^3\right)\Big|_{X=\psi_1} \\
&= \left(2c_2 + 6d_2X\right)\Big|_{X=\psi_1} \\
&= 2c_2 + 6d_2\psi_1 \\
&= 2(c_1 - 3\gamma\psi_1) + 6(d_1 + \gamma)\psi_1 \\
&= 2c_1 - 6\gamma\psi_1 + 6d_1\psi_1 + 6\gamma\psi_1 \\
&= 2c_1 + 6d_1\psi_1
\end{aligned}$$

As the second derivative from the left and the second derivative from the right are equivalent, the function $f(X)$ has a continuous second derivative at ψ_1. Similar logic can be used to show that the second derivative is continuous at ψ_2, and therefore, the function has a continuous second derivative.

We have thus proven that the function $f(X)$, which is a linear combination of the given basis functions, is a cubic spline with knots at ψ_1 and ψ_2 by proving:

- The function $f(X)$ is continuous and has continuous first and second derivatives
- The function $f(X)$ is cubic in all regions

Question 5

For the least squares regression, the fitted value corresponding to x_0 is

$$\hat{f}(x_0) = x_0^T(X^TX)^{-1}X^T\vec{y}$$

Thus the variance corresponding to this estimate is

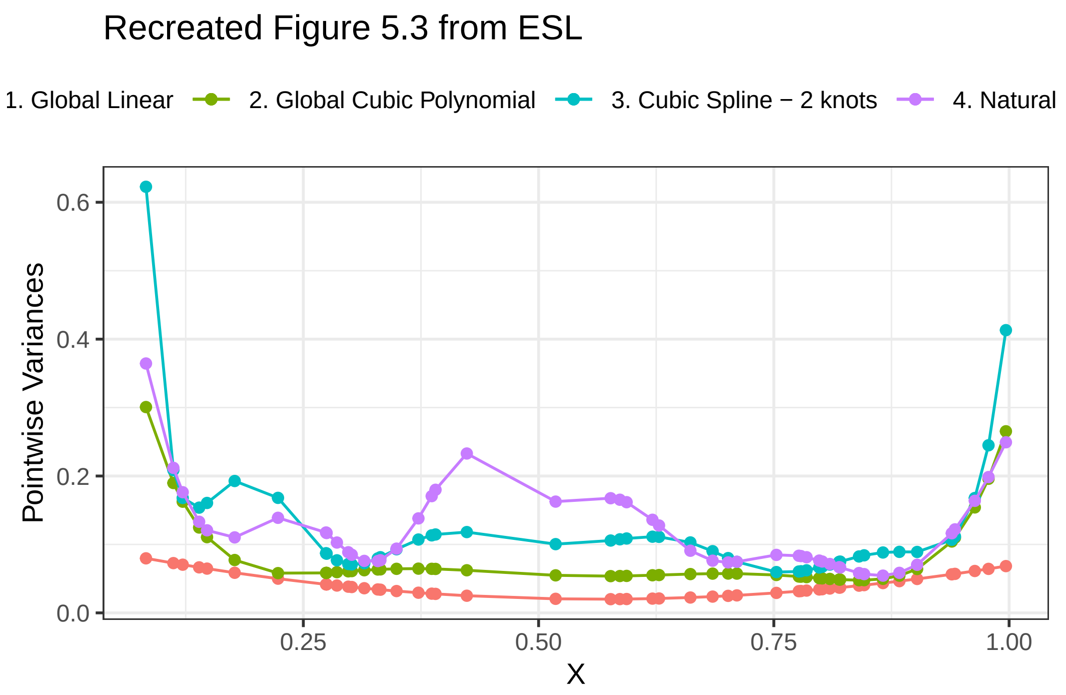
$$\begin{aligned}
Var(\hat{f}(x_0)) &= Var[x_0^T(X^TX)^{-1}X^T\vec{y}] \\
&= x_0^T(X^TX)^{-1}X^TVar(\vec{y})[x_0^T(X^TX)^{-1}X^T]^T \\
&= x_0^T(X^TX)^{-1}X^TVar(\vec{y})X[(X^TX)^T]^{-1}x_0 \\
&= x_0^T(X^TX)^{-1}X^TVar(\vec{y})X(X^TX)^{-1}x_0 \\
&\text{Note that } Var(\vec{y}) = \sigma^2 I \\
&= x_0^T(X^TX)^{-1}X^T\sigma^2 IX(X^TX)^{-1}x_0 \\
&= \sigma^2 x_0^T(X^TX)^{-1}x_0
\end{aligned}$$

