5
Basis Expansions and Regularization

5.1 Introduction

We have already made use of models linear in the input features, both for regression and classification. Linear regression, linear discriminant analysis, logistic regression and separating hyperplanes all rely on a linear model. It is extremely unlikely that the true function $f(X)$ is actually linear in X. In regression problems, $f(X) = \mathrm{E}(Y|X)$ will typically be nonlinear and nonadditive in X, and representing $f(X)$ by a linear model is usually a convenient, and sometimes a necessary, approximation. Convenient because a linear model is easy to interpret, and is the first-order Taylor approximation to $f(X)$. Sometimes necessary, because with N small and/or p large, a linear model might be all we are able to fit to the data without overfitting. Likewise in classification, a linear, Bayes-optimal decision boundary implies that some monotone transformation of $\Pr(Y = 1|X)$ is linear in X. This is inevitably an approximation.

In this chapter and the next we discuss popular methods for moving beyond linearity. The core idea in this chapter is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived input features.

Denote by $h_m(X) : \mathbb{R}^p \mapsto \mathbb{R}$ the mth transformation of X, $m = 1, \ldots, M$. We then model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X), \tag{5.1}$$

a *linear basis expansion* in X. The beauty of this approach is that once the basis functions h_m have been determined, the models are linear in these new variables, and the fitting proceeds as before.

Some simple and widely used examples of the h_m are the following:

- $h_m(X) = X_m,\ m = 1, \ldots, p$ recovers the original linear model.
- $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ allows us to augment the inputs with polynomial terms to achieve higher-order Taylor expansions. Note, however, that the number of variables grows exponentially in the degree of the polynomial. A full quadratic model in p variables requires $O(p^2)$ square and cross-product terms, or more generally $O(p^d)$ for a degree-d polynomial.
- $h_m(X) = \log(X_j),\ \sqrt{X_j}, \ldots$ permits other nonlinear transformations of single inputs. More generally one can use similar functions involving several inputs, such as $h_m(X) = ||X||$.
- $h_m(X) = I(L_m \leq X_k < U_m)$, an indicator for a region of X_k. By breaking the range of X_k up into M_k such nonoverlapping regions results in a model with a piecewise constant contribution for X_k.

Sometimes the problem at hand will call for particular basis functions h_m, such as logarithms or power functions. More often, however, we use the basis expansions as a device to achieve more flexible representations for $f(X)$. Polynomials are an example of the latter, although they are limited by their global nature—tweaking the coefficients to achieve a functional form in one region can cause the function to flap about madly in remote regions. In this chapter we consider more useful families of *piecewise-polynomials* and *splines* that allow for local polynomial representations. We also discuss the *wavelet* bases, especially useful for modeling signals and images. These methods produce a *dictionary* $\mathcal{D}$ consisting of typically a very large number $|\mathcal{D}|$ of basis functions, far more than we can afford to fit to our data. Along with the dictionary we require a method for controlling the complexity of our model, using basis functions from the dictionary. There are three common approaches:

- Restriction methods, where we decide before-hand to limit the class of functions. Additivity is an example, where we assume that our model has the form

$$\begin{aligned} f(X) &= \sum_{j=1}^{p} f_j(X_j) \\ &= \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j). \end{aligned} \tag{5.2}$$

The size of the model is limited by the number of basis functions M_j used for each component function f_j.

- Selection methods, which adaptively scan the dictionary and include only those basis functions h_m that contribute significantly to the fit of the model. Here the variable selection techniques discussed in Chapter 3 are useful. The stagewise greedy approaches such as CART, MARS and boosting fall into this category as well.

- Regularization methods where we use the entire dictionary but restrict the coefficients. Ridge regression is a simple example of a regularization approach, while the lasso is both a regularization and selection method. Here we discuss these and more sophisticated methods for regularization.

5.2 Piecewise Polynomials and Splines

We assume until Section 5.7 that X is one-dimensional. A piecewise polynomial function $f(X)$ is obtained by dividing the domain of X into contiguous intervals, and representing f by a separate polynomial in each interval. Figure 5.1 shows two simple piecewise polynomials. The first is piecewise constant, with three basis functions:

$$h_1(X) = I(X < \xi_1), \quad h_2(X) = I(\xi_1 \le X < \xi_2), \quad h_3(X) = I(\xi_2 \le X).$$

Since these are positive over disjoint regions, the least squares estimate of the model $f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$ amounts to $\hat{\beta}_m = \bar{Y}_m$, the mean of Y in the mth region.

