Rule 1	Global Mean	Box Mean	Box Support
Training	0.3931	0.9607	0.1413
Test	0.3958	1.0000	0.1536

$$\text{Rule 1}\left\{\begin{array}{rcl} \texttt{ch!} & > & 0.029 \\ \texttt{CAPAVE} & > & 2.331 \\ \texttt{your} & > & 0.705 \\ \texttt{1999} & < & 0.040 \\ \texttt{CAPTOT} & > & 79.50 \\ \texttt{edu} & < & 0.070 \\ \texttt{re} & < & 0.535 \\ \texttt{ch;} & < & 0.030 \end{array}\right.$$

Rule 2	Remain Mean	Box Mean	Box Support
Training	0.2998	0.9560	0.1043
Test	0.2862	0.9264	0.1061

$$\text{Rule 2}\left\{\begin{array}{rcl} \texttt{remove} & > & 0.010 \\ \texttt{george} & < & 0.110 \end{array}\right.$$

The box support is the proportion of observations falling in the box. The first box is purely `spam`, and contains about 15% of the test data. The second box contains 10.6% of the test observations, 92.6% of which are `spam`. Together the two boxes contain 26% of the data and are about 97% `spam`. The next few boxes (not shown) are quite small, containing only about 3% of the data.

The predictors are listed in order of importance. Interestingly the top splitting variables in the CART tree (Figure 9.5) do not appear in PRIM's first box.

9.4 MARS: Multivariate Adaptive Regression Splines

MARS is an adaptive procedure for regression, and is well suited for high-dimensional problems (i.e., a large number of inputs). It can be viewed as a generalization of stepwise linear regression or a modification of the CART method to improve the latter's performance in the regression setting. We introduce MARS from the first point of view, and later make the connection to CART.

MARS uses expansions in piecewise linear basis functions of the form $(x-t)_+$ and $(t-x)_+$. The "+" means positive part, so

$$(x-t)_+ = \begin{cases} x-t, & \text{if } x > t, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad (t-x)_+ = \begin{cases} t-x, & \text{if } x < t, \\ 0, & \text{otherwise.} \end{cases}$$

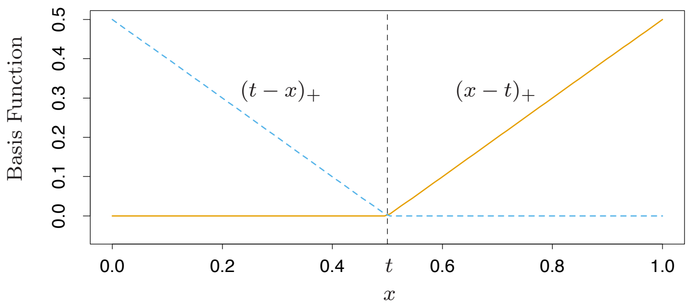

FIGURE 9.9. *The basis functions $(x - t)_+$ (solid orange) and $(t - x)_+$ (broken blue) used by MARS.*

As an example, the functions $(x - 0.5)_+$ and $(0.5 - x)_+$ are shown in Figure 9.9.

Each function is piecewise linear, with a *knot* at the value t. In the terminology of Chapter 5, these are linear splines. We call the two functions a *reflected pair* in the discussion below. The idea is to form reflected pairs for each input X_j with knots at each observed value x_{ij} of that input. Therefore, the collection of basis functions is

$$\mathcal{C} = \{(X_j - t)_+, \ (t - X_j)_+\} \ {}_{\substack{t \in \{x_{1j}, x_{2j}, \dots, x_{Nj}\} \\ j = 1, 2, \dots, p.}} \tag{9.18}$$

If all of the input values are distinct, there are $2Np$ basis functions altogether. Note that although each basis function depends only on a single X_j, for example, $h(X) = (X_j - t)_+$, it is considered as a function over the entire input space $\mathbb{R}^p$.

