The Elements of Statistical Learning - Chapter 5 Exercises

Exercise 5.1

Show that the truncated power basis functions in (5.3) represent a basis for a cubic spline with the two knots as indicated.

Solution

Recall that

$$h_1(X) = 1, \qquad h_3(X) = X^2, \qquad h_5(X) = (X - \xi_1)^3$$
$$h_2(X) = X, \qquad h_4(X) = X^3, \qquad h_6(X) = (X - \xi_2)^3.$$

First note that each of these has continuity up to the second derivative so any linear combination of them does also. Moreover they are linearly independent: if $\sum_{m=1}^{6} \beta_m h_m(X) \equiv 0$ then $\sum_{m=1}^{4} \beta_m h_m(X) \equiv 0$ for $X < \xi_1$ so $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$, so $\beta_5 h_5(X) \equiv 0$ for $X < \xi_2$ implying $\beta_5 = 0$, so $\beta_6 h_6(X) \equiv 0$ and thus all $\beta_m = 0$.

It remains to show that the $h_m(X)$ span the space of cubic splines with knots ξ_1 and ξ_2. We establish this via the following claim.

Claim: Suppose that $f^{(1)}(X)$ and $f^{(2)}(X)$ are cubics and

$$f(X) = \begin{cases} f^{(1)}(X) & \text{if } X < \xi_1 \\ f^{(2)}(X) & \text{if } X \geq \xi_1 \end{cases}$$

is continuous up to its second derivative. Then there exists $\beta \in \mathbb{R}$ such that

$$f^{(2)}(X) \equiv f^{(1)}(X) + \beta(X - \xi)^3.$$

for $X \geq \xi$.

Spanning follows from this: if $f(X)$ is a cubic spline with $f(X) = f^{(1)}(X)$ for $X < \xi_1$, $f(X) = f^{(2)}(X)$ for $\xi_1 \leq X < \xi_2$, and $f(X) = f^{(3)}(X)$ for $\xi_2 \leq X <$ then $f^{(1)}$ can be written as a linear combination of $h_1, \ldots, h_4$ and then claim shows that adding in suitable multiples of h_5 and h_6 yields a function that equals $f^{(2)}$ and $f^{(3)}$ on their domains.

Proof of Claim: To establish this, write

$$f^{(i)}(X) = \sum_{m=0}^{3} \alpha_m^{(i)} (X - \xi)^m$$

for $i = 1, 2$. Then

$$\begin{aligned} f^{(1)}(\xi) = f^{(2)}(\xi) &\quad\Rightarrow\quad \alpha_0^{(1)} = \alpha_0^{(2)} \\ (f^{(1)})'(\xi) = (f^{(2)})'(\xi) &\quad\Rightarrow\quad \alpha_1^{(1)} = \alpha_1^{(2)} \\ (f^{(1)})''(\xi) = (f^{(2)})''(\xi) &\quad\Rightarrow\quad \alpha_2^{(1)} = \alpha_2^{(2)} \end{aligned}$$

so $\beta = \alpha_3^{(2)} - \alpha_3^{(1)}$ suffices.

Exercise 5.2

Suppose that $B_{i,M}(x)$ is an order-M B-spline defined in the Apppendix on page 186 through the sequence (5.77)–(5.78).

(a) Show by induction that $B_{i,M}(x) = 0$ for $x \notin [\tau_i, \tau_{i+M}]$. This shows, for example, that the support of cubic B-splines is at most 5 knots.

(b) Show by induction that $B_{i,M}(x) > 0$ for $x \in (\tau_i, \tau_{i+M})$. The B-splines are positive in the interior of their support.

(c) Show by induction that $\sum_{i=1}^{K+M} B_{i,M}(x) = 1 \; \forall x \in [\xi_0, \xi_{K+1}]$.

(d) Show that $B_{i,M}$ is a piecewise polynomial of order M (degree $M - 1$) on $[\xi_0, \xi_{K+1}]$, with breaks only at the knots $\xi_1, \ldots, \xi_K$.

(e) Show that an order-M B-spline basis function is the density function of a convolution of M uniform random variables.

Solution

(a) For $M = 1$ this is by definition. For the inductive step, suppose that $B_{i,M-1}$ and $B_{i+1,M-1}$ are zero outside $[\tau_i, \tau_{i+M-1}]$ and $[\tau_{i+1}, \tau_{i+M}]$, respectively. Since $B_{i,M}$ is a linear combination of these it is zero outside $[\tau_i, \tau_{i+M}]$ as required.

(b) It makes the induction slightly easier to prove that $B_{i,M} > 0$ for $x \in [\tau_i, \tau_{i+M})$. Again, for $M = 1$ this is by definition. For $M > 1$, $B_{i,M}$ is a linear combination of $B_{i,M-1}$ and $B_{i+1,M-1}$, their coefficients $(x - \tau_i)/(\tau_{i+M-1} - \tau_i)$ and $(\tau_{i+M} - x)/(\tau_{i+M} - \tau_{i+1})$ are non-zero on the supports $[\tau_i, \tau_{i+M-1})$ and $[\tau_{i+1}, \tau_{i+M-1})$, and finally $[\tau_i, \tau_{i+M-1}) \cup [\tau_{i+1}, \tau_{i+M-1}) = [\tau_i, \tau_{i+M})$. The claim follows.

(c) For $M = 1$ this is clear. If we assume that it's true for $M - 1$ then

$$\begin{aligned} \sum_{i=1}^{K+M} B_{i,M}(x) &= \sum_{i=1}^{K+M} \left(\frac{x - \tau_i}{\tau_{i+M-1} - \tau_i} B_{i,M-1}(x) + \frac{\tau_{i+M} - x}{\tau_{i+M} - \tau_{i+1}} B_{i+1,M-1}(x) \right) \\ &= \frac{x - \tau_1}{\tau_M - \tau_1} B_{1,M-1}(x) + \sum_{i=2}^{K+M} \left(\frac{x - \tau_i}{\tau_{i+M-1} - \tau_i} + \frac{\tau_{i+M-1} - x}{\tau_{i+M-1} - \tau_i} \right) B_{i+1,M-1}(x) + \frac{\tau_{K+2M} - x}{\tau_{K+2M} - \tau_{K+M+1}} B_{K+M+1,M-1}(x) \\ &= \frac{x - \tau_1}{\tau_M - \tau_1} B_{1,M-1}(x) + \sum_{i=2}^{K+M} B_{i+1,M-1}(x) + \frac{\tau_{K+2M} - x}{\tau_{K+2M} - \tau_{K+M+1}} B_{K+M+1,M-1}(x). \end{aligned}$$

By part (a), $B_{1,M-1}(x)$ is zero for $x < \tau_M \leq \xi_0$ and $B_{K+M+1,M-1}(x)$ is zero for $x > \tau_{K+M+1} \geq \xi_{K+1}$ so on $[\xi_0, \xi_{K+1}]$,

$$\sum_{i=1}^{K+M} B_{i,M}(x) = \sum_{i=2}^{K+M-1} B_{i,M-1}(x) = 1.$$

(d) We prove by induction that $B_{i,M}$ is a piecewise polynomial of order M with breaks only at $\tau_i, \ldots, \tau_{i+M}$. This claim clearly follows from this. For $M = 1$ this is clear. The inductive step is clear since $B_{i,M}(x)$ is of the form $(a_1 x + b_1) B_{i,M-1}(x) + (a_2 x + b_2) B_{i+1,M-1}(x)$.

(e) For the statement to be true, the differences $\delta = \tau_{i+1} - \tau_i$ must be independent of i (this is necessary for the simplest non-trivial case $M = 2$), so suppose this is the case. Then

$$B_{i,M}(x) = \frac{1}{(M-1)\delta} \left((x - \tau_i) B_{i,M-1}(x) + (\tau_i + M\delta - x) B_{i+1,M-1}(x) \right).$$

Morevoer, a simple induction shows that $B_{i+1,M}(x) = B_{i,M}(x - \delta)$.

Assume for the moment that $\tau_i = 0$ (we will remove this assumption at the end). Let X be a be a random variable with a uniform distribution on $[0, \delta)$. For $M \in \mathbb{N}$, let f_M denote the probability density function of $MX = \sum_{m=1}^{M} X$. We claim that $B_{i,M}(x) = \delta f_M(x)$ for all x. By the previous paragraph, it suffices to show that $B_{i,1}(x) = \delta f_1(x)$ and

$$f_M(x) = \frac{1}{(M-1)\delta} \left(x f_{M-1}(x) + (M\delta - x) f_{M-1}(x - \delta) \right)$$

for $M > 1$. The $M = 1$ claim is immediate since $f_1(x) = \frac{1}{\delta} I_{[0,\delta)}(x)$. We will prove the recursive formula using induction.

We apply strong induction to the following statement. For $M \in \mathbb{N}$, let f_M and F_M denote the pdf and cumulative density function (cdf) of MX. Then for $M > 1$,

$$\begin{aligned} f_M(x) &= \frac{1}{(M-1)\delta} \left(x f_{M-1}(x) + (M\delta - x) f_{M-1}(x - \delta) \right) \\ F_M(x) &= \frac{1}{M\delta} \left(x F_{M-1}(x) + (M\delta - x) F_{M-1}(x - \delta) \right) \end{aligned}$$

and for $M \geq 1$,

$$F_M(x) = \sum_{k=1}^{M} \left(\frac{1}{k} (x - (M-k)\delta) f_k(x - (M-k)\delta) \right) + I_{[M\delta,\infty)}(x).$$

For the base case, it suffices to show the summation formula for $F_1(x)$. We calculate directly:

$$\begin{aligned} F_1(x) &= \int f_1(x)\,dx \\ &= \frac{1}{\delta} \int I_{[0,\delta)}(x)\,dx \\ &= \frac{1}{\delta} x I_{[0,\delta)}(x) + I_{[\delta,\infty)}(x) \end{aligned}$$

which agrees with the formula.

For the inductive step, take $M > 1$ and assume that all applicable formulas hold for $1 \leq k \leq M$. We will prove them for $M + 1$.

First we claim that

$$\sum_{k=1}^{M-1} \frac{1}{k\delta} ((x - (M-k)\delta) f_k(x - (M-k)\delta) + ((M-k+1)\delta - x) f_k(x - (M-k+1)\delta)) = f_M(x - \delta) - f_1(x - M\delta).$$

Indeed, we can rewrite the summand as

$$\frac{1}{k\delta} ((x - (M-k)\delta) f_k(x - (M-k)\delta) + ((k+1)\delta - (x - (M-k)\delta) f_k(x - (M-k+1)\delta)) - f_k(x - (M-k+1)\delta).$$

Using the recursive formula for f_k, this equals

$$f_{k+1}(x - (M-k)\delta) - f_k(x - (M-k+1)\delta)$$

so the sum telescopes to leave $f_M(x - \delta) - f_1(x - M\delta)$.