The top right panel shows a piecewise linear fit. Three additional basis functions are needed: $h_{m+3} = h_m(X)X$, $m = 1, \ldots, 3$. Except in special cases, we would typically prefer the third panel, which is also piecewise linear, but restricted to be continuous at the two knots. These continuity restrictions lead to linear constraints on the parameters; for example, $f(\xi_1^-) = f(\xi_1^+)$ implies that $\beta_1 + \xi_1\beta_4 = \beta_2 + \xi_1\beta_5$. In this case, since there are two restrictions, we expect to *get back* two parameters, leaving four free parameters.

A more direct way to proceed in this case is to use a basis that incorporates the constraints:

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = (X - \xi_1)_+, \quad h_4(X) = (X - \xi_2)_+,$$

where t_+ denotes the positive part. The function h_3 is shown in the lower right panel of Figure 5.1. We often prefer smoother functions, and these can be achieved by increasing the order of the local polynomial. Figure 5.2 shows a series of piecewise-cubic polynomials fit to the same data, with

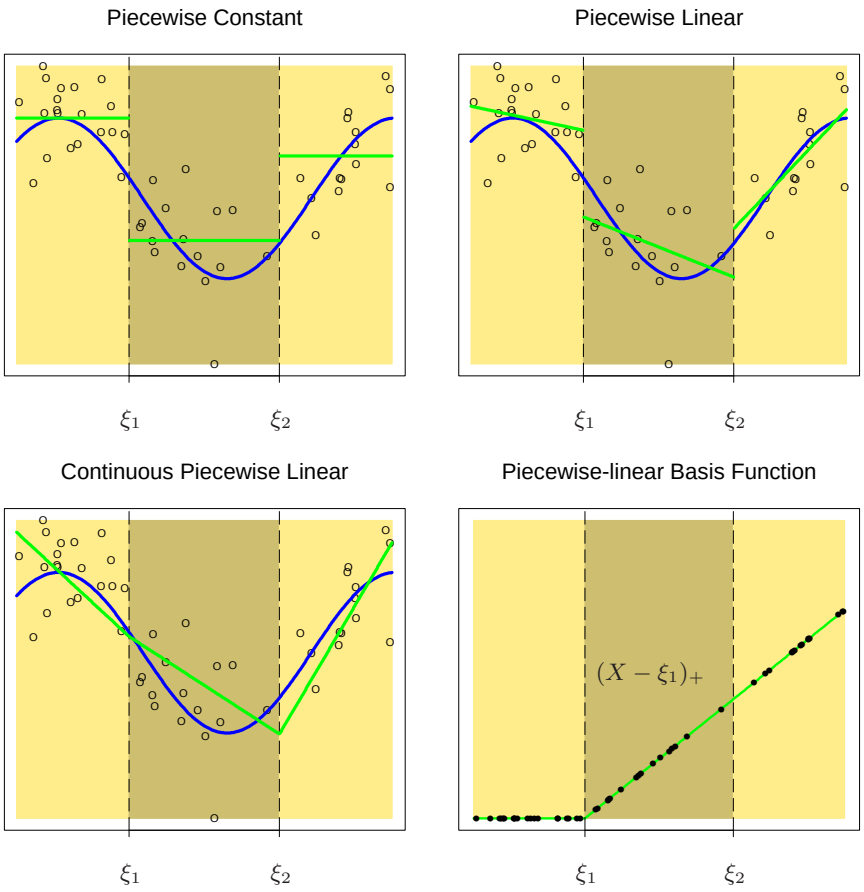

FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise linear functions fit to the same data—the top right unrestricted, and the lower left restricted to be continuous at the knots. The lower right panel shows a piecewise–linear basis function, $h_3(X) = (X - \xi_1)_+$, continuous at ξ_1. The black points indicate the sample evaluations $h_3(x_i)$, $i = 1, \ldots, N$.*