Note that the different fits in Figure 5.3 can be generated by performing a least squares regression on an appropriately transformed X matrix. Figure 5.3 states that the assumed error model has constant variance, so for simplicity, we will let $\sigma^2 = 1$. We will also let the test observations x_0 come from the original X used to fit the model, as it will cover all possible values for x_0 needed to regenerate the plot. Rather than loop through all values of x_0 to calculate the variance at each point, we can take the diagonal of the following matrix, reducing computational load:

$$X(X^T X)^{-1} X^T$$

The code to regenerate Figure 5.3 from ESL and the reproduced plot are shown below.

```
######
# 5
######
## Define initial variables
rm(list = ls())
set.seed(23)
x = runif(50,0,1)

## Compute pointwise variance for linear model
x_lin = as.matrix(cbind(rep(1,50),x))
var_lin = diag(x_lin%*%solve(t(x_lin)%*%x_lin)%*%t(x_lin))
# Using loops
var_lin2 = NULL
for (i in 1:nrow(x_lin)) {
  var_i = x_lin[i,]%*%solve(t(x_lin)%*%x_lin)%*%x_lin[i,]
  var_lin2 = c(var_lin2,var_i)
}
comp_var = identical(var_lin,var_lin2) # TRUE

## Compute pointwise variance for polynomial with order 3
x_poly3 = as.matrix(cbind(rep(1,50),x,x^2,x^3))
var_poly3 = diag(x_poly3%*%solve(t(x_poly3)%*%x_poly3)%*%t(x_poly3))

## Compute pointwise variance for cubic spline with 2 knots
x_cubic = bs(x,degree=3,knots=c(0.33,0.66),intercept=TRUE)
var_cubic = diag(x_cubic%*%solve(t(x_cubic)%*%x_cubic)%*%t(x_cubic))

## Compute pointwise variance for natural spline with 6 knots
x_nat_cubic = ns(x,
                 df=6,
                 knots=c(0.26,0.42,0.58,0.74),
                 Boundary.knots = c(0.1,0.9),
                 intercept=TRUE)
var_nat_cubic = diag(x_nat_cubic%*%solve(t(x_nat_cubic)%*%x_nat_cubic)%*%t(x_nat_cubic))

## Plot results
ggplot() +
  geom_line(aes(x = x, y = var_lin, colour = "1. Global Linear")) +
  geom_point(aes(x = x, y = var_lin, colour = "1. Global Linear")) +
  geom_line(aes(x = x, y = var_poly3, colour = "2. Global Cubic Polynomial")) +
  geom_point(aes(x = x, y = var_poly3, colour = "2. Global Cubic Polynomial")) +
  geom_line(aes(x = x, y = var_cubic, colour = "3. Cubic Spline - 2 knots")) +
  geom_point(aes(x = x, y = var_cubic, colour = "3. Cubic Spline - 2 knots")) +
  geom_line(aes(x = x, y = var_nat_cubic, colour = "4. Natural Cubic Spline - 6 knots")) +
  geom_point(aes(x = x, y = var_nat_cubic, colour = "4. Natural Cubic Spline - 6 knots")) +
  labs(x='X', y='Pointwise Variances', colour="Model") +
  ggtitle("Recreated Figure 5.3 from ESL") +
  theme_bw() +
  theme(legend.position="top")
```



Question 6

We will consider the cubic spline with knots at $\psi_1 < \psi_2 < \cdots < \psi_K$, which takes the form

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \psi_k)_+^3$$

for coefficients $\beta_0, \beta_1, \beta_2, \beta_3$ and $\theta_1, \dots, \theta_K$.