The model-building strategy is like a forward stepwise linear regression, but instead of using the original inputs, we are allowed to use functions from the set $\mathcal{C}$ and their products. Thus the model has the form

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X), \tag{9.19}$$

where each $h_m(X)$ is a function in $\mathcal{C}$, or a product of two or more such functions.

Given a choice for the h_m, the coefficients β_m are estimated by minimizing the residual sum-of-squares, that is, by standard linear regression. The real art, however, is in the construction of the functions $h_m(x)$. We start with only the constant function $h_0(X) = 1$ in our model, and all functions in the set $\mathcal{C}$ are candidate functions. This is depicted in Figure 9.10.

At each stage we consider as a new basis function pair all products of a function h_m in the model set $\mathcal{M}$ with one of the reflected pairs in $\mathcal{C}$. We add to the model $\mathcal{M}$ the term of the form

$$\hat{\beta}_{M+1} h_\ell(X) \cdot (X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X) \cdot (t - X_j)_+, \ h_\ell \in \mathcal{M},$$

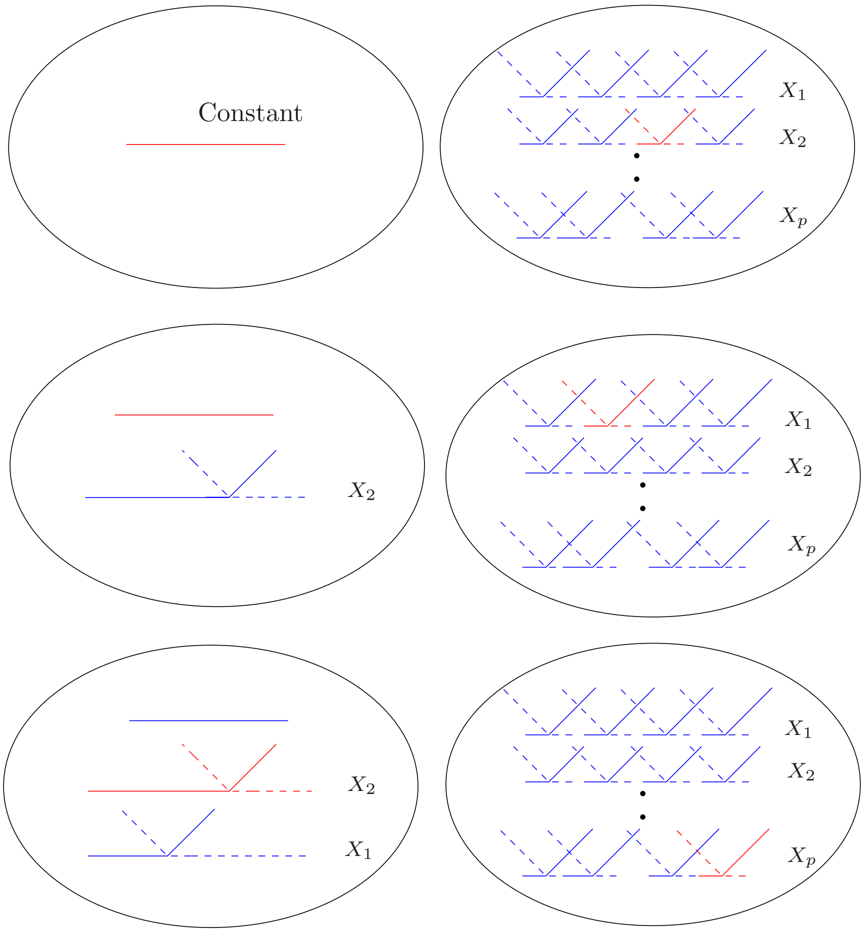

FIGURE 9.10. *Schematic of the MARS forward model-building procedure. On the left are the basis functions currently in the model: initially, this is the constant function $h(X) = 1$. On the right are all candidate basis functions to be considered in building the model. These are pairs of piecewise linear basis functions as in Figure 9.9, with knots t at all unique observed values x_{ij} of each predictor X_j. At each stage we consider all products of a candidate pair with a basis function in the model. The product that decreases the residual error the most is added into the current model. Above we illustrate the first three steps of the procedure, with the selected functions shown in red.*