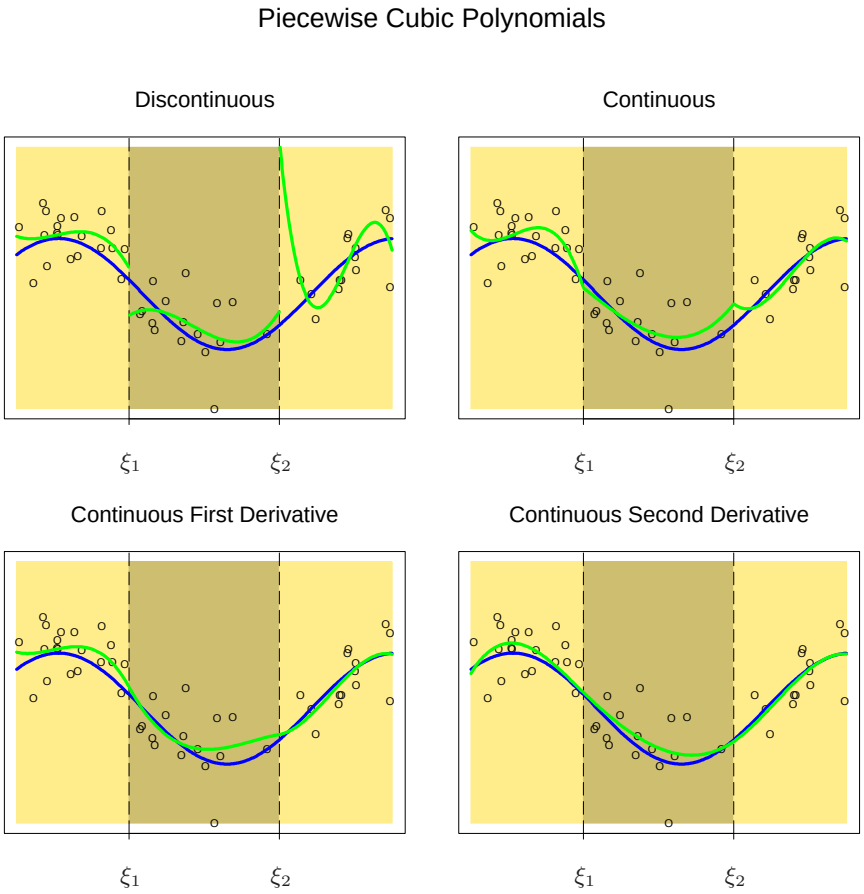

FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

increasing orders of continuity at the knots. The function in the lower right panel is continuous, and has continuous first and second derivatives at the knots. It is known as a *cubic spline*. Enforcing one more order of continuity would lead to a global cubic polynomial. It is not hard to show (Exercise 5.1) that the following basis represents a cubic spline with knots at ξ_1 and ξ_2:

$$\begin{aligned} h_1(X) &= 1, \quad h_3(X) = X^2, \quad h_5(X) = (X-\xi_1)^3_+, \\ h_2(X) &= X, \quad h_4(X) = X^3, \quad h_6(X) = (X-\xi_2)^3_+. \end{aligned} \tag{5.3}$$

There are six basis functions corresponding to a six-dimensional linear space of functions. A quick check confirms the parameter count: (3 regions)×(4 parameters per region) −(2 knots)×(3 constraints per knot)= 6.

More generally, an order-M spline with knots ξ_j, $j = 1, \ldots, K$ is a piecewise-polynomial of order M, and has continuous derivatives up to order $M-2$. A cubic spline has $M = 4$. In fact the piecewise-constant function in Figure 5.1 is an order-1 spline, while the continuous piecewise linear function is an order-2 spline. Likewise the general form for the truncated-power basis set would be

$$\begin{aligned} h_j(X) &= X^{j-1},\ j = 1, \ldots, M, \\ h_{M+\ell}(X) &= (X - \xi_\ell)_+^{M-1},\ \ell = 1, \ldots, K. \end{aligned}$$

It is claimed that cubic splines are the lowest-order spline for which the knot-discontinuity is not visible to the human eye. There is seldom any good reason to go beyond cubic-splines, unless one is interested in smooth derivatives. In practice the most widely used orders are $M = 1, 2$ and 4.

These fixed-knot splines are also known as *regression splines*. One needs to select the order of the spline, the number of knots and their placement. One simple approach is to parameterize a family of splines by the number of basis functions or degrees of freedom, and have the observations x_i determine the positions of the knots. For example, the expression `bs(x,df=7)` in `R` generates a basis matrix of cubic-spline functions evaluated at the N observations in `x`, with the $7-3 = 4$[1] interior knots at the appropriate percentiles of `x` (20, 40, 60 and 80th.) One can be more explicit, however; `bs(x, degree=1, knots = c(0.2, 0.4, 0.6))` generates a basis for linear splines, with three interior knots, and returns an $N \times 4$ matrix.