Now we show the recursive formula for f_{M+1}. The pdf of a sum of random variables is equal to the convolution of the constituent pdfs so

$$\begin{aligned} f_{M+1}(x) &= (f_1 * f_M)(x) \\ &= \int_{-\infty}^{\infty} f_1(t) f_M(x - t)\,dt \\ &= \frac{1}{\delta} \int_{-\infty}^{\infty} I_{[0,\delta)}(t) f_M(x - t)\,dt \\ &= \frac{1}{\delta} \int_0^{\delta} f_M(x - t)\,dt \\ &= \frac{1}{\delta} \int_x^{x-\delta} f_M(s)\,dt \\ &= \frac{1}{\delta} (F_M(x) - F_M(x - \delta)) \end{aligned}$$

Using the summation formula for F_M, this gives

$$f_{M+1}(x) = \frac{1}{\delta} \sum_{k=1}^{M} \left[\frac{1}{k} ((x - (M-k)\delta) f_k(x - (M-k)\delta) + ((M-k+1)\delta - x) f_k(x - (M-k+1)\delta)) \right] + \frac{1}{\delta} (I_{[M\delta,\infty)}(x) - I_{[M\delta,\infty)}(x - \delta)).$$

The formula we proved at the start of the inductive step shows that this simplifies to

$$\frac{1}{M\delta} (x f_M(x) + (\delta - x) f_M(x - \delta)) + (f_M(x - \delta) - f_1(x - M\delta)) + \frac{1}{\delta} (I_{[M\delta,\infty)}(x) - I_{[M\delta,\infty)}(x - \delta)).$$

But $I_{[a,b)}(x) = I_{[a-\delta,b-\delta)}(x - \delta)$, so

$$\begin{aligned} \frac{1}{\delta} (I_{[M\delta,\infty)}(x) - I_{[M\delta,\infty)}(x - \delta)) &= \frac{1}{\delta} (I_{[0,\infty)}(x - M\delta) - I_{[\delta,\infty)}(x - M\delta)) \\ &= \frac{1}{\delta} I_{[0,\delta)}(x - M\delta) \\ &= f_1(x - M\delta) \end{aligned}$$

and the expression above simplifies to give

$$f_{M+1}(x) = \frac{1}{M\delta} (x f_M(x) + ((M+1)\delta - x) f_M(x - \delta)).$$

Now we move onto the recursive formula for $F_{M+1}(x)$. It suffices to differentiate the expression given and show that this equals $f_{M+1}(x)$. Indeed,

$$\begin{aligned} \frac{d}{dx} (x F_M(x) + ((M+1)\delta - x) F_M(x - \delta)) &= F_M(x) + x f_M(x) - F_M(x - \delta) + ((M+1)\delta - x) f_M(x - \delta) \\ &= F_M(x) - F_M(x - \delta) + M\delta f_{M+1}(x) \end{aligned}$$

using the formula we just proved. But in the course of the proof of the formula for f_{M+1} we also showed that $f_{M+1}(x) = \frac{1}{\delta}(F_M(x) - F_M(x - \delta)$. Thus the derivative above equals $(M+1)\delta f_{M+1}(x)$ and

$$F_{M+1}(x) = \frac{1}{(M+1)\delta} (x F_M(x) + ((M+1)\delta - x) F_M(x - \delta))$$

as required.

We will now use this formula to prove the summation formula for $F_{M+1}(x)$. Indeed, using the summation formulas for $F_M(x)$ and $F_M(x - \delta)$, we can expand it to

$$\frac{1}{(M+1)\delta} \sum_{k=1}^{M} \left[\frac{1}{k} \left(x(x - (M-k)\delta) f_k(x - (M-k)\delta) + ((M+1)\delta - x)(x - (M-k+1)\delta) f_k(x - (M-k+1)\delta) \right) \right] + \frac{1}{(M+1)\delta} \left(x I_{[M\delta,\infty)}(x) + ((M+1)\delta - x) I_{[M\delta,\infty)}(x - \delta) \right).$$

The second expression can be simplified:

$$\begin{aligned} \frac{1}{(M+1)\delta} \left(x I_{[M\delta,\infty)}(x) + ((M+1)\delta - x) I_{[M\delta,\infty)}(x - \delta) \right) &= \frac{1}{(M+1)\delta} \left(x I_{[M\delta,\infty)}(x) + ((M+1)\delta - x) I_{[(M+1)\delta,\infty)}(x) \right) \\ &= \frac{1}{(M+1)\delta} x I_{[M\delta,(M+1)\delta)}(x) + I_{[(M+1)\delta,\infty)}(x) \\ &= \frac{1}{M+1} x f_1(x - M\delta) + I_{[(M+1)\delta,\infty)}(x). \end{aligned}$$

Moreover, we can rearrange the summation to get

$$\frac{1}{M+1} \left[\sum_{k=1}^{M} \frac{x}{k\delta} \left((x - (M-k)\delta) f_k(x - (M-k)\delta) + ((k+1)\delta - (x - (M-k)\delta) f_k(x - (M-k+1)\delta) \right) + \frac{M-k+1}{k} (x - (M+1)\delta) f_k(x - (M-k+1)\delta) \right]$$

and using the recursive formula for f_k, this simplifies to

$$\frac{1}{M+1} \sum_{k=1}^{M} \left[x f_{k+1}(x - (M-k)\delta) + \frac{M-k+1}{k} (x - (M+1)\delta) f_k(x - (M-k+1)\delta). \right].$$

Thus

$$\begin{aligned} F_{M+1}(x) &= \frac{x}{M+1} f_{M+1}(x) + \sum_{k=2}^{M} \frac{1}{M+1} \left(x + \frac{M-k+1}{k} (x - (M+1)\delta) \right) f_k(x - (M-k+1)\delta) + I_{[(M+1)\delta,\infty)} \\ &= \sum_{k=1}^{M+1} \frac{1}{k} (x - (M-k+1)\delta) f_k(x - (M-k+1)\delta) + I_{[(M+1)\delta,\infty)} \end{aligned}$$

as required. This concludes the induction.

So we have shown that $B_{i,M}(x) = \delta f_M(x)$. It just remains the remove the assumption $\tau_i = 0$. For this it suffices to note that if X' is uniformly distributed on $[a, a + \delta)$ then $f_{X'}(x) = f_1(x - a)$. So if X_k is distributed uniformly on $[a_k, a_k + \delta)$ and $Y = \sum_{k=1}^{M} X_k$ then $f_Y(x) = f_M(x - \sum a_k)$. Thus if we choose the a_k such that $\sum a_k = \tau_i$, then $B_{i,M}(x) = f_Y(x)$.

Exercise 5.3

Write a program to reproduce Figure 5.3 on page 145.

Solution

In each of the models - global linear, global cubic, cubic spline, and natural cubic spline - we are assuming a model $Y = f(X) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(X) = h(X)^T \theta$ for some *basis function* $h(X) = (h_1(X), \ldots, h_M(X))$. Let $\mathbf{X}$ denote the $N \times 1$ input data matrix ($N = 50$) and for a given model, let $\mathbf{H}$ denote the $N \times M$ *basis matrix* with ith row $h(x_i)$. Then, as in Section 3.2, the least squares estimate $\hat{\theta}$ for θ is distributed normally with mean θ and variance $(\mathbf{H}^T\mathbf{H})^{-1}\sigma^2$. Thus, $\hat{f}(X) = h(X)^T \hat{\theta}$ has pointwise variance

$$v(X) = h(X)^T (\mathbf{H}^T\mathbf{H})^{-1} h(X) \sigma^2.$$

In the script below we generate basis functions $h(X)$ and pointwise variance functions $v(X)$ and plot $v(\mathbf{X})$ against $\mathbf{X}$ for each model. Since σ^2 just acts as a scaling factor, we set $\sigma^2 = 1$ without loss of generality.

In [1]:
```
# pydata stack
import numpy as np
from matplotlib import pyplot as plt

# Custom module for cubic splines
from cubspl import gen_nat_cubic_spl_basis_fun, gen_cubic_spl_basis_fun
```

We begin with a function to generate the pointwise variance function $v(X)$ from a basis function.

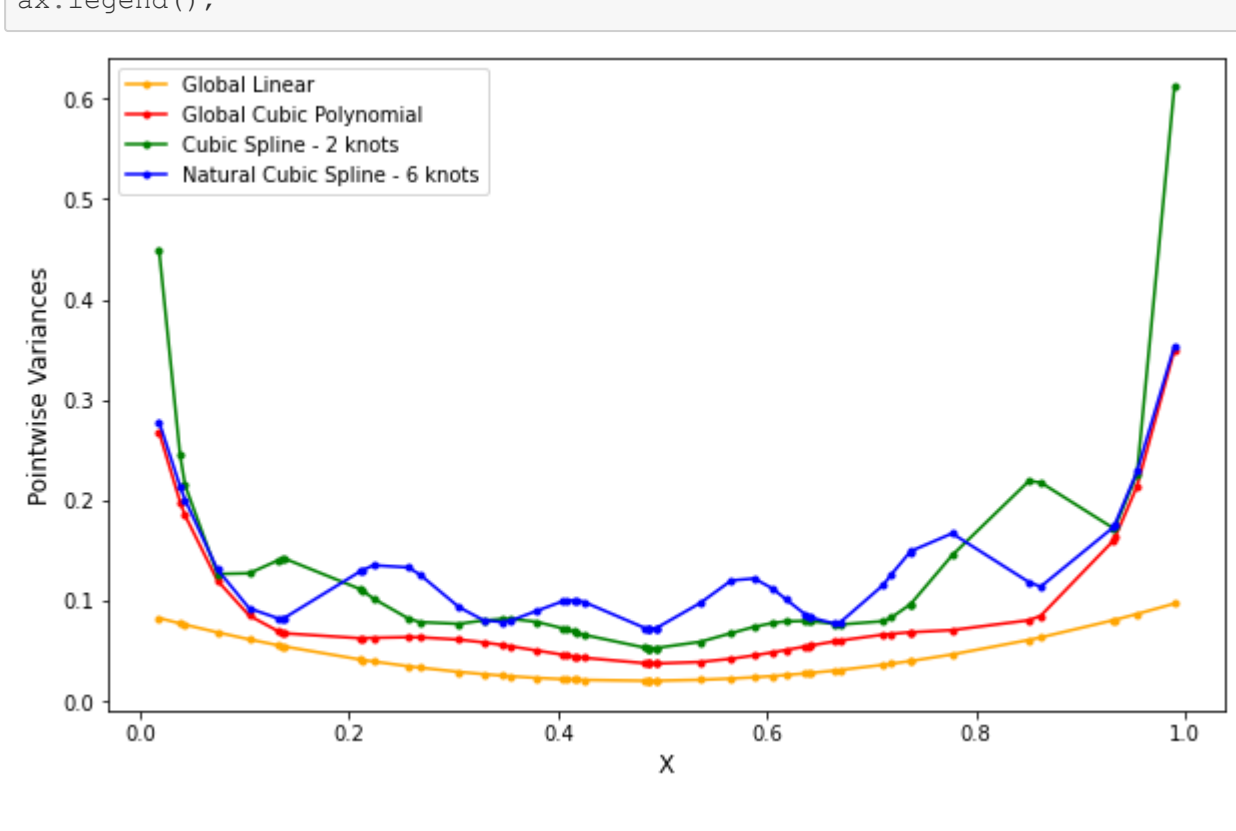
In [2]:
```
# Function for pointwise variance based on input data and basis function
def gen_pointwise_variance(X, basis_fun):
    basis_matrix = np.array([basis_fun(x) for x in X])

    def pointwise_variance(x):
        h = basis_fun(x)

        return h @ np.linalg.inv(basis_matrix.T @ basis_matrix) @ h

    return pointwise_variance
```

We we define basis functions for each model. For splines we use functions from a custom module `cubspl`.

In [3]:
```
# Basis function for global linear model
def lin_basis_fun(x):
    return np.array([1, x])

# Basis function for global cubic model
def cubic_basis_fun(x):
    return np.array([1, x, x**2, x**3])

# Basis function for cubic spline
cubic_knots = [0.33, 0.66]
cubic_spl_basis_fun = gen_cubic_spl_basis_fun(knots=cubic_knots)

# Basis function for natural cubic spline
nat_cubic_knots = np.linspace(0.1, 0.9, 6)
nat_cubic_spl_basis_fun = gen_nat_cubic_spl_basis_fun(knots=nat_cubic_knots)
```

Now we generate variance functions for each model based on simulated data and plot the results.

In [7]:
```
# Sample 50 points uniformly on [0, 1]
N = 50
X = np.random.uniform(low=0, high=1, size=(N,))
X = np.sort(X)

# Generate variance functions for each model
lin_var = gen_pointwise_variance(X, lin_basis_fun)
cubic_var = gen_pointwise_variance(X, cubic_basis_fun)
cubic_spl_var = gen_pointwise_variance(X, cubic_spl_basis_fun)
nat_cubic_spl_var = gen_pointwise_variance(X, nat_cubic_spl_basis_fun)

# Initialise plot
fig, ax = plt.subplots(figsize=(10, 6))

# Plot variance against X for each model
ax.plot(X, [lin_var(x) for x in X], marker='.', color='orange', label='Global Linear')
ax.plot(X, [cubic_var(x) for x in X], marker='.', color='red', label='Global Cubic Polynomial')
ax.plot(X, [cubic_spl_var(x) for x in X], marker='.', color='green', label='Cubic Spline - 2 knots')
ax.plot(X, [nat_cubic_spl_var(x) for x in X], marker='.', color='blue', label='Natural Cubic Spline - 6
 knots')

# Axis labels
ax.set_xlabel('X', fontsize=12)
ax.set_ylabel('Pointwise Variances', fontsize=12)

ax.legend();
```



Exercise 5.4

Consider the truncated power series representation for cubic splines with K interior knots. Let

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

Prove that the natural boundary conditions for natural cubic splines (Section 5.2.1) imply the following linear constraints on the coefficients:

$$\beta_2 = 0, \qquad \sum_{k=1}^{K} \theta_k = 0,$$
$$\beta_3 = 0, \qquad \sum_{k=1}^{K} \xi_k \theta_k = 0.$$

Hence derive the basis (5.4) and (5.5).