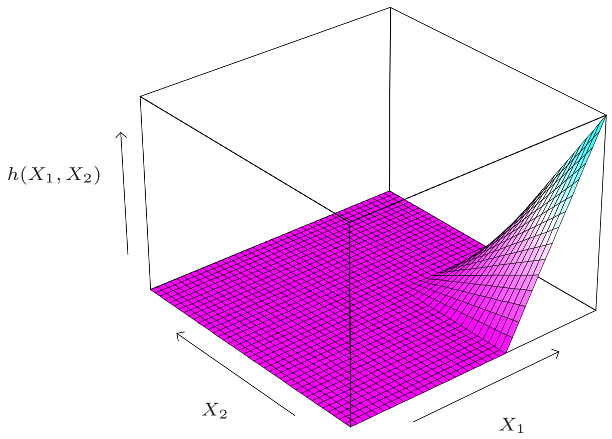

FIGURE 9.11. *The function $h(X_1, X_2) = (X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$, resulting from multiplication of two piecewise linear MARS basis functions.*

that produces the largest decrease in training error. Here $\hat{\beta}_{M+1}$ and $\hat{\beta}_{M+2}$ are coefficients estimated by least squares, along with all the other $M+1$ coefficients in the model. Then the winning products are added to the model and the process is continued until the model set $\mathcal{M}$ contains some preset maximum number of terms.

For example, at the first stage we consider adding to the model a function of the form $\beta_1(X_j - t)_+ + \beta_2(t - X_j)_+$; $t \in \{x_{ij}\}$, since multiplication by the constant function just produces the function itself. Suppose the best choice is $\hat{\beta}_1(X_2 - x_{72})_+ + \hat{\beta}_2(x_{72} - X_2)_+$. Then this pair of basis functions is added to the set $\mathcal{M}$, and at the next stage we consider including a pair of products the form

$$h_m(X) \cdot (X_j - t)_+ \quad \text{and} \quad h_m(X) \cdot (t - X_j)_+,\ t \in \{x_{ij}\},$$

where for h_m we have the choices

$$\begin{aligned} h_0(X) &= 1, \\ h_1(X) &= (X_2 - x_{72})_+, \text{ or} \\ h_2(X) &= (x_{72} - X_2)_+. \end{aligned}$$

The third choice produces functions such as $(X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$, depicted in Figure 9.11.

At the end of this process we have a large model of the form (9.19). This model typically overfits the data, and so a backward deletion procedure is applied. The term whose removal causes the smallest increase in residual squared error is deleted from the model at each stage, producing an estimated best model $\hat{f}_\lambda$ of each size (number of terms) λ. One could use cross-validation to estimate the optimal value of λ, but for computational savings the MARS procedure instead uses generalized cross-validation. This criterion is defined as

$$\mathrm{GCV}(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - M(\lambda)/N)^2}. \tag{9.20}$$

The value $M(\lambda)$ is the effective number of parameters in the model: this accounts both for the number of terms in the models, plus the number of parameters used in selecting the optimal positions of the knots. Some mathematical and simulation results suggest that one should pay a price of three parameters for selecting a knot in a piecewise linear regression.

Thus if there are r linearly independent basis functions in the model, and K knots were selected in the forward process, the formula is $M(\lambda) = r + cK$, where $c = 3$. (When the model is restricted to be additive—details below—a penalty of $c = 2$ is used). Using this, we choose the model along the backward sequence that minimizes $\mathrm{GCV}(\lambda)$.

Why these piecewise linear basis functions, and why this particular model strategy? A key property of the functions of Figure 9.9 is their ability to operate locally; they are zero over part of their range. When they are multiplied together, as in Figure 9.11, the result is nonzero only over the small part of the feature space where both component functions are nonzero. As a result, the regression surface is built up parsimoniously, using nonzero components locally—only where they are needed. This is important, since one should "spend" parameters carefully in high dimensions, as they can run out quickly. The use of other basis functions such as polynomials, would produce a nonzero product everywhere, and would not work as well.