Since the space of spline functions of a particular order and knot sequence is a vector space, there are many equivalent bases for representing them (just as there are for ordinary polynomials.) While the truncated power basis is conceptually simple, it is not too attractive numerically: powers of large numbers can lead to severe rounding problems. The *B-spline* basis, described in the Appendix to this chapter, allows for efficient computations even when the number of knots K is large.

5.2.1 Natural Cubic Splines

We know that the behavior of polynomials fit to data tends to be erratic near the boundaries, and extrapolation can be dangerous. These problems are exacerbated with splines. The polynomials fit beyond the boundary knots behave even more wildly than the corresponding global polynomials in that region. This can be conveniently summarized in terms of the pointwise variance of spline functions fit by least squares (see the example in the next section for details on these variance calculations). Figure 5.3 compares

[1] A cubic spline with four knots is eight-dimensional. The `bs()` function omits by default the constant term in the basis, since terms like this are typically included with other terms in the model.

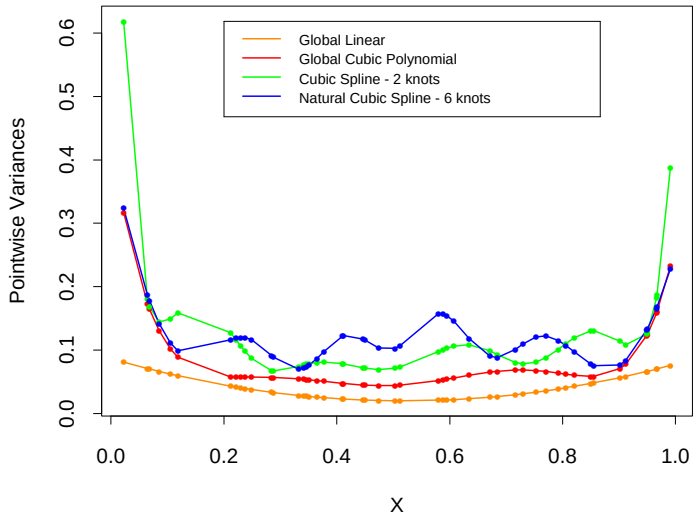

FIGURE 5.3. *Pointwise variance curves for four different models, with X consisting of 50 points drawn at random from $U[0, 1]$, and an assumed error model with constant variance. The linear and cubic polynomial fits have two and four degrees of freedom, respectively, while the cubic spline and natural cubic spline each have six degrees of freedom. The cubic spline has two knots at 0.33 and 0.66, while the natural spline has boundary knots at 0.1 and 0.9, and four interior knots uniformly spaced between them.*

the pointwise variances for a variety of different models. The explosion of the variance near the boundaries is clear, and inevitably is worst for cubic splines.

A *natural cubic spline* adds additional constraints, namely that the function is linear beyond the boundary knots. This frees up four degrees of freedom (two constraints each in both boundary regions), which can be spent more profitably by sprinkling more knots in the interior region. This tradeoff is illustrated in terms of variance in Figure 5.3. There will be a price paid in bias near the boundaries, but assuming the function is linear near the boundaries (where we have less information anyway) is often considered reasonable.

A natural cubic spline with K knots is represented by K basis functions. One can start from a basis for cubic splines, and derive the reduced basis by imposing the boundary constraints. For example, starting from the truncated power series basis described in Section 5.2, we arrive at (Exercise 5.4):

$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2}(X) = d_k(X) - d_{K-1}(X), \qquad (5.4)$$

where

$$d_k(X) = \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k}. \tag{5.5}$$

Each of these basis functions can be seen to have zero second and third derivative for $X \geq \xi_K$.

5.2.2 Example: South African Heart Disease (Continued)

In Section 4.4.2 we fit linear logistic regression models to the South African heart disease data. Here we explore nonlinearities in the functions using natural splines. The functional form of the model is

$$\text{logit}[\Pr(\texttt{chd}|X)] = \theta_0 + h_1(X_1)^T\theta_1 + h_2(X_2)^T\theta_2 + \cdots + h_p(X_p)^T\theta_p, \tag{5.6}$$

where each of the θ_j are vectors of coefficients multiplying their associated vector of natural spline basis functions h_j.