Solution

The condition that $f(X)$ be linear for $X < \xi_1$ clearly implies that $\beta_2 = \beta_3 = 0$. If this is the case then for $X > \xi_K$

$$f(X) = \left(\sum_{k=1}^{K} \theta_k \right) X^3 + 3 \left(\sum_{k=1}^{K} \xi_k \theta_k \right) X^2 + \text{lower order terms}.$$

So for $f(X)$ to be linear for $X > \xi_K$ we need $\sum \theta_k = \sum \xi_k \theta_k = 0$.

Now let

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

where

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}.$$

First note that all of these are natural cubic splines: $N_{k+2}(X)$ has

$$\sum_{k=1}^{K} \theta_k = \frac{1}{\xi_K - \xi_k} - \frac{1}{\xi_K - \xi_{K-1}} + \left(\frac{1}{\xi_K - \xi_{K-1}} - \frac{1}{\xi_K - \xi_k} \right) = 0$$

and

$$\begin{aligned} \sum_{k=1}^{K} \xi_k \theta_k &= \frac{\xi_k}{\xi_K - \xi_k} - \frac{\xi_{K-1}}{\xi_K - \xi_{K-1}} + \left(\frac{\xi_K}{\xi_K - \xi_{K-1}} - \frac{\xi_K}{\xi_K - \xi_k} \right) \\ &= \frac{1}{(\xi_K - \xi_k)(\xi_K - \xi_{K-1})} \left(\xi_k(\xi_K - \xi_{K-1}) - (\xi_K - \xi_k)\xi_{K-1} + \xi_K((\xi_K - \xi_k) - (\xi_K - \xi_{K-1})) \right) \\ &= 0. \end{aligned}$$

Moreover, the $N_k(x)$ are clearly linearly independent for $k \leq K$ since they are linear combinations of the truncated power series basis and $N_{k+2}(x)$ is the only term for which $(X - \xi_k)_+^3$ has a non-zero coefficient.

Finally take a natural cubic spline $f(X)$ as above. We claim that

$$f(X) = \beta_1 N_1(X) + \beta_2 N_2(X) + \sum_{k=1}^{K-2} (\xi_K - \xi_k) \theta_k N_{k+2}(X).$$

It suffices to show that the coefficients of $(X - \xi_{K-1})_+^3$ and $(X - \xi_K)_+^3$ match up. The coefficient of $(X - \xi_{K-1})_+^3$ in the above expression is

$$\begin{aligned} -\sum_{k=1}^{K-2} \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} \theta_k &= \frac{1}{\xi_K - \xi_{K-1}} \left(-\xi_K \sum_{k=1}^{K-2} \theta_k + \sum_{k=1}^{K-2} \xi_k \theta_k \right) \\ &= \frac{1}{\xi_K - \xi_{K-1}} \left(\xi_K(\theta_{K-1} + \theta_K) - (\xi_{K-1}\theta_K + \xi_K \theta_K) \right) \\ &= \theta_{K-1} \end{aligned}$$

and the coefficient of $(X - \xi_K)_+^3$ is

$$\sum_{k=1}^{K-2} (\xi_K - \xi_k) \left(\frac{1}{\xi_K - \xi_{K-1}} - \frac{1}{\xi_K - \xi_k} \right) \theta_k = \theta_{K-1} - \sum_{k=1}^{K-2} \theta_k = \theta_K.$$

Thus the $N_k(x)$ for $1 \leq k \leq K$ form a basis for the space of natural cubic splines.

Exercise 5.5

Write a program to classify the phoneme data using a quadratic discriminant analysis (Section 4.3). Since there are many correlated features, you should filter them using a smooth basis of natural cubic splines (Section 5.2.3). Decide beforehand on a series of five different choices for the number and position of the knots, and use tenfold cross-validation to make the final selection. The phoneme data are available from the book website https://web.stanford.edu/~hastie/ElemStatLearn/.

Solution

Data Processing

In [43]:
```
# Pydata stack
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

# Custom module for cubic splines
from cubspl import gen_nat_cubic_spl_basis_fun

# QDA model
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

# Metrics
from sklearn.metrics import accuracy_score

# To apply data filter
from sklearn.preprocessing import FunctionTransformer

# Pipelines
from sklearn.pipeline import Pipeline

# For cross-validation
from sklearn.model_selection import GridSearchCV
```

We begin by importing and taking a look at the phoneme data set.

In [9]:
```
# Import data
df = pd.read_csv('phoneme-data.csv', skipinitialspace=True, header=0, index_col=0)

# Look at data
df.head()
```

Out[9]:

	x.1	x.2	x.3	x.4	x.5	x.6	x.7	x.8	x.9	x.10	...	x.249	x.250
row.names													
1	9.85770	9.20711	9.81689	9.01692	9.05675	8.92518	11.28308	11.52980	10.79713	9.04747	...	12.68076	11.20767
2	13.23079	14.19189	15.34428	18.11737	19.53875	18.32726	17.34169	17.16861	19.63557	20.15212	...	8.45714	8.77266
3	10.81889	9.07615	9.77940	12.20135	12.59005	10.53364	8.54693	9.46049	11.96755	12.05282	...	5.00824	5.51019
4	10.53679	9.12147	10.84621	13.92331	13.52476	10.27831	8.97459	11.57109	12.35839	10.47826	...	5.85688	5.40324
5	12.96705	13.69454	14.91182	18.22292	18.45390	17.25760	17.79614	17.76387	18.99632	17.40394	...	8.00151	7.58624

5 rows × 258 columns

The data consists of 4509 observations of log-periodograms of different speech phonemes.

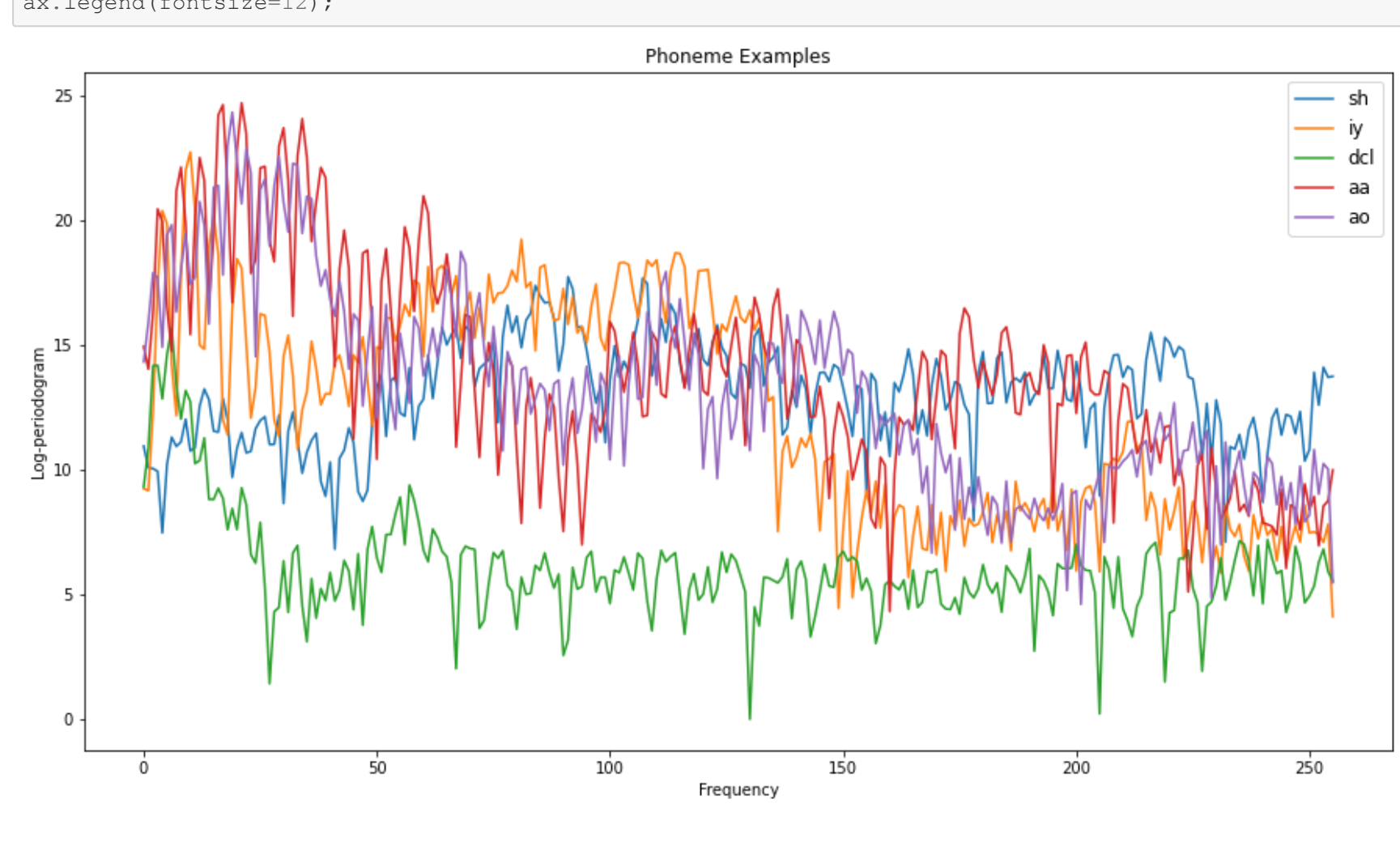
Inputs: `x.1,..,x.256` continuous numeric. Together they form the log-periodogram for frequencies from 1 to 256.

Output: `g` - phoneme, categorical. Values: `aa`, `ao`, `dcl`, `iy`, `sh`.

We also have a column `speaker` which identifies the speaker with a string consisting of four parts separated by periods, e.g. `train.dr1.mcpm0.sa1`. The first part is either `test` or `train`, the second is one of `dr1`,..., `dr8`, the third part is a 4 letter code followed by a number, and the final part is always `sa1`.

We will ignore the speaker information except to split the data into a training and test set.

In [10]:
```
# Plot some sample periodograms
fig, ax = plt.subplots(figsize=(15, 8))

# Sample a log-periodograms for each phoneme
N = len(df)
phonemes = df.g.unique()

for phoneme in phonemes:
    sample = df[df.g == phoneme].sample(n=1)
    ax.plot(sample.iloc[0, :-2], label=phoneme)

# Plot parameters
ax.set_xticks(range(0, 256, 50))
ax.set_xticklabels(range(0, 256, 50))

ax.set_xlabel('Frequency')
ax.set_ylabel('Log-periodogram')

ax.set_title('Phoneme Examples')

ax.legend(fontsize=12);
```



We split the data into training and test sets and separate into inputs and outputs.

In [11]:
```
# Split into training and test
df['set'] = df.speaker.str.split('.', expand=True)[0]
train = df[df.set == 'train']
test = df[df.set == 'test']
df = df.drop(columns=['set'])

# Store useful constants
N = len(df)
N_train = len(train)
N_test = len(test)
p = 256

# Split into inputs and outputs
X_train = train.iloc[:, :p].to_numpy()
y_train = train.loc[:, 'g'].to_numpy()

X_test = test.iloc[:, :p].to_numpy()
y_test = test.loc[:, 'g'].to_numpy()
```

Model Specification

We will use two models: 'raw' quadratic discriminant analysis (QDA) on the features $x_1, \ldots, x_{256}$ and regularised QDA, implemented by filtering the features using a basis of natural cubic splines. In this section we fit both models and use 10-fold cross-validation to choose the position of the knots for the regularised QDA.

