6.2. Problem background: to understand better the role of the \( l_i(x) \) functions, note the logic of the discussion in section 6.1.1. We are working with the one dimensional model \( y = f(x) + \epsilon \), and assume our dataset \( T = \{(x_i, y_i)\}_{i=1}^{N} \) is given. We use a standard basis method assuming

\[
f(x) = \alpha + \sum_{j=1}^{d} \beta_j x^j = \sum_{j=0}^{d} \beta_j x^j. \tag{1a}
\]

However, we want our approximation to focus on points near some fixed point \( x_0 \), so we weight the training points \( x_i \) more if they are close to \( x_0 \). We assume \( x_0 \) remains fixed. The weight of the training point \( x_i \) will be given by a formula

\[
w_i = K(x_0, x_i)
\]

where \( K \) is largest when \( x_i \) is close to \( x_0 \). Otherwise we go through the identical least squares regression with the basis functions \( h_0(x) = 1, h_1(x) = x, h_2(x) = x^2, \ldots, h_d(x) = x^d \). This basis method works just like linear regression (see class notes), in that we define the function

\[
h(x) = \begin{bmatrix} h_0(x) \\ h_1(x) \\ \vdots \\ h_d(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{bmatrix}.
\]

(notation: in the text for this problem \( h(x) = b(x) \)).

Thus using our previous notation (letting \( \alpha = \beta_0 \)) we have

\[
f(x) = \sum_{j=0}^{d} \beta_j h_j(x) = \beta^T h(x).
\]

We then can write the estimate for \( f \) using least squares error as

\[
\hat{f} = \arg\min_{f = \beta^T h} \sum_{i=1}^{N} w_i (f(x_i) - y_i)^2. \tag{1}
\]

Equivalently, \( \hat{f}(x) = \beta^T h(x) \), where

\[
\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} w_i (\beta^T h(x_i) - y_i)^2 \tag{2}
\]

We form the standard data matrix (which we usually call \( H \))
\[ B = H = \begin{bmatrix} h(x_1)^T \\ h(x_2)^T \\ \vdots \\ h(x_N)^T \end{bmatrix}, \]

and then can write from (2):

\[ \hat{\beta} = \arg\min_{\beta} (B\beta - y)^T W (B\beta - y) = \arg\min_{\beta} (\sqrt{W} (y - B^T \beta))^2. \]

where

\[ W = \text{diag}(w_i) = \begin{bmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \ldots & w_N \end{bmatrix} \]

(we assume \( \sum w_i = 1 \)) and \( \sqrt{W} \) has diagonal entries \( \sqrt{w_i} \).

We obtain the usual solution of (1) (since this problem is identical to the regression problem of Chapter 3), obtaining (including matrix \( W \), which was previously just \( I \))

\[ \hat{\beta} = (B^T W B)^{-1} B^T W y. \]

Thus

\[ \hat{f}(x) = h^T(x) \hat{\beta} \equiv b^T(x) \hat{\beta} = b^T(x)(B^T W B)^{-1} B^T W y, \quad (3) \]

as in (6.8). Note we do not need to have \( x = x_0 \). In (3) we are assuming that \( W = W(x_0) \) (i.e. the weights are frozen to fit with the point \( x_0 \)). However, the approximation is valid for all \( x \), assuming we understand that it is only interesting near \( x \approx x_0 \). Finally, equation (6.9) can be correspondingly written not just for \( x_0 \):

\[ \hat{f}(x) = \sum_{i=1}^{N} l_i(x) y_i, \quad (4) \]

where now the functions \( l_i(x) \) are not just defined for \( x_0 \) as in the text, but for all \( x \).

Of course comparing (3) to (4), we see that the vector \( L(x) = (l_1(x), \ldots, l_N(x)) \) is just the \( (1 \times N) \) matrix \( b^T(x)(B^T W B)^{-1} B^T W \) in (3) above.