The second important advantage of the piecewise linear basis function concerns computation. Consider the product of a function in $\mathcal{M}$ with each of the N reflected pairs for an input X_j. This appears to require the fitting of N single-input linear regression models, each of which uses $O(N)$ operations, making a total of $O(N^2)$ operations. However, we can exploit the simple form of the piecewise linear function. We first fit the reflected pair with rightmost knot. As the knot is moved successively one position at a time to the left, the basis functions differ by zero over the left part of the domain, and by a constant over the right part. Hence after each such move we can update the fit in $O(1)$ operations. This allows us to try every knot in only $O(N)$ operations.

The forward modeling strategy in MARS is hierarchical, in the sense that multiway products are built up from products involving terms already in the model. For example, a four-way product can only be added to the model if one of its three-way components is already in the model. The philosophy here is that a high-order interaction will likely only exist if some of its lower-order "footprints" exist as well. This need not be true, but is a reasonable working assumption and avoids the search over an exponentially growing space of alternatives.

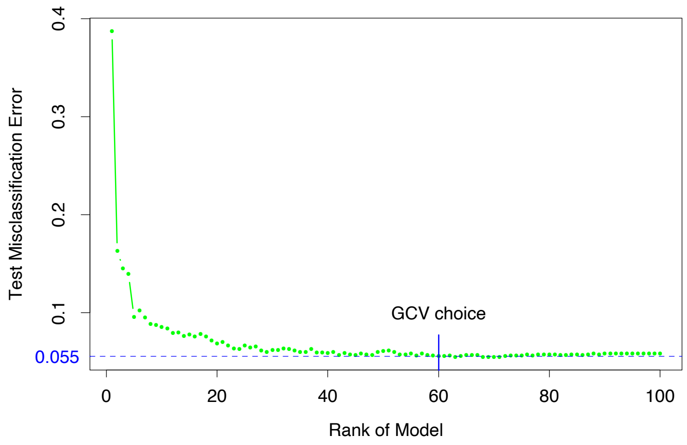

FIGURE 9.12. *Spam data: test error misclassification rate for the MARS procedure, as a function of the rank (number of independent basis functions) in the model.*

There is one restriction put on the formation of model terms: each input can appear at most once in a product. This prevents the formation of higher-order powers of an input, which increase or decrease too sharply near the boundaries of the feature space. Such powers can be approximated in a more stable way with piecewise linear functions.

A useful option in the MARS procedure is to set an upper limit on the order of interaction. For example, one can set a limit of two, allowing pairwise products of piecewise linear functions, but not three- or higher-way products. This can aid in the interpretation of the final model. An upper limit of one results in an additive model.

9.4.1 Spam Example (Continued)

We applied MARS to the "spam" data analyzed earlier in this chapter. To enhance interpretability, we restricted MARS to second-degree interactions. Although the target is a two-class variable, we used the squared-error loss function nonetheless (see Section 9.4.3). Figure 9.12 shows the test error misclassification rate as a function of the rank (number of independent basis functions) in the model. The error rate levels off at about 5.5%, which is slightly higher than that of the generalized additive model (5.3%) discussed earlier. GCV chose a model size of 60, which is roughly the smallest model giving optimal performance. The leading interactions found by MARS involved inputs (`ch$, remove`), (`ch$, free`) and (`hp, CAPTOT`). However, these interactions give no improvement in performance over the generalized additive model.

9.4.2 Example (Simulated Data)

Here we examine the performance of MARS in three contrasting scenarios. There are $N = 100$ observations, and the predictors $X_1, X_2, \ldots, X_p$ and errors ε have independent standard normal distributions.