(a)

If we want to include constraints that $f(X)$ is linear to the left of ψ_1 and to the right of ψ_K, it will require all coefficient on terms of X^2 and X^3 to be 0 in these regions. Consider the function to the left of ψ_1:

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

For this to be linear, $\beta_2 = \beta_3 = 0$. Now consider the function to the right of ψ_K:

$$
\begin{aligned}
f(X) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K} \theta_k (X - \psi_k)^3 \\
&= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K} \theta_k (X^3 - 3X^2\psi_k + 3X\psi_k^2 - \psi_k^3)
\end{aligned}
$$

Similar to the previous argument, we will set the coefficients on X^3 and X^3 to be zero. For X^3, this means

$$
\begin{gathered}
\beta_3 = 0 \\
\sum_{k=1}^{K} \theta_k = 0
\end{gathered}
$$

For X^2, this means

$$
\begin{gathered}
\beta_2 = 0 \\
-3\sum_{k=1}^{K} \theta_k \psi_k = 0 \Rightarrow \sum_{k=1}^{K} \theta_k \psi_k = 0
\end{gathered}
$$

(b)

The goal in this problem is to show that the basis functions

$$
\begin{aligned}
N_1 &= 1, \\
N_2 &= X, \\
N_{k+2} &= d_k(X) - d_{K-1}(X)
\end{aligned}
$$

where

$$
d_k(X) = \frac{(X - \psi_k)_+^3 - (X - \psi_K)_+^3}{\psi_K - \psi_k}
$$

have second and third derivatives that equal 0 to the left of ψ_1 and to the right of ψ_K. For $X < \psi_1$, all $N_{k+2}(X) = 0$ so

$$
\begin{aligned}
N_1(X) &= 1 \\
N_2(X) &= X \\
f(X) &= \beta_0 + \beta_1 X
\end{aligned}
$$

It is clear that the second and third derivative of this linear function are 0. For $X > \psi_1$, we have

$$
\begin{aligned}
N_1(X) &= 1\\
N_2(X) &= X\\
N_{k+2} &= \frac{(X-\psi_k)^3-(X-\psi_K)^3}{\psi_K-\psi_k} - \frac{(X-\psi_{K-1})^3-(X-\psi_K)^3}{\psi_K-\psi_{K-1}}\\
&= \frac{(X^3-3\psi_k X^2+3\psi_k^2 X-\psi_k^3)-(X^3-3\psi_K X^2+3\psi_K^2 X-\psi_K^3)}{\psi_K-\psi_k}\\
&- \frac{(X^3-3\psi_{K-1}X^2+3\psi_{K-1}^2 X-\psi_{K-1}^3)-(X^3-3\psi_K X^2+3\psi_K^2 X-\psi_K^3)}{\psi_K-\psi_{K-1}}\\
&= \left(\frac{-3\psi_k+3\psi_K}{\psi_K-\psi_k} - \frac{-3\psi_{K-1}+3\psi_K}{\psi_K-\psi_{K-1}}\right)X^2 + \left(\frac{3\psi_k^2-3\psi_K^2}{\psi_K-\psi_k} - \frac{3\psi_{K-1}^2-3\psi_K^2}{\psi_K-\psi_{K-1}}\right)X\\
&+ \left(\frac{-\psi_k+\psi_K}{\psi_K-\psi_k} - \frac{\psi_{K-1}+\psi_K}{\psi_K-\psi_{K-1}}\right)
\end{aligned}
$$