We use four natural spline bases for each term in the model. For example, with X_1 representing `sbp`, $h_1(X_1)$ is a basis consisting of four basis functions. This actually implies three rather than two interior knots (chosen at uniform quantiles of `sbp`), plus two boundary knots at the extremes of the data, since we exclude the constant term from each of the h_j.

Since `famhist` is a two-level factor, it is coded by a simple binary or dummy variable, and is associated with a single coefficient in the fit of the model.

More compactly we can combine all p vectors of basis functions (and the constant term) into one big vector $h(X)$, and then the model is simply $h(X)^T\theta$, with total number of parameters $\text{df} = 1 + \sum_{j=1}^p \text{df}_j$, the sum of the parameters in each component term. Each basis function is evaluated at each of the N samples, resulting in a $N \times \text{df}$ basis matrix $\mathbf{H}$. At this point the model is like any other linear logistic model, and the algorithms described in Section 4.4.1 apply.

We carried out a backward stepwise deletion process, dropping terms from this model while preserving the group structure of each term, rather than dropping one coefficient at a time. The AIC statistic (Section 7.5) was used to drop terms, and all the terms remaining in the final model would cause AIC to increase if deleted from the model (see Table 5.1). Figure 5.4 shows a plot of the final model selected by the stepwise regression. The functions displayed are $\hat{f}_j(X_j) = h_j(X_j)^T\hat{\theta}_j$ for each variable X_j. The covariance matrix $\text{Cov}(\hat{\theta}) = \mathbf{\Sigma}$ is estimated by $\hat{\mathbf{\Sigma}} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}$, where $\mathbf{W}$ is the diagonal weight matrix from the logistic regression. Hence $v_j(X_j) = \text{Var}[\hat{f}_j(X_j)] = h_j(X_j)^T\hat{\mathbf{\Sigma}}_{jj}h_j(X_j)$ is the pointwise variance function of $\hat{f}_j$, where $\text{Cov}(\hat{\theta}_j) = \hat{\mathbf{\Sigma}}_{jj}$ is the appropriate sub-matrix of $\hat{\mathbf{\Sigma}}$. The shaded region in each panel is defined by $\hat{f}_j(X_j) \pm 2\sqrt{v_j(X_j)}$.

The AIC statistic is slightly more generous than the likelihood-ratio test (deviance test). Both `sbp` and `obesity` are included in this model, while