Scenario 1: The data generation model is

$$Y = (X_1 - 1)_+ + (X_1 - 1)_+ \cdot (X_2 - .8)_+ + 0.12 \cdot \varepsilon. \tag{9.21}$$

The noise standard deviation 0.12 was chosen so that the signal-to-noise ratio was about 5. We call this the tensor-product scenario; the product term gives a surface that looks like that of Figure 9.11.

Scenario 2: This is the same as scenario 1, but with $p = 20$ total predictors; that is, there are 18 inputs that are independent of the response.

Scenario 3: This has the structure of a neural network:

$$\begin{aligned} \ell_1 &= X_1 + X_2 + X_3 + X_4 + X_5, \\ \ell_2 &= X_6 - X_7 + X_8 - X_9 + X_{10}, \\ \sigma(t) &= 1/(1 + e^{-t}), \\ Y &= \sigma(\ell_1) + \sigma(\ell_2) + 0.12 \cdot \varepsilon. \end{aligned} \tag{9.22}$$

Scenarios 1 and 2 are ideally suited for MARS, while scenario 3 contains high-order interactions and may be difficult for MARS to approximate. We ran five simulations from each model, and recorded the results.

In scenario 1, MARS typically uncovered the correct model almost perfectly. In scenario 2, it found the correct structure but also found a few extraneous terms involving other predictors.

Let $\mu(x)$ be the true mean of Y, and let

$$\begin{aligned} \text{MSE}_0 &= \text{ave}_{x \in \text{Test}} (\bar{y} - \mu(x))^2, \\ \text{MSE} &= \text{ave}_{x \in \text{Test}} (\hat{f}(x) - \mu(x))^2. \end{aligned} \tag{9.23}$$

These represent the mean-square error of the constant model and the fitted MARS model, estimated by averaging at the 1000 test values of x. Table 9.4 shows the proportional decrease in model error or R^2 for each scenario:

$$R^2 = \frac{\text{MSE}_0 - \text{MSE}}{\text{MSE}_0}. \tag{9.24}$$

The values shown are means and standard error over the five simulations. The performance of MARS is degraded only slightly by the inclusion of the useless inputs in scenario 2; it performs substantially worse in scenario 3.

TABLE 9.4. *Proportional decrease in model error (R^2) when MARS is applied to three different scenarios.*

Scenario	Mean (S.E.)
1: Tensor product $p = 2$	0.97 (0.01)
2: Tensor product $p = 20$	0.96 (0.01)
3: Neural network	0.79 (0.01)

9.4.3 Other Issues

MARS for Classification

The MARS method and algorithm can be extended to handle classification problems. Several strategies have been suggested.

For two classes, one can code the output as 0/1 and treat the problem as a regression; we did this for the `spam` example. For more than two classes, one can use the indicator response approach described in Section 4.2. One codes the K response classes via 0/1 indicator variables, and then performs a multi-response MARS regression. For the latter we use a common set of basis functions for all response variables. Classification is made to the class with the largest predicted response value. There are, however, potential masking problems with this approach, as described in Section 4.2. A generally superior approach is the "optimal scoring" method discussed in Section 12.5.

Stone et al. (1997) developed a hybrid of MARS called PolyMARS specifically designed to handle classification problems. It uses the multiple logistic framework described in Section 4.4. It grows the model in a forward stagewise fashion like MARS, but at each stage uses a quadratic approximation to the multinomial log-likelihood to search for the next basis-function pair. Once found, the enlarged model is fit by maximum likelihood, and the process is repeated.

Relationship of MARS to CART

Although they might seem quite different, the MARS and CART strategies actually have strong similarities. Suppose we take the MARS procedure and make the following changes:

- Replace the piecewise linear basis functions by step functions $I(x-t > 0)$ and $I(x - t \leq 0)$.
- When a model term is involved in a multiplication by a candidate term, it gets replaced by the interaction, and hence is not available for further interactions.

With these changes, the MARS forward procedure is the same as the CART tree-growing algorithm. Multiplying a step function by a pair of reflected