Consider the quadratic term:

$$
\begin{aligned}
\frac{-3\psi_k+3\psi_K}{\psi_K-\psi_k} - \frac{-3\psi_{K-1}+3\psi_K}{\psi_K-\psi_{K-1}} &= \frac{(-3\psi_k+3\psi_K)(\psi_K-\psi_{K-1})-(-3\psi_{K-1}+3\psi_K)(\psi_K-\psi_k)}{(\psi_K-\psi_k)(\psi_K-\psi_{K-1})}\\
&= \frac{-3\psi_k\psi_K+3\psi_k\psi_{K-1}+3\psi_K^2-3\psi_K\psi_{K-1}+3\psi_K\psi_{K-1}-3\psi_k\psi_{K-1}-3\psi_K^2+3\psi_k\psi_K}{(\psi_K-\psi_k)(\psi_K-\psi_{K-1})}\\
&= 0
\end{aligned}
$$

Thus the function to the right of ψ_K is linear with second and third derivatives equal to 0.

Question 7

(a)

The goal is to show that $S_\lambda = (I+\lambda K)^{-1}$ for some K.

We claim that the matrix N is invertible because it is a square matrix composed of the basis functions, which would make it linearly independent and thus full rank.

If N is invertible, S_λ can be written

$$
\begin{aligned}
S_\lambda &= (N^{-T}(N^T N+\lambda\Omega_N)N^{-1})^{-1}\\
&= (N^{-T}N^T N N^{-1} + N^{-T}\lambda\Omega_N N^{-1})^{-1}\\
&= (I+\lambda N^{-T}\Omega_N N^{-1})^{-1}\\
&= (I+\lambda K)^{-1}
\end{aligned}
$$

This matches the desired form with $K = N^{-T}\Omega_N N^{-1}$.

(b)

We begin with the equation $S_\lambda = (I+\lambda K)^{-1}$, and consider the eigendecomposition of K:

$$
K = UDU^T
$$

where D is the diagonal matrix with eigenvalues of K, and U contains the eigenvectors of K. If K is symmetric, U is an orthogonal matrix, meaning $UU^T = I$. Since U is a square matrix, this would also mean $U^T = U^{-1}$. To prove the symmetry of K, consider K^T:

$$K^T = (N^{-T}\Omega_N N^{-1})^T = N^{-T}\Omega_N^T N^{-1}$$

Consider the terms in Ω_N:

$$(\Omega_N)_{jk} = \int N_j''(t)N_k''(t)dt = \int N_k''(t)N_j''(t)dt = (\Omega_N)_{kj}$$

Thus, the matrix Ω_N is symmetric and

$$K^T = (N^{-T}\Omega_N N^{-1})^T = N^{-T}\Omega_N N^{-1} = K$$

This proves that K is symmetric and the matrix U is orthogonal. Returning to the expression for S_λ:

$$\begin{aligned} S_\lambda &= (I + \lambda K)^{-1} \\ &= (I + \lambda UDU^T)^{-1} \\ &= (UU^T + \lambda UDU^T)^{-1} \\ &= (U(I + \lambda D)U^T)^{-1} \\ &= U(I + \lambda D)^{-1}U^T \end{aligned}$$

This expression could be evaluated to efficiently compute S_λ for all λ while only needing to take a trivial matrix inverse.