We begin by fitting the raw QDA model.

In [91]:
```
# Fit raw QDA
qda_raw = QDA()
qda_raw.fit(X_train, y_train);

# Calculate raw training error
y_train_est_raw = qda_raw.predict(X_train)
qda_raw_train_err = 1 - accuracy_score(y_train, y_train_est_raw)
print('Raw QDA training misclassification rate: {:.3f}'.format(qda_raw_train_err))
```

```
Raw QDA training misclassification rate: 0.000
```

Note that the misclassification rate is zero which suggests that we have severly overfit the training data.

To fit the regularised model we define functions to generate the basis matrix $\mathbf{H}$ and the filtering function $\mathbf{X} \to \mathbf{XH}$ given a set of knots.

In [22]:
```
def gen_basis_matrix(knots, p=256):
    basis_fun = gen_nat_cubic_spl_basis_fun(knots=knots)
    basis_matrix = np.array([basis_fun(i) for i in range(1, p+1)])

    return basis_matrix
```

In [21]:
```
def gen_filter(basis_matrix):
    def data_filter(X):
        if X.ndim == 1:
            X = X[np.newaxis, :]

        return X @ basis_matrix

    return data_filter
```

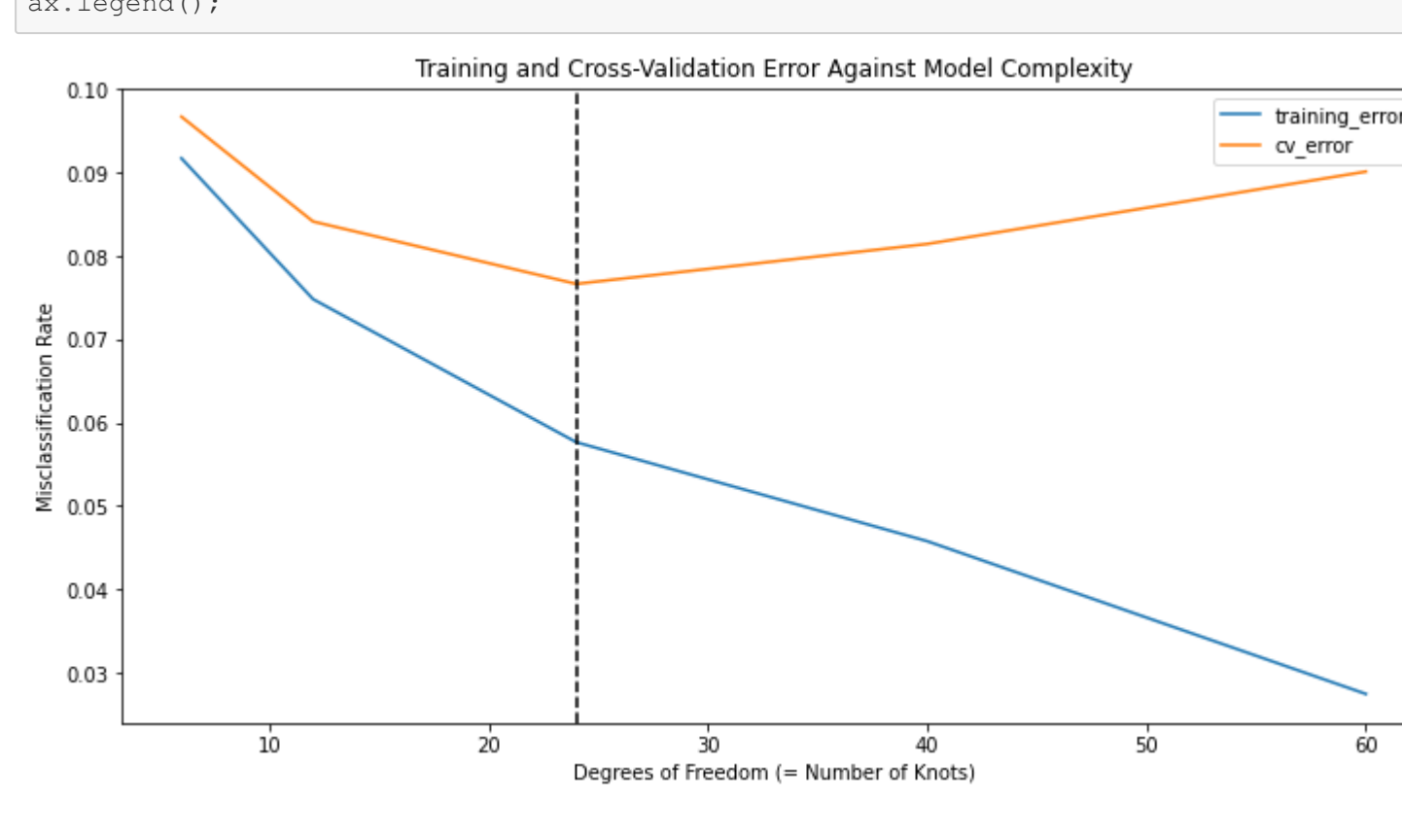
Now we specify 5 sets of positions for the knots, fit a filtered QDA model for each, and examine the training and test error to choose the best configuration.

In [84]:
```
# Different positions to try for knots
num_knots = [6, 12, 24, 40, 60]
knot_positions = [np.linspace(1, 256, k, endpoint=False) + 255 / (2*k) for k in num_knots]

# Generate filter function for each set of knots
basis_matrices = [gen_basis_matrix(knots) for knots in knot_positions]
data_filters = [gen_filter(matrix) for matrix in basis_matrices]

# Create pipeline
estimators = [('filter', FunctionTransformer()),
              ('qda', QDA())]
pipe = Pipeline(estimators)

# Create GridSearch for CV
param_grid = {'filter__func' : data_filters}
qda_reg = GridSearchCV(pipe,
                       param_grid,
                       scoring='accuracy',
                       cv=10,
                       return_train_score=True,
                       refit=True)

# Fit to data
qda_reg.fit(X_train, y_train);
```

Out[84]:
```
GridSearchCV(cv=10,
             estimator=Pipeline(steps=[('filter', FunctionTransformer()),
                                       ('qda',
                                        QuadraticDiscriminantAnalysis())]),
             param_grid={'filter__func': [<function gen_filter.<locals>.data_filter at 0x12f77d320>,
                                          <function gen_filter.<locals>.data_filter at 0x12f77d0e0>,
                                          <function gen_filter.<locals>.data_filter at 0x12f77d950>,
                                          <function gen_filter.<locals>.data_filter at 0x12f77da70>,
                                          <function gen_filter.<locals>.data_filter at 0x12f77d5f0
>]},
             return_train_score=True, scoring='accuracy')
```

Compare the training and test error for different knot positions.

In [92]:
```
# Initialise figure
fig, ax = plt.subplots(figsize=(12, 6))

# Plot errors
ax.plot(num_knots, 1-qda_reg.cv_results_['mean_train_score'], label='training_error')
ax.plot(num_knots, 1-qda_reg.cv_results_['mean_test_score'], label='cv_error')

# Plot value that minimises cv error
ax.axvline(num_knots[qda_reg.best_index_], color='k', ls='--')

# Fig parameters
ax.set_title('Training and Cross-Validation Error Against Model Complexity')
ax.set_xlabel('Degrees of Freedom (= Number of Knots)')
ax.set_ylabel('Misclassification Rate')

ax.legend();
```



Cross-validation suggests that 24 equally spaced knots on $[1, 256]$ is the best out of the options tried.

Testing

Now we compare misclassification errors of the 'raw' QDA and the regularised one.

In [89]:
```
# Calculate raw training error
y_train_est_raw = qda_raw.predict(X_train)
qda_raw_train_err = 1 - accuracy_score(y_train, y_train_est_raw)
print('Raw QDA training misclassification rate: {:.3f}'.format(qda_raw_train_err))

# Calculate regularised training error
y_train_est_reg = qda_reg.predict(X_train)
qda_reg_train_err = 1 - accuracy_score(y_train, y_train_est_reg)
print('Regularised QDA training misclassification rate: {:.3f} \n'.format(qda_reg_train_err))

# Calculate raw test error
y_test_est_raw = qda_raw.predict(X_test)
qda_raw_test_err = 1 - accuracy_score(y_test, y_test_est_raw)
print('Raw QDA test misclassification rate: {:.3f}'.format(qda_raw_test_err))

# Calculate regularised test error
y_test_est_reg = qda_reg.predict(X_test)
qda_reg_test_err = 1 - accuracy_score(y_test, y_test_est_reg)
print('Regularised QDA test misclassification rate: {:.3f}'.format(qda_reg_test_err))
```

```
Raw QDA training misclassification rate: 0.000
Regularised QDA training misclassification rate: 0.059

Raw QDA test misclassification rate: 0.158
Regularised QDA test misclassification rate: 0.074
```

Smoothing enabled us to halve the misclassification rate.

Exercise 5.6

Suppose you wish to fit a periodic function, with a known period T. Describe how you could modify the truncated power series basis to achieve this goal.

Solution

It is a little unclear what this question is asking for but I will give my interpretation. Suppose that $f(X)$ is periodic with period T and we wish to approximate f using splines. One approach is to take knots $0 < \xi_1 < \cdots < \xi_K < T$, approximate f on the interval $(0, T)$ using a spline $\hat f$ of order M, and extend this to $\mathbf{R}$ by $\hat f(X - kT) = \hat f(X)$ for $X \in [kT, (k+1)T)$. To get smoothness at the 'joins' $\{kT\}$ we need $\hat f(X) \equiv \hat f(X - T)$ for $X \in (\xi_K, \xi_1 + T)$. So we will find a basis for the space of order M splines with the given knot sequence satisfying this condition.

First suppose that $p(X)$ is a polynomial of degree $M - 2$. We claim that for each $c \in \mathbf{R}$ there is a unique polynomial $q(X)$ of degree $M - 1$ such that $q(X) - q(X - T) \equiv p(X)$ and $q(0) = c$. Indeed, there is a unique degree $M - 1$ polynomial q such that

$$q(kT) = c + \sum_{j=1}^{k} p(jT)$$

for $0 \le k \le M - 1$ and the difference $q(X) - q(X - T)$ is a degree $M - 2$ polynomial which evaluates to $p(kT)$ at $X = kT$ for $1 \le k \le M - 1$, so $q(X) - q(X - T) = p(X)$.

Before continuing we will just mention that such a polynomial $q(X)$ can easily be constructed from $p(X)$. For example, if Δ denotes the forward difference operator $\Delta p(X) = p(X + T) - p(X)$ then the Newton interpolating polynomial

$$q(X) = \sum_{k=1}^{M-1} \binom{s}{k} \Delta^{k-1} p(X)$$

will do the job, where $s = \frac{X}{T}$.

Now recall the basis for the space of order M splines at knots $\xi_1, \dots, \xi_K$:

$$h_j(X) = X^{j-1}, \quad j = 1, \dots, M$$
$$h_{M+l}(X) = (X - \xi_l)_+^{M-1}, \quad l = 1, \dots, K.$$

For $1 \le l \le K$, let $R_l(X)$ be a degree $M - 1$ polynomial such that $R_l(0) = 0$ and

$$R_l(X) - R_l(X - T) \equiv X^{M-1} - (X - \xi_l)^{M-1},$$

and set

$$Q_l(X) = (X - \xi_l)_+^{M-1} + R_l(X) = h_{M+l}(X) + R_l(X).$$

Now define $P_0(X) = 1$ and

$$P_l(X) = Q_l(X) - Q_K(X)$$

for $1 \le l \le K - 1$. We claim that $P_0, \dots, P_{K-1}$ form a basis for the space of order M splines at knots $\xi_1, \dots, \xi_K$ with $\hat f(X) \equiv \hat f(X - T)$ for $X \in (\xi_K, \xi_1 + T)$.