Problem suggestion:

To solve the problem, you can set up a standard local polynomial (weighted) regression (1a). Again fixing \( x_0 \), you want a least-squares fit for the coefficient vector \( \beta = (\beta_1, \beta_2, \ldots, \beta_k)^T \), given by

\[ \hat{\beta} = \arg\min_{\beta} (\sqrt{W} (y - B^T \beta))^2, \]
where $B$ is the design matrix with entry $b_{ij} = (x_i - x_0)^j$, and $W$ is the diagonal weight matrix with $i^{th}$ diagonal element $w_i = K_\lambda(x_0, x_i)$ (with $\sqrt{W}$ diagonal having entries $\sqrt{K_\lambda(x_0, x_i)}$). Show this means we are freezing the point $x_0$, and doing a regular least squares regression by minimizing

$$(\sqrt{W}(y - B^T\beta))^2 = \sum_i W_{ii}(y_i - \hat{f}(x_i))^2 = \sum_i K_\lambda(x_0, x_i)(y_i - \hat{f}(x_i))^2,$$

where

$$\hat{f}(x) = \sum_{j=1}^k \beta_j(x - x_0)^j.$$

Fixing $j$, define $\hat{f}_j(x)$ to be the best fit regression function if we happen to have $y_i = (x_i - x_0)^j$ for the training set $T$ (that is, if the data set $T$ happened to fit this equation). Show (you can use the definition of least squares) then $\hat{f}_j(x) = (x - x_0)^j$.

Show

$$\hat{f}_j(x) = \sum_{i=1}^N (x_i - x_0)^j l_i(x),$$

and

$$0 = \sum_{i=1}^N (x_i - x_0)^j l_i(x_0).$$

Show the implication based on (6.10) is that a local polynomial regression has bias $E(\hat{f}(x_0) - f(x_0))$ at $x = x_0$ which is not quite zero, but has a first non-zero remainder term

$$\frac{f^{(k+1)}(x_0)}{(k + 1)!} \sum_{i=1}^N (x_i - x_0)^{k+1} l_i(x_0).$$

Use Taylor's theorem with remainder to show then that a good estimate for the full sum of the remainder terms in the bias would be the actual remainder

$$R = \sum_{i=1}^N \frac{f^{(k+1)}(x_i^*)}{(k + 1)!} (x_i - x_0)^{k+1} l_i(x_0),$$

with $x_i^*$ some point between $x_0$ and $x_i$. Given higher terms in a Taylor expansion are less significant, why should this be small?

6.10. Look at 7.4 first. What is Cov$(y_i, \tilde{y}_i)$ in the present case? Write $y_i = f(x_i) + \epsilon_i$ and note you can replace $y_i$ by $\epsilon_i$ in the Cov formula. Write $S_\lambda \epsilon$ as a sum and compute.
7.4. First consider

\[ E_y(\text{Err}_\text{in}) = \frac{1}{N} \sum_{i=1}^{N} E_{y_{i,y}} [(Y_i^0 - \hat{f}(x_i))^2] = \frac{1}{N} \sum_{i=1}^{N} E_{y_{i,y}}[(Y_i^0 - \tilde{y}_i)^2], \]

while

\[ E_y(\text{err}) = \frac{1}{N} \sum_{i=1}^{N} E_y [(y_i - \hat{f}(x_i))^2] = \frac{1}{N} \sum_{i=1}^{N} E_{y_{i,y}}[(y_i - \tilde{y}_i)^2]. \]

Show first that \( \hat{f}(x) \) implicitly depends on the original dataset \( y = (y_1, ..., y_N)^T \), but not on the new data \( Y_0 = (Y_1^0, ..., Y_N^0) \). Conclude \( E_y \) affects \( \hat{f}(x) \), but \( E_{y_0} \) does not.

You can define \( A = Y_i^0 - \tilde{y}_i \), \( B = y_i - \tilde{y}_i \). Show \( E(A) = E(B) = f(x_i) - E(\tilde{y}_i) \).

Now show (letting \( E = E_{Y \cdot Y} \)):

\[ = E[(Y_i^0 - \tilde{y}_i)^2] - E[(y_i - \tilde{y}_i)^2] \\
= E(A^2) - E(B^2) \\
= V(A) - V(B) \\
= V(Y_i^0) + V(\tilde{y}_i) - 2\text{Cov}(Y_i^0, \tilde{y}_i) - V(y_i) - V(\tilde{y}_i) + 2\text{Cov}(y_i, \tilde{y}_i) \\
= 2\text{Cov}(y_i, \tilde{y}_i). \]

Why is \( \text{Cov}(y_i, Y_i^0) = 0? \) Why does \( V(Y_i^0) = V(y_i)? \)