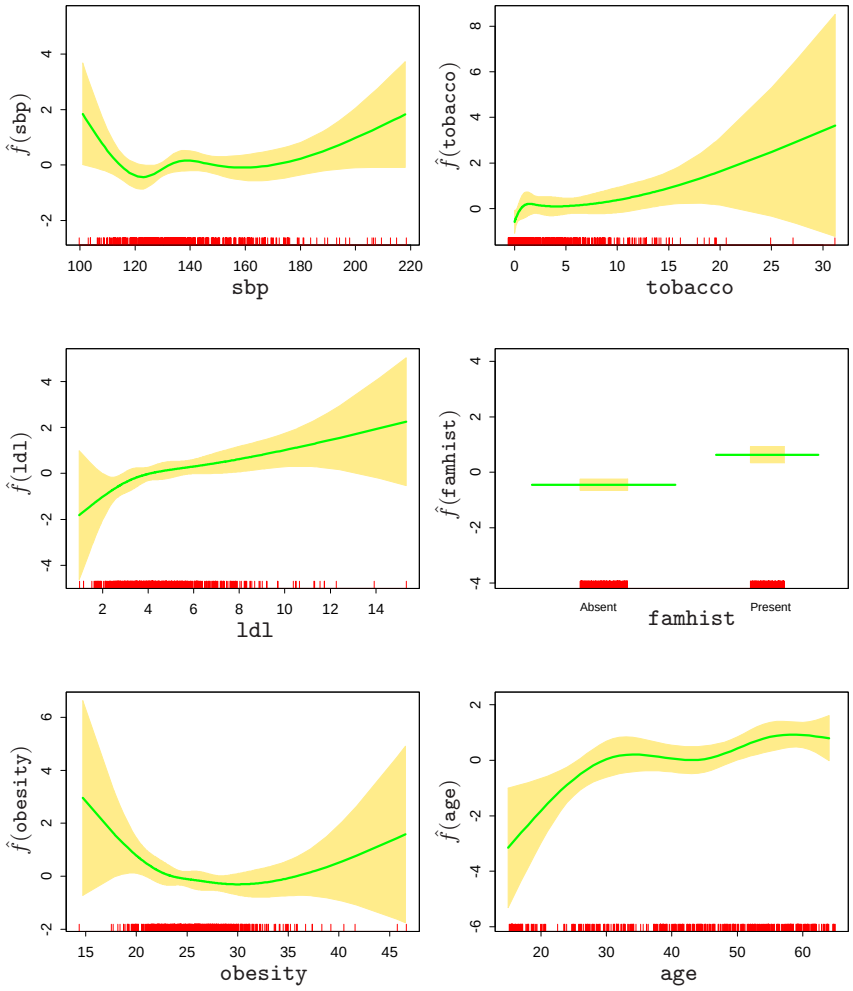

FIGURE 5.4. *Fitted natural-spline functions for each of the terms in the final model selected by the stepwise procedure. Included are pointwise standard-error bands. The* rug plot *at the base of each figure indicates the location of each of the sample values for that variable (jittered to break ties).*

TABLE 5.1. *Final logistic regression model, after stepwise deletion of natural splines terms. The column labeled "LRT" is the likelihood-ratio test statistic when that term is deleted from the model, and is the change in deviance from the full model (labeled "none").*

Terms	Df	Deviance	AIC	LRT	P-value
none		458.09	502.09		
sbp	4	467.16	503.16	9.076	0.059
tobacco	4	470.48	506.48	12.387	0.015
ldl	4	472.39	508.39	14.307	0.006
famhist	1	479.44	521.44	21.356	0.000
obesity	4	466.24	502.24	8.147	0.086
age	4	481.86	517.86	23.768	0.000

they were not in the linear model. The figure explains why, since their contributions are inherently nonlinear. These effects at first may come as a surprise, but an explanation lies in the nature of the retrospective data. These measurements were made sometime after the patients suffered a heart attack, and in many cases they had already benefited from a healthier diet and lifestyle, hence the apparent *increase* in risk at low values for `obesity` and `sbp`. Table 5.1 shows a summary of the selected model.

5.2.3 Example: Phoneme Recognition

In this example we use splines to reduce flexibility rather than increase it; the application comes under the general heading of *functional* modeling. In the top panel of Figure 5.5 are displayed a sample of 15 log-periodograms for each of the two phonemes "aa" and "ao" measured at 256 frequencies. The goal is to use such data to classify a spoken phoneme. These two phonemes were chosen because they are difficult to separate.

The input feature is a vector x of length 256, which we can think of as a vector of evaluations of a function $X(f)$ over a grid of frequencies f. In reality there is a continuous analog signal which is a function of frequency, and we have a sampled version of it.

The gray lines in the lower panel of Figure 5.5 show the coefficients of a linear logistic regression model fit by maximum likelihood to a training sample of 1000 drawn from the total of 695 "aa"s and 1022 "ao"s. The coefficients are also plotted as a function of frequency, and in fact we can think of the model in terms of its continuous counterpart

$$\log \frac{\Pr(\text{aa}|X)}{\Pr(\text{ao}|X)} = \int X(f)\beta(f)df, \tag{5.7}$$