Appendix

```r
## Load libraries and set working directory
setwd("/Users/bstan/Documents/UW/Courses/BIOST 527")
rm(list = ls())
library(tidyverse)
library(tidyr)
library(tinytex)
library(class)
library(MASS)
library(SemiPar)
library(splines)
set.seed(28)

######
# 1a
######
mu1 = 1
mu2 = 2
sigma = matrix(c(1,0.5,0.5,2), nrow = 2, ncol = 2)
lda_class1 = mvrnorm(n=600, mu=c(mu1,mu1), Sigma=sigma) # class 1 obs
lda_class2 = mvrnorm(n=400, mu=c(mu2,mu2), Sigma=sigma) # class 2 obs
lda_data = data.frame(rbind(lda_class1,lda_class2),class=as.factor(c(rep(1,600),rep(2,400))))
```

```r
colnames(lda_data) <- c("X1","X2","class")

ggplot() +
  geom_point(data=lda_data, aes(x=X1, y=X2, color=class),alpha=0.8,size=2) +
  labs(x='X1', y='X2') +
  ggtitle("Linear Discriminant Analysis Model Data") +
  theme_bw()

######
# 1b
######
## Create train and test sets
is_train = sample(c(TRUE,FALSE), 1000, replace=TRUE, prob = c(0.7,0.3))
## Fit LDA on train
lda.fit = lda(class~X1+X2,data=lda_data,subset=is_train)
#lda.fit
## Compute LDA training error
lda.pred.train = predict(lda.fit,lda_data[is_train,])
table_lda_train = table(lda.pred.train$class,as.numeric(lda_data[is_train,]$class))
print(1-(table_lda_train[1,1]+table_lda_train[2,2])/sum(table_lda_train))
## Compute LDA test error
lda.pred.test = predict(lda.fit,lda_data[!is_train,])
table_lda_test = table(lda.pred.test$class,as.numeric(lda_data[!is_train,]$class))
print(1-(table_lda_test[1,1]+table_lda_test[2,2])/sum(table_lda_test))

## Fit QDA on train
qda.fit = qda(class~X1+X2,data=lda_data,subset=is_train)
#qda.fit
## Compute QDA training error
qda.pred.train = predict(qda.fit,lda_data[is_train,])
table_qda_train = table(qda.pred.train$class,as.numeric(lda_data[is_train,]$class))
print(1-(table_qda_train[1,1]+table_qda_train[2,2])/sum(table_qda_train))
## Compute QDA test error
qda.pred.test = predict(qda.fit,lda_data[!is_train,])
table_qda_test = table(qda.pred.test$class,as.numeric(lda_data[!is_train,]$class))
print(1-(table_qda_test[1,1]+table_qda_test[2,2])/sum(table_qda_test))

######
# 1c
######
mu1 = 1.2
mu2 = 3
sigma1 = matrix(c(1,0.2,0.2,1), nrow = 2, ncol = 2)
sigma2 = matrix(c(4,2,2,6), nrow = 2, ncol = 2)
qda_class1 = mvrnorm(n=600, mu=c(mu1,mu1), Sigma=sigma1) # class 1 obs
qda_class2 = mvrnorm(n=400, mu=c(mu2,mu2), Sigma=sigma2) # class 2 obs
qda_data = data.frame(rbind(qda_class1,qda_class2),
                      class=as.factor(c(rep(1,600),rep(2,400))))
colnames(qda_data) <- c("X1","X2","class")

ggplot() +
  geom_point(data=qda_data, aes(x=X1, y=X2, color=class),alpha=0.8,size=2) +
  labs(x='X1', y='X2') +
  ggtitle("Quadratic Discriminant Analysis Model Data") +
  theme_bw()

######
# 1d
######
## Create train and test sets
is_train = sample(c(TRUE,FALSE), 1000, replace=TRUE, prob = c(0.7,0.3))
## Fit LDA on train
lda.fit = lda(class~X1+X2,data=qda_data,subset=is_train)
#lda.fit
## Compute LDA training error
lda.pred.train = predict(lda.fit,qda_data[is_train,])
table_lda_train = table(lda.pred.train$class,as.numeric(qda_data[is_train,]$class))