Take $1 \le l \le K - 1$. Then $P_l(X)$ is clearly a spline at the given knots. Moreover if $X \in (\xi_K, \xi_1 + T)$ then, by definition,

$$\begin{aligned} R_l(X) - R_l(X - T) &= (X - \xi_l)^{M-1} - X^{M-1} \\ \Rightarrow R_l(X - T) - R_K(X - T) &= R_l(X) + (X - \xi_l)^{M-1} - R_K(X) - (X - \xi_K)^{M-1} \\ \Rightarrow P_l(X - T) &= P_l(X) \end{aligned}$$

This condition is clearly also true for $P_0(X)$. It is also immediate that the $P_l(X)$ are linearly independent since the $h_l(X)$ are linearly independent and $P_l(X)$ contains the term $h_{M+l}(X)$ which doesn't appear in any other $P_k(X)$.

So it remains to show spanning. Take an order M spline $f(X)$ on knots $\xi_1, \dots, \xi_K$. This has a unique representation in the form

$$f(X) = g(X) + \sum_{l=1}^{K} \theta_l (X - \xi_l)_+^{M-1},$$

where $g(X)$ is a polynomial of degree $M - 1$. Suppose that $f(X - T) \equiv f(X)$ on $(\xi_K, \xi_1 + T)$. Using the representation above, this implies that

$$g(X - T) = g(X) + \sum_{l=1}^{K} \theta_l (X - \xi_l)^{M-1}.$$

We immediately note that equating the coefficients of X^{M-1} implies that $\sum_{l=1}^K \theta_l = 0$, so $\theta_K = -\sum_{l=1}^{K-1} \theta_l$.

Let

$$\begin{aligned} \tilde f(X) &= g(0)P_0(X) + \sum_{l=1}^{K-1} \theta_l P_l(X) \\ &= g(0) + \sum_{l=1}^{K} \theta_l R_l(X) + \sum_{l=1}^{K} \theta_l (X - \xi_l)_+^{M-1}. \end{aligned}$$

We claim that $f = \tilde f$. It suffices to show that the polynomial terms are equal:

$$g(X) = g(0) + \sum_{l=1}^{K} \theta_l R_l(X).$$

But both polynomials evaluate to $g(0)$ at $X = 0$ and their 'backward differences' agree:

$$\begin{aligned} \left(g(0) + \sum_{l=1}^{K} \theta_l R_l(X)\right) - \left(g(0) + \sum_{l=1}^{K} \theta_l R_l(X - T)\right) &= \sum_{l=1}^{K} \theta_l (X - \xi_l)^{M-1} - \sum_{l=1}^{K} \theta_l X^{M-1} \\ &= \sum_{l=1}^{K} \theta_l (X - \xi_l)^{M-1} \\ &= g(X - T) - g(X) \end{aligned}$$

so the polynomials must be equal by the uniqueness argument at the beginning.

Therefore the set $\{P_0, \dots, P_{K-1}\}$ is a basis and the problem is solved.

Exercise 5.7

Derivation of smoothing splines (Green and Silverman, 1994). Suppose that $N \ge 2$, and that g is the natural cubic spline interpolant to the pairs $\{x_i, z_i\}_1^N$, with $a < x_1 < \cdots < x_N < b$. This is a natural spline with a knot at every x_i; being an N-dimensional space of functions, we can determine the coefficients such that it interpolates the sequence z_i exactly. Let $\tilde g$ be any other differentiable function on $[a, b]$ that interpolates the N pairs.

(a) Let $h(x) = \tilde g(x) - g(x)$. Use integration by parts and the fact that g is a natural cubic spline to show that

$$\begin{aligned} \int_a^b g''(x)h''(x)\,dx &= -\sum_{j=1}^{N-1} g'''(x_j^+)\{h(x_{j+1}) - h(x_j)\} \\ &= 0. \end{aligned}$$

(b) Hence show that

$$\int_a^b \tilde g''(t)^2\,dt \ge \int_a^b g''(t)^2\,dt,$$

and that equality can only hold if h is identically zero on $[a, b]$.

(c) Consider the penalized least squares problem

$$\min_f \left[\sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int_a^b f''(t)^2\,dt\right].$$

Use (b) to argue that the minimizer must be a cubic spline with knots at each of the x_i.

Solution

(a) Write $x_0 = a$ and $x_{N+1} = b$ for notational convenience. By integration by parts,

$$\begin{aligned} \int_a^b g''(x)h''(x)\,dx &= \sum_{j=0}^{N} \int_{x_j}^{x_{j+1}} g''(x)h''(x)\,dx \\ &= \sum_{j=0}^{N} \left([g''(x)h'(x)]_{x_j}^{x_{j+1}} - \int_{x_j}^{x_{j+1}} g'''(x)h'(x)\,dx\right) \\ &= (g''(b)h'(b) - g''(a)h'(a)) - \sum_{j=0}^{N} \left(\int_{x_j}^{x_{j+1}} g'''(x)h'(x)\,dx\right). \end{aligned}$$

Since $g(x)$ is a cubic spline, $g'''(x)$ is piecewise constant and is equal to $g'''(x_j^+)$ for all $x \in (x_j, x_{j+1})$, where $x_j^+ = x_j + \varepsilon$ for some suitably small ε. Moreover, since $g(x)$ is natural, $g''(a) = g''(b) = g'''(x_0^+) = g'''(x_N^+) = 0$ and thus

$$\int_a^b g''(x)h''(x)\,dx = -\sum_{j=1}^{N-1} g'''(x_j^+)[h(x)]_{x_j}^{x_{j+1}} = -\sum_{j=1}^{N-1} g'''(x_j^+)\left(h(x_{j+1}) - h(x_j)\right) = 0,$$

as $h(x_j) = \tilde g(x_j) - g(x_j) = 0$.

(b) Part (a) implies that

$$\begin{aligned} 0 &= \int_a^b g''(t)h''(t)\,dt = \int_a^b g''(t)\tilde g''(t)\,dt - \int_a^b g''(t)^2\,dt \\ &\Rightarrow \int_a^b g''(t)\tilde g''(t)\,dt = \int_a^b g''(t)^2\,dt. \end{aligned}$$

Moreover,

$$\begin{aligned} 0 \le \int_a^b h''(t)^2\,dt &= \int_a^b \tilde g''(t)h''(t)\,dt - \int_a^b g''(t)h''(t)\,dt \\ &= \int_a^b \tilde g''(t)h''(t)\,dt \\ &= \int_a^b \tilde g''(t)^2\,dt - \int_a^b \tilde g''(t)g''(t)\,dt \end{aligned}$$

with equality if and only if $h \equiv 0$ on $[a, b]$. Combining these two pieces yields the result.

(c) Let $RSS(f)$ denote the expression to be minimized and let f be a minimizer. Define $z_i = f(x_i)$ for all i and let g denote the unique natural cubic spline with $g(x_i) = z_i$ for all i. By minimality of f, $RSS(f) \le RSS(g)$. But by part (b)

$$\int_a^b f''(t)^2\,dt \ge \int_a^b g''(t)^2\,dt,$$

so $RSS(g) \le RSS(f)$, as $f(x_i) = g(x_i)$ for all i. So $RSS(g) = RSS(f)$ and the inequality above is an equality. Part (b) now implies that $f \equiv g$ on $[a, b]$ and so f is a natural cubic spline with knots at the x_i.

Exercise 5.8

In the appendix to this chapter we show how the smoothing spline computations could be more efficiently carried out using a $(N + 4)$ dimensional basis of B-splines. Describe a slightly simpler scheme using a $(N + 2)$-dimensional B-spline basis defined on the $N - 2$ interior knots.

Solution

By Exercise 5.7, if we replace the integral in the smoothing minimisation problem with a finite integral $\int_a^b f''(t)^2\,dt$ where $a \le x_1$ and $b \ge x_N$, the minimiser is still a natural cubic spline and thus is independent of the choice of $a \in [-\infty, x_1]$, $b \in [x_N, \infty]$. Let $\hat f_\lambda$ denote this unique solution. Its second derivative will be zero outside of $[x_1, x_N]$.

Recall that the definition of the B-splines requires additional exterior knots $\tau_1 \le \cdots \le \tau_4 \le x_0 < x_1$ and $x_N < x_{N+1} \le \tau_1' \le \cdots \le \tau_4'$. The cubic B-splines on these knots form a basis for the set of cubic splines on $[x_0, x_{N+1}]$. So if $\hat f$ is the minimiser of

$$\sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int_{x_0}^{x_{N+1}} f''(t)^2\,dt$$

among functions of the form $\sum_{j=1}^{N+4} \gamma_j N_j(x)$, then $\hat f \equiv \hat f_B$ on $[x_0, x_{N+1}]$.

Consider restricting to $B_2(x), \dots, B_{N+3}(x)$. These are B-splines for the knot sequence with initial exterior knots τ_2, τ_3, x_0, x_1, interior knots $x_2, \dots, x_{N-1}$, and final exterior knots $x_N, x_{N+1}, \tau_2', \tau_3'$. Thus they form a basis for the set of cubic splines on $[x_1, x_N]$ and so if $\tilde f_B$ is the minimiser of

$$\sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int_{x_1}^{x_N} f''(t)^2\,dt$$

among functions of the form $\sum_{j=2}^{N+3} \gamma_j N_j(x)$, then $\hat f \equiv \tilde f_B$ on $[x_1, x_N]$.

It just remains to observe that a natural cubic spline with knots $x_1, \dots, x_N$ is uniquely determined by its restriction to $[x_1, x_N]$, so we can recover the whole of $\hat f$ from $\tilde f_B$. Thus we can find the smoothing spline by doing the usual computation with $\mathbf{B}_{i,j} = B_j(x_i)$ and $\Omega_{j,k} = \int_{x_1}^{x_N} B_j''(t)B_k''(t)\,dt$ for $2 \le j, k \le N + 3$, restricting to $[x_1, x_N]$, and taking the unique extension to a natural cubic spline on $[x_1, x_N]$.

Exercise 5.9

Derive the Reinsche form $\mathbf{S}_\lambda = (\mathbf{I} + \lambda\mathbf{K})^{-1}$ for the smoothing spline.

Solution

Since $\mathbf{N}$ has rank N it is invertible and

$$\begin{aligned} \mathbf{S}_\lambda &= \mathbf{N}\left(\mathbf{N}^T\mathbf{N} + \lambda\Omega_N\right)^{-1}\mathbf{N}^T \\ &= \mathbf{N}\left(\mathbf{N}^T\left(\mathbf{I} + \lambda(\mathbf{N}^{-1})^T\Omega_N\mathbf{N}^{-1}\right)\mathbf{N}\right)^{-1}\mathbf{N}^T \\ &= \left(\mathbf{I} + \lambda(\mathbf{N}^{-1})^T\Omega_N\mathbf{N}^{-1}\right)^{-1}. \end{aligned}$$

So just take $\mathbf{K} = (\mathbf{N}^{-1})^T\Omega_N\mathbf{N}^{-1}$.

Exercise 5.10

Derive an expression for $\mathrm{Var}(\hat f_\lambda(x_0))$ and $\mathrm{bias}(\hat f_\lambda(x_0))$. Using the example (5.22), create a version of Figure 5.9 where the mean and several (pointwise) quantiles of $\hat f_\lambda(x)$ are shown.

Solution

Write $N(x_0) = (N_1(x_0), \dots, N_N(x_0))$ so that $\hat f_\lambda(x_0) = N(x_0)^T\hat\theta = N(x_0)^T\mathbf{N}^{-1}\mathbf{S}_\lambda\mathbf{y}$. By assumption, $Y \sim \mathcal{N}(f(X), \sigma^2)$ so $\hat f_\lambda(x_0)$ is also normally distributed with mean $N(x_0)^T\mathbf{N}^{-1}\mathbf{S}_\lambda\mathbf{f}$ and variance

$$N(x_0)^T\mathbf{N}^{-1}\mathbf{S}_\lambda\left(\sigma^2\mathbf{I}_N\right)\mathbf{S}_\lambda^T\left(\mathbf{N}^{-1}\right)^T N(x_0) = \sigma^2\|N(x_0)^T\mathbf{N}^{-1}\mathbf{S}_\lambda\|^2$$

Therefore

$$\mathrm{Var}(\hat f_\lambda(x_0)) = \sigma^2\|N(x_0)^T\mathbf{N}^{-1}\mathbf{S}_\lambda\|^2, \qquad \mathrm{bias}(\hat f_\lambda(x_0)) = \mathbf{f} - N(x_0)^T\mathbf{N}^{-1}\mathbf{S}_\lambda\mathbf{f}.$$

We use this to recreate Figure 5.9 with the mean and pointwise quantiles using a purpose built module `splquant.py`. To avoid overcrowding the figure we only show one quantile.