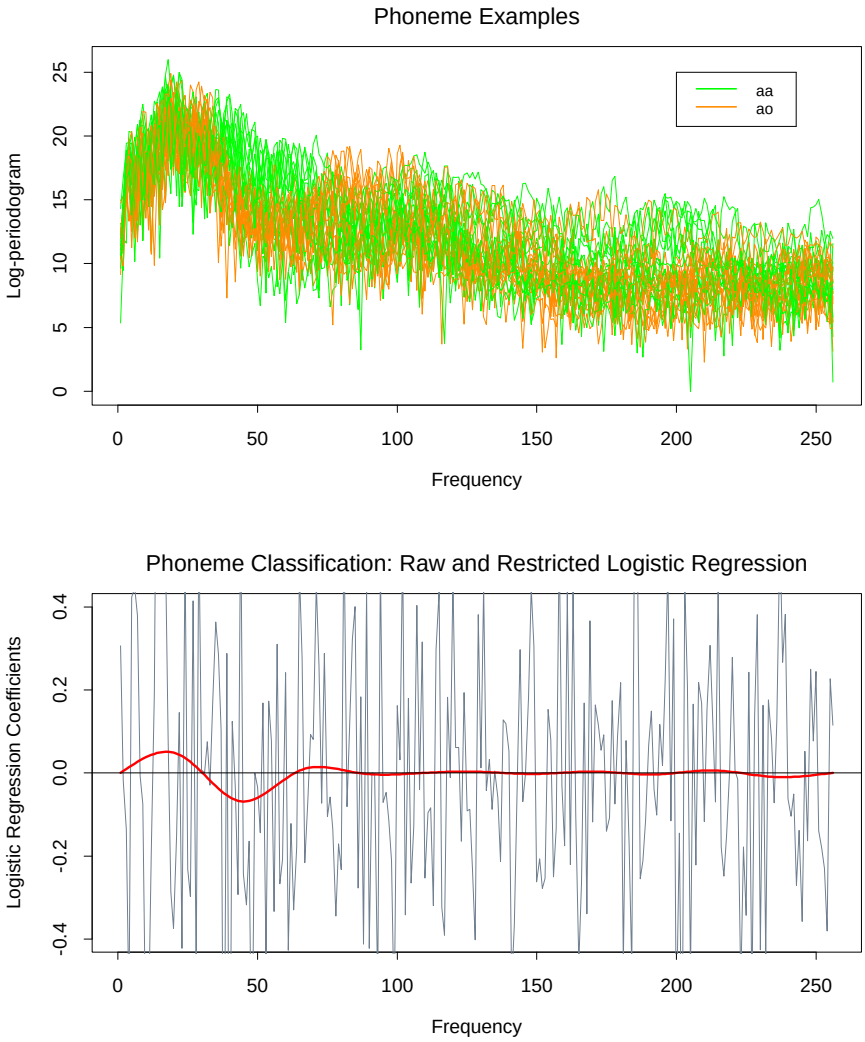

FIGURE 5.5. *The top panel displays the log-periodogram as a function of frequency for* 15 *examples each of the phonemes "aa" and "ao" sampled from a total of* 695 *"aa"s and* 1022 *"ao"s. Each log-periodogram is measured at* 256 *uniformly spaced frequencies. The lower panel shows the coefficients (as a function of frequency) of a logistic regression fit to the data by maximum likelihood, using the* 256 *log-periodogram values as inputs. The coefficients are restricted to be smooth in the red curve, and are unrestricted in the jagged gray curve.*

which we approximate by

$$\sum_{j=1}^{256} X(f_j)\beta(f_j) = \sum_{j=1}^{256} x_j\beta_j. \tag{5.8}$$

The coefficients compute a contrast functional, and will have appreciable values in regions of frequency where the log-periodograms differ between the two classes.

The gray curves are very rough. Since the input signals have fairly strong positive autocorrelation, this results in negative autocorrelation in the coefficients. In addition the sample size effectively provides only four observations per coefficient.

Applications such as this permit a natural regularization. We force the coefficients to vary smoothly as a function of frequency. The red curve in the lower panel of Figure 5.5 shows such a smooth coefficient curve fit to these data. We see that the lower frequencies offer the most discriminatory power. Not only does the smoothing allow easier interpretation of the contrast, it also produces a more accurate classifier:

	Raw	Regularized
Training error	0.080	0.185
Test error	0.255	0.158

The smooth red curve was obtained through a very simple use of natural cubic splines. We can represent the coefficient function as an expansion of splines $\beta(f) = \sum_{m=1}^{M} h_m(f)\theta_m$. In practice this means that $\beta = \mathbf{H}\theta$ where, $\mathbf{H}$ is a $p \times M$ basis matrix of natural cubic splines, defined on the set of frequencies. Here we used $M = 12$ basis functions, with knots uniformly placed over the integers $1, 2, \ldots, 256$ representing the frequencies. Since $x^T\beta = x^T\mathbf{H}\theta$, we can simply replace the input features x by their *filtered* versions $x^* = \mathbf{H}^T x$, and fit θ by linear logistic regression on the x^*. The red curve is thus $\hat{\beta}(f) = h(f)^T\hat{\theta}$.

5.3 Filtering and Feature Extraction

In the previous example, we constructed a $p \times M$ basis matrix $\mathbf{H}$, and then transformed our features x into new features $x^* = \mathbf{H}^T x$. These filtered versions of the features were then used as inputs into a learning procedure: in the previous example, this was linear logistic regression.

Preprocessing of high-dimensional features is a very general and powerful method for improving the performance of a learning algorithm. The preprocessing need not be linear as it was above, but can be a general