print(1-(table_lda_train[1,1]+table_lda_train[2,2])/sum(table_lda_train))
## Compute LDA test error
lda.pred.test = predict(lda.fit,qda_data[!is_train,])
table_lda_test = table(lda.pred.test$class,as.numeric(qda_data[!is_train,]$class))
print(1-(table_lda_test[1,1]+table_lda_test[2,2])/sum(table_lda_test))

## Fit QDA on train
qda.fit = qda(class~X1+X2,data=qda_data,subset=is_train)
#qda.fit
## Compute QDA training error
qda.pred.train = predict(qda.fit,qda_data[is_train,])
table_qda_train = table(qda.pred.train$class,as.numeric(qda_data[is_train,]$class))
print(1-(table_qda_train[1,1]+table_qda_train[2,2])/sum(table_qda_train))
## Compute QDA test error
qda.pred.test = predict(qda.fit,qda_data[!is_train,])
table_qda_test = table(qda.pred.test$class,as.numeric(qda_data[!is_train,]$class))
print(1-(table_qda_test[1,1]+table_qda_test[2,2])/sum(table_qda_test))

######
# 2a
######
data(lidar)
ggplot() +
  geom_point(data = lidar, aes(x=range, y=logratio)) +
  labs(x='range', y='logratio') +
  ggtitle("logratio v. range in LIDAR Data Set") +
  theme_bw()

######
# 2b
######
ggplot(data = lidar, aes(x=range, y=logratio)) +
  geom_point() +
  labs(x='range', y='logratio') +
  ggtitle("logratio v. range in LIDAR Data Set") +
  geom_smooth(aes(col="1"), method='lm', formula=y~x, se=FALSE) +
  geom_smooth(aes(col="2"), method='lm', formula=y~poly(x,2), se=FALSE) +
  geom_smooth(aes(col="6"), method='lm', formula=y~poly(x,6), se=FALSE) +
  geom_smooth(aes(col="8"), method='lm', formula=y~poly(x,8), se=FALSE) +
  scale_colour_manual("Degree", values = c("purple","red", "blue", "green")) +
  theme_bw() +
  theme(legend.position="top")

######
# 2c
######
set.seed(44)
train_obs = sample.int(n = nrow(lidar), size = floor(.5*nrow(lidar)), replace = F)
train_error = NULL
test_error = NULL
for (i in 1:12) {
  ## Fit model
  train_df = lidar[train_obs,]
  lm_fit = lm(logratio~poly(range,i),data=train_df)
  ## Calculate train error
  ls_train_error = mean(lm_fit$residuals^2)
  train_error = c(train_error,ls_train_error)
  ## Calculate test error
  test_df = lidar[-train_obs,]
  predicted_vals = predict(lm_fit,newdata=test_df)
  actuals_preds = data.frame(cbind(actual=test_df$logratio, predicted=predicted_vals))
  actuals_preds$error = actuals_preds$actual - actuals_preds$predicted
  ls_test_error = mean(actuals_preds$error^2) # MSE for test data
  test_error = c(test_error,ls_test_error)
}

ggplot() +
  geom_line(aes(x=1:12, y = train_error, colour = "train")) +
  geom_line(aes(x=1:12, y = test_error, colour = "test")) +
  labs(x='Degree of Polynomial', y='MSE', colour="Set") +
  ggtitle("Train and Test MSE v. Degree of Polynomial Fit") +
  theme_bw()

######
# 2d
######
ggplot(data = lidar, aes(x=range, y=logratio)) +
  geom_point() +
  labs(x='range', y='logratio') +
  ggtitle("logratio v. range in LIDAR Data Set") +
  geom_smooth(aes(col="04"), method='lm', formula=y~bs(x,df=3), se=FALSE) +
  geom_smooth(aes(col="06"), method='lm', formula=y~bs(x,df=5), se=FALSE) +
  geom_smooth(aes(col="08"), method='lm', formula=y~bs(x,df=7), se=FALSE) +