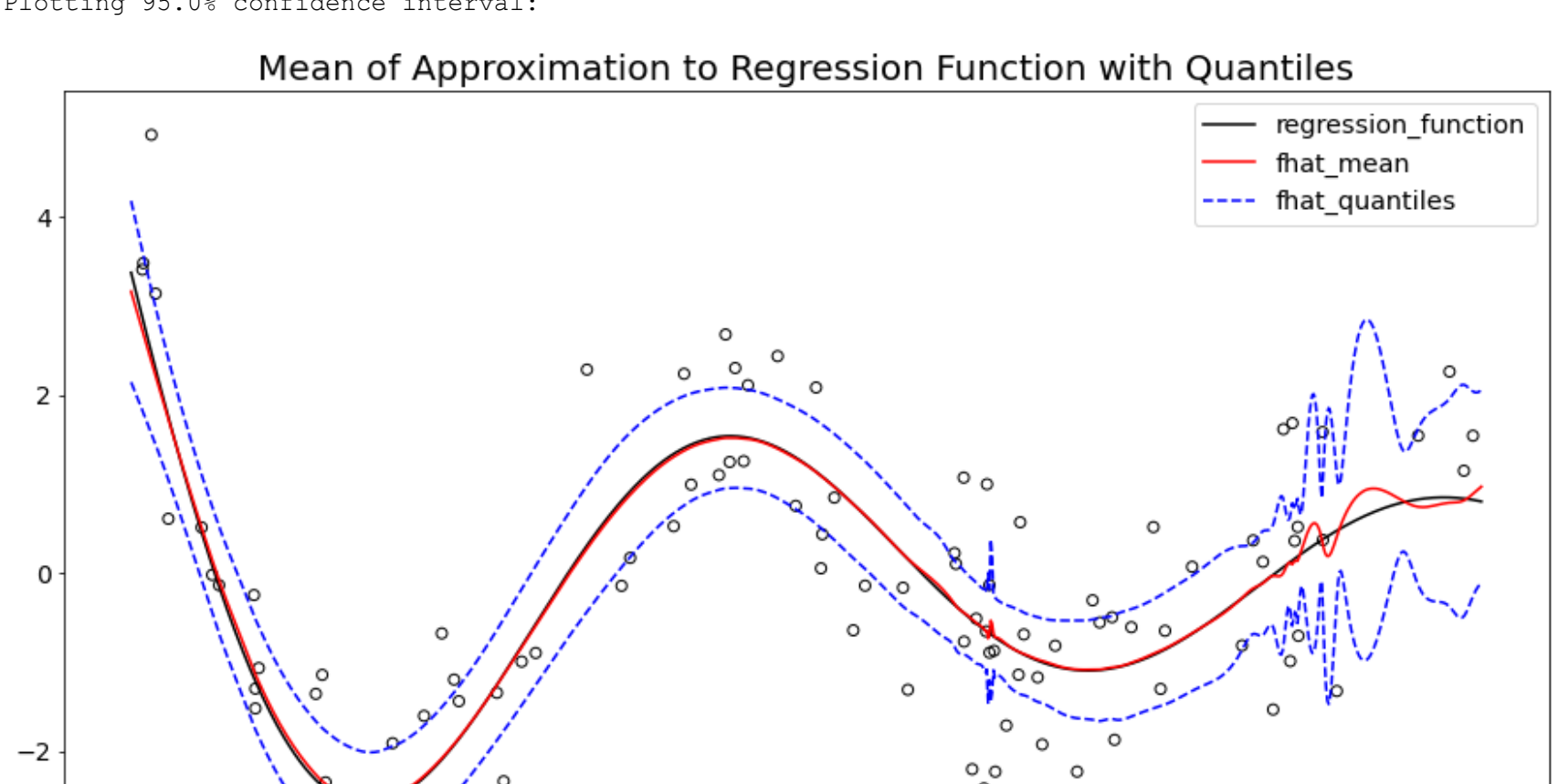
In [1]:

```
import numpy as np
from splquant import run_sim
```

In [3]:

```
# Parameters for simulation
eff_df = 12 # Effective degrees of freedom
alpha = 0.05
sample_range = (0, 1)
N = 100
sigma = 1

# Regression function to use
def reg_fun(x):
    return np.sin(12 * (x + 0.2)) / (x + 0.2)

run_sim(reg_fun, eff_df, alpha, sample_range, N, sigma)
```

```
Sampling 100 points uniformly on [0, 1]

Effective degrees of freedom: 12
Smoothing parameter lambda: 0.00009

Plotting 95.0% confidence interval:
```



Note that the red curve is $\mathrm{E}(\hat f_\lambda(x_0) \mid x_1, \dots, x_N)$, **not** the function $\hat f_\lambda(x_0)$ fitted to the sample data $\{(x_i, y_i)\}$. In fact none of the curves shown depend on the outputs y_i. It is also important to understand that the quantiles do **not** represent confidence bounds for the sample data or the regression function - we should not expect $f(x_0)$ to lie within the dashed lines. Rather, if we fit $\hat f_\lambda$ to the outputs y_i we expect $\hat f_\lambda(x_0)$ to lie within the quantiles with a specified degree of confidence.

Indeed, the difference between the regression function and $\mathrm{E}(\hat f_\lambda(x_0))$ is the (pointwise) bias of $\hat f_\lambda(x_0)$ as an estimator of $f(x_0)$ and for a few degrees of freedom this is significant. As the degrees of freedom increase, the bias decreases (the black and red lines become difficult to distinguish) but the variance increases and the quantiles move further apart.

Exercise 5.11

Prove that for a smoothing spline the null space of $\mathbf{K}$ is spanned by functions linear in X.

Solution

Suppose that $\mathbf{u} = (u_1, \dots, u_N)$ lies in the null space of $\mathbf{K}$. Since the space of natural cubic splines is N-dimensional, we can find one that interpolates the pairs $\{(x_i, u_i)\}_{i=1}^N$. That is, there exists $\beta = (\beta_1, \dots, \beta_N)$ such that $u_i = \sum_{j=1}^N \beta_j N_j(x_i) \Rightarrow \mathbf{u} = \mathbf{N}\beta$. We can also think of $\mathbf{u}$ as the evaluation of the function $u(X) = \sum_{j=1}^N \beta_j N_j(X)$ at $x_1, \dots, x_N$.

From Exercise 5.11, $\mathbf{K} = (\mathbf{N}^{-1})^T\Omega_N\mathbf{N}^{-1}$. Since $\mathbf{u}$ is in the null space,

$$0 = \mathbf{K}\mathbf{u} = \mathbf{K}\mathbf{N}\beta \quad \Rightarrow \quad \Omega_N\beta = 0.$$

From the definition, the jth entry of $\Omega_N\beta$ is

$$\sum_{k=1}^{N} \Omega_{jk}\beta_k = \int_0^1 \sum_{k=1}^{N} \beta_k N_j''(t)N_k''(t)\,dt = \int_0^1 N_j''(t)u''(t)\,dt.$$

Since this is zero for all j, $\int_0^1 s''(t)u''(t)\,dt = 0$ for any natural cubic spline $s(t)$. In particular this is true for $s(t) = u(t)$, so $u''(t)$ is identically zero. Thus $u(t)$ is linear.

Exercise 5.12

Characterize the solution to the following problem,

$$\min_f \mathrm{RSS}(f, \lambda) = \sum_{i=1}^{N} w_i\{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2\,dt,$$

where the $w_i \ge 0$ are observation weights.

Characterize the solution to the smoothing spline problem (5.9) when the training data have ties in X.

Solution

As Exercise 5.7, this problem has a unique minimiser which is a natural cubic spline with knots at the unique values of x_i. So we can write $f(x) = \sum_{j=1}^N N_j(x)\theta_j$. Let $\mathbf{W}$ denote the diagonal matrix with (i, i) entry w_i. Using the same notation as in (5.11), we can rewrite

$$\mathrm{RSS}(f, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T\mathbf{W}(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\Omega_N\theta$$

and so the solution of the minimisation problem is

$$\hat\theta = \left(\mathbf{N}^T\mathbf{W}\mathbf{N} + \lambda\Omega_N\right)^{-1}\mathbf{N}^T\mathbf{W}\mathbf{y}.$$

This is the same as the solution to the unweighted problem with $\mathbf{y}$ and $\mathbf{N}$ replaced with $\mathbf{W}^{1/2}\mathbf{y}$ and $\mathbf{W}^{1/2}\mathbf{N}$, respectively.

Now consider the smoothing spline problem where there are ties in X. Let $x^{(1)}, \dots, x^{(M)}$ denote the unique x-values. Suppose there are w_j observations for $x^{(j)}$ and let $\bar y^{(j)}$ denote their mean. That is,

$$\bar y^{(j)} = \frac{1}{w_j}\sum_{x_i = x^{(j)}} y_i.$$

As in Exercise 2.6, we have

$$\sum_{i=1}^{N} \{y_i - f(x_i)\}^2 = \sum_{j=1}^{M} w_j\{\bar y^{(j)} - f(x^{(j)})\}^2 + G(y_1, \dots, y_N).$$

So the smoothing spline problem is equivalent to the weighted problem

$$\min_f \sum_{j=1}^{M} w_j\{\bar y^{(j)} - f(x^{(j)})\}^2 + \lambda \int f''(t)^2\,dt.$$

Thus the solution of this problem is the same as the solution to the smoothing spline problem with no repeated X-values and

$$\tilde{\mathbf{y}} = \begin{pmatrix} \sqrt{w_1}\,\bar y^{(1)} \\ \sqrt{w_2}\,\bar y^{(2)} \\ \vdots \\ \sqrt{w_M}\,\bar y^{(M)} \end{pmatrix} \quad \text{and} \quad \tilde{\mathbf{N}} = \begin{pmatrix} \sqrt{w_1}N_1(x^{(1)}) & \sqrt{w_1}N_2(x^{(1)}) & \cdots \\ \sqrt{w_2}N_1(x^{(2)}) & \sqrt{w_2}N_2(x^{(2)}) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix},$$

where the $N_k(x)$ are a basis of natural cubic splines with knots $x^{(1)}, \dots, x^{(M)}$.

Exercise 5.13

You have fitted a smoothing spline $\hat f_\lambda$ to a sample of N pairs (x_i, y_i). Suppose you augment your original sample with the pair $x_0, \hat f_\lambda(x_0)$, and refit; describe the result. Use this to derive the N-fold cross-validation formula (5.26).

Solution

Let $\mathrm{RSS}_0(f, \lambda)$ denote the penalised residual sum-of-squares (5.9) with the extra point x_0 added. Then

$$\mathrm{RSS}_0(f, \lambda) = \mathrm{RSS}(f, \lambda) + (\hat f_\lambda(x_0) - f(x_0))^2 \ge \mathrm{RSS}(f, \lambda)$$

with equality if and only if $f(x_0) = \hat f_\lambda(x_0)$. Thus $\hat f_\lambda$ is still the minimiser for the augmented sample.

Take $1 \le i \le N$ and let $\hat f_\lambda^{(-i)}$ denote the fitted smoothing spline for the data with x_i removed. By the above, this is the same as the smoothing spline for the full data sample with y_i replaced with $\hat f_\lambda^{(-i)}(x_i)$. Let $\mathbf{y}^{(-i)}$ denote the corresponding output vector, so $\mathbf{y}^{(-i)} - \mathbf{y} = (\hat f_\lambda^{(-i)}(x_i) - y_i)\mathbf{e}_i$, where $\mathbf{e}_i$ is a standard basis vector.

Recall that the smoother matrix $\mathbf{S}_\lambda$ depends only on the x_i and λ, so in particular is the same for $\hat f_\lambda$ and $\hat f_\lambda^{(-i)}$. Thus $\hat f_\lambda^{(-i)}(x_i)$ equals the ith entry of

$$\mathbf{S}_\lambda\mathbf{y}^{(-i)} = \mathbf{S}_\lambda\mathbf{y} + \mathbf{S}_\lambda(\hat f_\lambda^{(-i)}(x_i) - y_i)\mathbf{e}_i.$$

That is,

$$\hat f_\lambda^{(-i)}(x_i) = \hat f_\lambda(x_i) + S_\lambda(i, i)(\hat f_\lambda^{(-i)}(x_i) - y_i).$$

This implies that

$$\begin{aligned} y_i - \hat f_\lambda^{(-i)}(x_i) &= y_i - \hat f_\lambda(x_i) - S_\lambda(i, i)(y_i - \hat f_\lambda^{(-i)}(x_i)) \\ \Rightarrow y_i - \hat f_\lambda^{(-i)}(x_i) &= \frac{y_i - \hat f_\lambda(x_i)}{1 - S_\lambda(i, i)}. \end{aligned}$$

Therefore the cross-validation error is

$$\begin{aligned} \mathrm{CV}(\hat f_\lambda) &= \frac{1}{N}\sum_{i=1}^{N} \left(y_i - \hat f_\lambda^{(-i)}(x_i)\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N} \left(\frac{y_i - \hat f_\lambda(x_i)}{1 - S_\lambda(i, i)}\right)^2. \end{aligned}$$

Exercise 5.14

Derive the constraints on the α_j in the thin-plate spline expansion (5.39) to guarantee that the penalty $J(f)$ is finite. How else could one ensure that the penalty was finite?

Solution

I didn't make much headway with this question. One can show that

$$\frac{\partial^2 f}{\partial z_i \partial z_j}(z) = \sum_{j=1}^{N} \alpha_j\left(\frac{2(z - x_j)(z - x_j)^T}{\|z - x_j\|^2} + (1 + 2\ln\|z - x_j\|)\,\mathbf{I}_2\right).$$

We need to determine when the components of this are integrable over $\mathbb{R}^2$.

We certainly require $\sum_{j=1}^N \alpha_j = 0$, since

$$\frac{\partial^2 f}{\partial z_1^2}(z) = \sum_{j=1}^{N} 2\alpha_j\frac{(z_1 - x_{j1})(z_2 - x_{j2})}{(z_1 - x_{j1})^2 + (z_2 - x_{j2})^2}$$

converges to $\sum_j \alpha_j$ as $z_1, z_2 \to \infty$ along the line $z_1 = z_2$, and the integral of its square can't be finite unless it converges to zero as $\|z\| \to \infty$.

However, I haven't been able to prove or disprove integrability over $\mathbb{R}^2$ with this condition, or derive more conditions. It seems to me that we need to use that $h_k''(z) \approx h_j''(z)$ for z large and derive a bound on the difference between them.

Alternatively, one could ensure that the penalty is finite by integrating over a (large) disk in $\mathbb{R}^2$ containing all the points x_j (the singularities at the x_j don't prevent convergence).

Exercise 5.15

This exercise derives some of the results quoted in Section 5.8.1. Suppose $K(x, y)$ satisfying the conditions (5.45) and let $f(x) \in \mathcal{H}_K$. Show that

(a) $\langle K(\cdot, x_i), f\rangle_{\mathcal{H}_K} = f(x_i)$.

(b) $\langle K(\cdot, x_i), K(\cdot, x_j)\rangle_{\mathcal{H}_K} = K(x_i, x_j)$.

(c) If $g(x) = \sum_{i=1}^N \alpha_i K(x, x_i)$, then

$$J(g) = \sum_{i=1}^{N}\sum_{j=1}^{N} K(x_i, x_j)\alpha_i\alpha_j.$$

(d) Suppose that $\tilde g(x) = g(x) + \rho(x)$, with $\rho(x) \in \mathcal{H}_K$, and orthogonal in $\mathcal{H}_K$ to each of $K(x, x_i)$, $i = 1, \dots, N$. Show that

$$\sum_{i=1}^{N} L(y_i, \tilde g(x_i)) + \lambda J(\tilde g) \ge \sum_{i=1}^{N} L(y_i, g(x_i)) + \lambda J(g)$$

with equality iff $\rho(x) = 0$.

Solution

The inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$ isn't explicitly defined in the text. There are two natural choices: take the bilinear extension of $\langle K(\cdot, x), K(\cdot, y)\rangle_{\mathcal{H}_K} := K(x, y)$, or, using the eigen-expansion of K, define

$$\langle f, g\rangle_{\mathcal{H}_K} = \sum_{i=1}^{\infty} \frac{c_i d_i}{\gamma_i}$$

where $f = \sum_{i=1}^\infty c_i\phi_i$ and $g = \sum_{i=1}^\infty d_i\phi_i$. In fact, taking $f = K(\cdot, x)$ and $g = K(\cdot, y)$ gives $c_i = \gamma_i\phi_i(x)$ and $d_i = \gamma_i\phi_i(y)$ so

$$\sum_{i=1}^{\infty} \frac{c_i d_i}{\gamma_i} = \sum_{i=1}^{\infty} \gamma_i\phi_i(x)\phi_i(y) = K(x, y)$$

so by bilinearity the two definitions agree.

(a) By bilinearity of $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$ it suffices to show this for $f = K(\cdot, y)$ and this is an immediate consequence of the definition.

(b) This is immediate from the definition of $\langle\cdot,\cdot\rangle_{\mathcal{H}_K}$.

(c) We have

$$\begin{aligned} J(g) &= \langle g, g\rangle_{\mathcal{H}_K} \\ &= \left\langle \sum_{i=1}^{N} \alpha_i K(\cdot, x_i), \sum_{j=1}^{N} \alpha_j K(\cdot, x_j)\right\rangle_{\mathcal{H}_K} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j K(x_i, x_j) \end{aligned}$$

as required.

(d) Orthogonality implies that

$$\rho(x_i) = \langle\rho, K(\cdot, x_i)\rangle_{\mathcal{H}_K} = 0$$

so $g(x_i) = \tilde g(x_i)$ for all i. Moreover since g is a linear combination of $K(\cdot, x_i)$, ρ is orthogonal to g and

$$J(\tilde g) = \|g + \rho\|_{\mathcal{H}_K}^2 = \|g\|_{\mathcal{H}_K}^2 + \|\rho\|_{\mathcal{H}_K}^2 = J(g) + \|\rho\|_{\mathcal{H}_K}^2.$$

Thus

$$\sum_{i=1}^{N} L(y_i, \tilde g(x_i)) + \lambda J(\tilde g) \ge \sum_{i=1}^{N} L(y_i, g(x_i)) + \lambda J(g)$$

with equality iff $\|\rho\|_{\mathcal{H}_K} = 0 \Leftrightarrow \rho = 0$.

This implies that the solution to the minimisation problem (5.48) is of the form $g = \sum_{i=1}^N \alpha_i K(\cdot, x_i)$ as claimed.

Exercise 5.16

Consider the ridge regression problem (5.53), and assume $M \ge N$. Assume you have a kernel K that computes the inner product $K(x, y) = \sum_{m=1}^M h_m(x)h_m(y)$.

(a) Derive (5.62) on page 171 of the text. How would you compute the matrices $\mathbf{V}$ and $\mathbf{D}_\gamma$, given K? Hence show that (5.63) is equivalent to (5.53).

(b) Show that

$$\begin{aligned} \hat{\mathbf{f}} &= \mathbf{H}\hat\beta \\ &= \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}, \end{aligned}$$

where $\mathbf{H}$ is the $N \times M$ matrix of evaluations $h_m(x_i)$, and $\mathbf{K} = \mathbf{H}\mathbf{H}^T$ the $N \times N$ matrix of inner products $h(x_i)^T h(x_j)$.

(c) Show that

$$\begin{aligned} \hat f(x) &= h(x)^T\hat\beta \\ &= \sum_{i=1}^{N} K(x, x_i)\hat\alpha_i \end{aligned}$$

and $\hat\alpha = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$.

(d) How would you modify your solution if $M < N$?

Solution

(a) Take $\mathcal{X} \subseteq \mathbb{R}^p$, assume that $h_m(x) \in L^2(\mathcal{X})$ for all m, and consider K as a function $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Define a linear operator $T_K : L^2(\mathcal{X}) \to L^2(\mathcal{X})$ by

$$(T_K\phi)(x) = \int_{\mathcal{X}} K(x, y)\phi(y)\,dy.$$

When we speak of eigenfunctions of K we really mean eigenfunctions of T_K.

If $\phi(x)$ is an eigenfunction of T_K then

$$(T_K\phi)(x) = \sum_{m=1}^{M}\left(\int_{\mathcal{X}} h_m(y)\phi(y)\,dy\right)h_m(x) = \gamma\phi(x)$$

for some $\gamma \in \mathbb{R}$. If $\gamma = 0$ then ϕ doesn't appear in the eigen-expansion (5.45) so we assume $\gamma \ne 0$. Then the equation above implies that we can write $\phi(x) = \sum_{m=1}^M a_m h_m(x)$. Substituting this in gives

$$\begin{aligned} \sum_{m=1}^{M}\sum_{n=1}^{M} a_n\left(\int_{\mathcal{X}} h_m(y)h_n(y)\,dy\right)h_m(x) &= \gamma\sum_{m=1}^{M} a_m h_m(x) \\ \Rightarrow h(x)^T\Omega\mathbf{a} &= \gamma h(x)^T\mathbf{a}, \end{aligned}$$

where $h(x) = (h_1(x), \dots, h_M(x))$ and Ω is the (m, n) entry

$$\omega_{m,n} = \int_{\mathcal{X}} h_m(y)h_n(y)\,dy.$$

Assume that $\mathrm{span}(\{h(x) \mid x \in \mathbb{R}^p\}) = \mathbb{R}^M$ (this isn't necessarily guaranteed since we could always add $h_{M+1}(x) = 0$ but seems like a harmless assumption). Then we must have $\Omega\mathbf{a} = \gamma\mathbf{a}$, so $\mathbf{a}$ is an eigenvector of Ω with eigenvalue γ.

Observe that Ω is positive-definite: if $z \in \mathbb{R}^M$ then

$$z^T\Omega z = \int\sum_{m=1}^{M}\sum_{n=1}^{M} z_m z_n h_m(y)h_n(y)\,dy = \int\left(h(y)^T z\right)^2 dy$$

so equals zero if and only if $z = 0$. Therefore we can decompose $\Omega = \mathbf{V}\mathbf{D}_\gamma\mathbf{V}^T$, where $\mathbf{V}$ is an orthogonal matrix whose columns are eigenvectors $v_1, \dots, v_M$ and $\mathbf{D}_\gamma$ is a diagonal matrix with positive entries $\gamma = (\gamma_1, \dots, \gamma_M)$. So the eigenfunctions are equal to $h(x)^T v_i$ up to a scalar multiple.

To set the scaling we require that $\phi_i(x)$ has $L^2(\mathcal{X})$ norm equal to one. Since

$$\int\left(h(y)^T v_i\right)^2 dy = v_i^T\Omega v_i = \gamma_i,$$

this implies that $\phi_i(x) = \gamma_i^{-1/2}h(x)^T v_i$ and hence

$$\phi(x) = \mathbf{D}_\gamma^{-\frac12}\mathbf{V}^T h(x) \quad \Rightarrow \quad h(x) = \mathbf{V}\mathbf{D}_\gamma^{\frac12}\phi(x),$$

where $\phi(x) = (\phi_1(x), \dots, \phi_M(x))$. This also demonstrates how to calculate $\mathbf{V}$ and $\mathbf{D}_\gamma$.

Now consider the minimisation problem (5.63). Using the equation above we can rewrite

$$\beta^T h(x_i) = \beta^T\mathbf{V}\mathbf{D}_\gamma^{\frac12}\phi(x) = \left(\mathbf{D}_\gamma^{\frac12}\mathbf{V}^T\beta\right)\phi(x)$$

so let $c = \mathbf{D}_\gamma^{\frac12}\mathbf{V}^T\beta$. Then

$$\sum_{j=1}^{\infty} \frac{c_j^2}{\gamma_j} = \|\mathbf{D}_\gamma^{-\frac12}c\|_2^2 = \|\mathbf{V}^T\beta\|_2^2 = \beta^T\mathbf{V}\mathbf{V}^T\beta = \sum_{m=1}^{M} \beta_m^2.$$

So minimising (5.63) over all β is equivalent to minimising (5.53) over all c.