  geom_smooth(aes(col="12"), method='lm', formula=y~bs(x,df=11), se=FALSE) +
  scale_colour_manual("Degrees of Freedom", values = c("red", "blue", "green","purple")) +
  theme_bw() +
  theme(legend.position="top")

######
# 2e
######
train_error = NULL
```

```r
test_error = NULL
for (i in 3:11) {
  ## Fit model
  train_df = lidar[train_obs,]
  lm_fit = lm(logratio~bs(range,df=i),data=train_df)
  ## Calculate train error
  ls_train_error = mean(lm_fit$residuals^2)
  train_error = c(train_error,ls_train_error)
  ## Calculate test error
  test_df = lidar[-train_obs,]
  predicted_vals = predict(lm_fit,newdata=test_df)
  actuals_preds = data.frame(cbind(actual=test_df$logratio, predicted=predicted_vals))
  actuals_preds$error = actuals_preds$actual - actuals_preds$predicted
  ls_test_error = mean(actuals_preds$error^2) # MSE for test data
  test_error = c(test_error,ls_test_error)
}

ggplot() +
  geom_line(aes(x=4:12, y = train_error, colour = "train")) +
  geom_line(aes(x=4:12, y = test_error, colour = "test")) +
  labs(x='Degrees of Freedom of Cubic Spline', y='MSE', colour="Set") +
  ggtitle("Train and Test MSE v. Degrees of Freedom of Cubic Spline") +
  theme_bw()

######
# 5
######
## Define initial variables
rm(list = ls())
set.seed(23)
x = runif(50,0,1)

## Compute pointwise variance for linear model
x_lin = as.matrix(cbind(rep(1,50),x))
var_lin = diag(x_lin%*%solve(t(x_lin)%*%x_lin)%*%t(x_lin))
# Using loops
var_lin2 = NULL
for (i in 1:nrow(x_lin)) {
  var_i = x_lin[i,]%*%solve(t(x_lin)%*%x_lin)%*%x_lin[i,]
  var_lin2 = c(var_lin2,var_i)
}
comp_var = identical(var_lin,var_lin2) # TRUE

## Compute pointwise variance for polynomial with order 3
x_poly3 = as.matrix(cbind(rep(1,50),x,x^2,x^3))
var_poly3 = diag(x_poly3%*%solve(t(x_poly3)%*%x_poly3)%*%t(x_poly3))

## Compute pointwise variance for cubic spline with 2 knots
x_cubic = bs(x,degree=3,knots=c(0.33,0.66),intercept=TRUE)
var_cubic = diag(x_cubic%*%solve(t(x_cubic)%*%x_cubic)%*%t(x_cubic))

## Compute pointwise variance for natural spline with 6 knots
x_nat_cubic = ns(x,
                  df=6,
                  knots=c(0.26,0.42,0.58,0.74),
                  Boundary.knots = c(0.1,0.9),
                  intercept=TRUE)
var_nat_cubic = diag(x_nat_cubic%*%solve(t(x_nat_cubic)%*%x_nat_cubic)%*%t(x_nat_cubic))

## Plot results
ggplot() +
  geom_line(aes(x = x, y = var_lin, colour = "1. Global Linear")) +
  geom_point(aes(x = x, y = var_lin, colour = "1. Global Linear")) +
  geom_line(aes(x = x, y = var_poly3, colour = "2. Global Cubic Polynomial")) +
  geom_point(aes(x = x, y = var_poly3, colour = "2. Global Cubic Polynomial")) +
  geom_line(aes(x = x, y = var_cubic, colour = "3. Cubic Spline - 2 knots")) +
  geom_point(aes(x = x, y = var_cubic, colour = "3. Cubic Spline - 2 knots")) +
  geom_line(aes(x = x, y = var_nat_cubic, colour = "4. Natural Cubic Spline - 6 knots")) +
  geom_point(aes(x = x, y = var_nat_cubic, colour = "4. Natural Cubic Spline - 6 knots")) +
  labs(x='X', y='Pointwise Variances', colour="Model") +
  ggtitle("Recreated Figure 5.3 from ESL") +
  theme_bw() +
  theme(legend.position="top")
```