(b) This minimisation problem (5.63) can be rewritten as

$$\min_\beta\,(\mathbf{y} - \mathbf{H}\beta)^T(\mathbf{y} - \mathbf{H}\beta) + \lambda\beta^T\beta.$$

As with ridge regression, this has a unique solution

$$\hat\beta = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}_M\right)^{-1}\mathbf{H}^T\mathbf{y}$$

and then the vector of fitted values is $\hat{\mathbf{f}} = \mathbf{H}\hat\beta$.

It just remains to note that

$$\begin{aligned} \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}_M\right)\mathbf{H}^T &= \mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}_N\right) \\ \Rightarrow \mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}_N\right)^{-1} &= \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}_M\right)^{-1}\mathbf{H}^T, \end{aligned}$$

so

$$\begin{aligned} \hat{\mathbf{f}} &= \mathbf{H}\hat\beta \\ &= \mathbf{H}\mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}_N\right)^{-1}\mathbf{y} \\ &= \mathbf{K}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{y}. \end{aligned}$$

This requires various matrices to be invertible, but this will be the case for all but a finite number of values of λ.

(c) We have

$$\begin{aligned} \hat f(x) &= h(x)^T\hat\beta \\ &= h(x)^T\mathbf{H}^T(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{y} \\ &= (\mathbf{H}h(x))^T\hat\alpha \end{aligned}$$

and the ith entry of $\mathbf{H}h(x)$ is

$$\sum_{m=1}^{M} h_m(x_i)h_m(x) = K(x, x_i).$$

The claim follows.

(d) As far as I can see the only place that we might have used the assumption $M \ge N$ was in taking the inverse of the $N \times N$ matrix $\mathbf{K} + \lambda\mathbf{I}_N$. But as noted then, this will only be singular for finitely many values of λ so this isn't an issue.

Exercise 5.17

Show how to convert the discrete eigen-decomposition of K in Section 5.8.2 to estimates of the eigenfunctions of K.

Solution

Let K be a positive-definite kernel and let $\mathbf{K}$ be the kernel matrix with $\mathbf{K}_{m,n} = K(x_m, x_n)$. Take the eigendecomposition of $\mathbf{K}$:

$$\mathbf{K} = \hat{\mathbf{\Phi}}\mathbf{D}_{\hat\gamma}\hat{\mathbf{\Phi}}^T.$$

Note that we have added hats to the notation in Section 5.8.2. Denote the (m, i) entry of $\hat{\mathbf{\Phi}}$ by $\hat\phi_{m,i}$

The eigen-expansion $K(x, y) = \sum_{i=1}^\infty \gamma_i\phi_i(x)\phi_j(y)$ of K implies that

$$\mathbf{K} = \mathbf{\Phi}\mathbf{D}_\gamma\mathbf{\Phi}^T,$$

where $\mathbf{\Phi}$ is the $N \times \mathbb{N}$ with (m, i) entry $\phi_i(x_m)$ and $\mathbf{D}_\gamma$ is an $\mathbb{N} \times \mathbb{N}$ matrix. Motivated by this we approximate $\gamma_i \approx \hat\gamma_i$ and $\phi_i(x_m) \approx \hat\phi_{m,i}$ for $1 \le i \le M$ and $\gamma_i \approx 0$ for $i > M$.

More generally, the eigen-expansion of K yields

$$\begin{pmatrix} \vdots \\ K(x, x_m) \\ \vdots \end{pmatrix} = \mathbf{\Phi}\mathbf{D}_\gamma\phi(x).$$

for any $x \in \mathbb{R}^p$. Using the approximations above and rearranging, this gives

$$\begin{aligned} \phi(x) &= \mathbf{D}_{\hat\gamma}^{-1}\hat{\mathbf{\Phi}}^T\begin{pmatrix} \vdots \\ K(x, x_m) \\ \vdots \end{pmatrix} \\ \Rightarrow \phi_i(x) &\approx \hat\gamma_i^{-1}\sum_{m=1}^{M} \hat\phi_{m,i}K(x, x_m). \end{aligned}$$

Exercise 5.18

The wavelet function $\psi(x)$ of the symmlet-p wavelet basis has vanishing moments up to order p. Show that this implies that polynomials of order p are represented exactly in V_0, defined on page 176.

Solution

We will require a standard fact about summlet-p wavelets: the nested subspaces $\cdots \subseteq V_{j-1} \subseteq V_j \subseteq V_{j+1} \subseteq \cdots$ defined by the symmlet-p wavelets comprise a *multiresolution analysis* of $L^2(\mathbb{R})$. In fact all we require it that the union $\bigcup_{j\in\mathbb{N}} V_j$ is dense in $L^2(\mathbb{R})$.

Suppose that f is a polynomial with $\deg(f) \le p - 1$. Recall that 'vanishing moments' means that $\int\psi(x)x^m\,dx = 0$ for $m = 0, \dots, p - 1$. This implies that

$$\int\psi_{j,k}(x)x^m\,dx = \int 2^{j/2}\psi(2^j x - k)x^m\,dx = 2^{-j/2}\int\psi(t)\left(\frac{t + k}{2^j}\right)^m dt = 0$$

for $m = 0, \dots, p - 1$ and $j, k \in \mathbb{Z}$ and thus $\langle f, \psi_{j,k}\rangle_2 = 0$ for all $j, k \in \mathbb{Z}$, where $\langle\cdot,\cdot\rangle_2$ denotes the inner product in $L^2(\mathbb{R})$.

Take a finite interval $I = [a, b] \subseteq \mathbb{R}$. Let S_I denote the vector space generated by those symmlet-p wavelets $\psi_{j,k}$ ($j \ge 0$) whose support intersects I but is not contained in it. Then

$$\langle f\mathbf{1}_I, \psi_{j,k}\rangle_2 = \langle f, \psi_{j,k}\rangle_2 = 0$$

for $\psi_{j,k} \notin S_I$.

By the density property, there exists a sequence $(g_n)_{n\in\mathbb{N}}$ of elements of $\bigcup_{j\in\mathbb{N}} V_j$ such that $\|f\mathbf{1}_I - g_n\|_2 \to 0$ as $n \to \infty$. We can write each g_n as a sum of basis elements:

$$g_n = \sum_{k=-\infty}^{\infty} \alpha_k\phi_{j,k} + \sum_{j=0}^{\infty}\sum_{k=-\infty}^{\infty} \beta_{j,k}\cdot\psi_{j,k}$$

Then

$$\begin{aligned} \|f\mathbf{1}_I - g_n\|_2^2 &= \|f\mathbf{1}_I\|_2^2 - 2\langle f\mathbf{1}_I, g_n\rangle_2 + \|g_n\|_2^2 \\ &= \|f\mathbf{1}_I\|_2^2 - 2\langle f\mathbf{1}_I, g_n\rangle_2 + \left(\sum_{k=-\infty}^{\infty} \alpha_k^2 + \sum_{j=0}^{\infty}\sum_{k=-\infty}^{\infty} \beta_{j,k}^2\right). \end{aligned}$$

Since $\psi_{j,k} \notin S_I$ implies that $\langle f\mathbf{1}_I, \psi_{j,k}\rangle_2 = 0$, removing these components can only decrease $\|f\mathbf{1}_I - g_n\|_2$ and we may assume that $g_n \in V_0 \oplus S_I$ for all n. We express this decomposition as $g_n = g_n^{(v)} + g_n^{(w)}$.

Suppose that the support of ψ is an interval of length $2p\tau$ (the text mentions '$2p - 1$ consective time intervals' but doesn't give the precise scaling). Then the support of S_I satisfies

$$\mathrm{supp}(S_I) \subseteq [a - 2p\tau, a + 2p\tau] \cup [b - 2p\tau, b + 2p\tau]$$

(recall $I = [a, b]$), and so if $\tilde I = (a + 2p\tau, b - 2p\tau)$ then $\mathrm{supp}(S_I) \cap \tilde I = \emptyset$. In particular, this implies that $g_n(x) = g_n^{(v)}(x)$ on $\tilde I$.

Since

$$\begin{aligned} \|f\mathbf{1}_I - g_n\|_2^2 &= \int_{\mathbb{R}} (f(x)\mathbf{1}_I(x) - g_n(x))^2\,dx \\ &= \int_{\mathbb{R}\setminus\tilde I} (f(x)\mathbf{1}_I(x) - g_n(x))^2\,dx + \int_{\tilde I} (f(x)\mathbf{1}_I(x) - g_n(x))^2\,dx \\ &= \int_{\mathbb{R}\setminus\tilde I} (f(x)\mathbf{1}_I(x) - g_n(x))^2\,dx + \int_{\tilde I} \left(f(x)\mathbf{1}_I(x) - g_n^{(v)}(x)\right)^2 dx, \end{aligned}$$

the sequence $g_n^{(v)}$ converges to $f\mathbf{1}_I$ in $L^2(\tilde I)$. But the quantity $\int_{\tilde I} (f(x)\mathbf{1}_I(x) - g(x))^2\,dx$ has a minimum among $g \in V_0$, so this must be zero and thus $f = g$ on $\tilde I$ by continuity.

Therefore, by taking I sufficiently large we can represent f by elements of V_0 on any finite subinterval of $\mathbf{R}$. Moreover, since elements of V_0 have finite support, we can represent f on $\mathbf{R}$ by an element of the completion of V_0 in which infinite sums of $\phi_{0,k}$ are permitted.

Exercise 5.19

Show that the Haar wavelet transform of a signal of length $N = 2^J$ can be computed in $O(N)$ computations.

Solution

We slightly alter the definition of the Haar father to be $\phi = I_{[0,1)}$ with an open bracket on the right. Suppose that the domain is discrete with points at $x_i^{(J)} = \frac{i-1}{2^J}$ for $1 \le i \le N = 2^J$ and take a signal $\mathbf{y}$ of length N. Recall that the Haar basis consists of ϕ and $\psi_{j,k}$ for $0 \le j \le J - 1$ and $0 \le k \le 2^j - 1$. The Haar wavelet transform of $\mathbf{y}$ is $\mathbf{y}^* = \mathbf{W}^T\mathbf{y}$, where the ith column of $\mathbf{W}$ is the ith Haar basis element evaluated at $x_1^{(J)}, \dots, x_N^{(J)}$.

We can view the Haar wavelet transform as a recursive procedure: for $0 \le j < J - 1$ and $0 \le k \le 2^j - 1$, applying $(\psi_{j,k}(x_1^{(J)}) \cdots \psi_{j,k}(x_{2^J}^{(J)}))$ to $\mathbf{y}$ is the same as applying $(\psi_{j,k}(x_1^{(J-1)}) \cdots \psi_{j,k}(x_{2^{J-1}}^{(J-1)}))$ to the vector $(y_1 + y_2, \cdots, y_{N-1} + y_N)$ of length 2^{J-1}. Thus, since

$$\begin{pmatrix} \psi_{J,k}(x_1^{(J)}) & \cdots & \psi_{J,k}(x_N^{(J)}) \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = y_{2k+1} - y_{2k+2},$$

taking the Haar wavelet transform of $\mathbf{y}$ is equivalent to taking the transform of the pairwise sum signal $(y_1 + y_2, \cdots, y_{N-1} + y_N)$ of length 2^{J-1} and calculating the pairwise difference vector $(y_1 - y_2, \cdots, y_{N-1} - y_N)$.

Let C_J denote the computation time for a signal of length 2^J. Calculating the pairwise sums and differences takes 2^J computations and so $C_J = 2^J + C_{J-1}$. Since $C_0 = 0$ (the Haar transform of a single point is itself), this implies that

$$C_J = \sum_{j=1}^{J} 2^j = 2^{J+1} - 2 = 2N - 2 \in O(N)$$

as required.

In [ ]: