SLAC PUB-4960 Rev
Tech Report 102 Rev†
August 1990
M

MULTIVARIATE ADAPTIVE REGRESSION SPLINES*

Jerome H. Friedman,

Stanford Linear Accelerator Center
and
Department of Statistics
Stanford University
Stanford, California 94309

ABSTRACT

A new method is presented for flexible regression modeling of high dimensional data. The model takes the form of an expansion in product spline basis functions, where the number of basis functions as well as the parameters associated with each one (product degree and knot locations) are automatically determined by the data. This procedure is motivated by the recursive partitioning approach to regression and shares its attractive properties. Unlike recursive partitioning, however, this method produces continuous models with continuous derivatives. It has more power and flexibility to model relationships that are nearly additive or involve interactions in at most a few variables. In addition, the model can be represented in a form that separately identifies the additive contributions and those associated with the different multivariable interactions.

(Submitted to *The Annals of Statistics*)

*This work was supported in part by the Department of Energy under contract DE-AC03-76SF00515, and the National Security Agency under contract MDA904-88-H-2029.
† Also published as Dept. of Statistics Tech. Report 102 Rev, August 1990.

1.0. Introduction

A problem common to many disciplines is that of adequately approximating a function of several to many variables, given only the value of the function (often perturbed by noise) at various points in the dependent variable space. Research on this problem occurs in applied mathematics (multivariate function approximation), statistics (nonparametric multiple regression), and in computer science and engineering (statistical learning "neural" networks). The goal is to model the dependence of a response variable y on one or more predictor variables $x_1, \cdots, x_n$ given realizations (data) $\{y_i, x_{1i}, \cdots, x_{ni}\}_1^N$. The system that generated the data is presumed to be described by

$$y = f(x_1, \cdots, x_n) + \epsilon \tag{1}$$

over some domain $(x_1, \cdots, x_n) \in D \subset R^n$ containing the data. The single valued deterministic function f, of its n-dimensional argument, captures the joint predictive relationship of y on $x_1, \cdots, x_n$. The additive stochastic component ϵ, whose expected value is defined to be zero, usually reflects the dependence of y on quantities other than $x_1, \cdots, x_n$ that are neither controlled nor observed. The aim of regression analysis is to use the data to construct a function $\hat{f}(x_1, \cdots, x_n)$ that can serve as a reasonable approximation to $f(x_1, \cdots, x_n)$ over the domain D of interest.

The notion of reasonableness depends on the purpose for which the approximation is to be used. In nearly all applications however accuracy is important. Lack of accuracy is often defined by the integral error

$$I = \int_D w(\mathbf{x}) \Delta[\hat{f}(\mathbf{x}), f(\mathbf{x})] d\mathbf{x} \tag{2}$$

or the expected error

$$E = \frac{1}{N} \sum_{i=1}^{N} w(\mathbf{x}_i) \Delta[\hat{f}(\mathbf{x}_i), f(\mathbf{x}_i)]. \tag{3}$$

Here $\mathbf{x} = (x_1, \cdots, x_n)$, Δ is some measure of distance, and $w(\mathbf{x})$ is a possible weight function. The integral error (2) characterizes the average accuracy of the approximation over the entire domain of interest whereas the expected error (3) reflects average accuracy only on the design points $\mathbf{x}_1, \cdots, \mathbf{x}_N$. In high dimensional settings especially, low integral error is generally much more difficult to achieve than low expected error.

If the sole purpose of the regression analysis is to obtain a rule for predicting future values of the response y, given values for the covariates $(x_1, \cdots, x_n)$, then accuracy is the only important virtue of the model. If future joint covariate values $\mathbf{x}$ can only be realized at the design points $\mathbf{x}_1, \cdots, \mathbf{x}_N$ (with probabilities $w(\mathbf{x}_i)$) then the expected error (3) is the appropriate measure; otherwise the integral error (2) is more relevant. Often, however, one wants to use $\hat{f}$ to try to understand the properties of the true underlying function f (1) and thereby the system that generated the data. In this case the interpretability of the representation of the model is also very important. Depending on the application, other desirable properties of the approximation might include rapid computability and smoothness; that is $\hat{f}$ be a smooth function of its n-dimensional argument and at least its low order derivatives exist everywhere in D.

This paper presents a new method of flexible nonparametric regression modeling that attempts to meet the objectives outlined above. It appears to have the potential to be a substantial improvement over existing methodology in settings involving moderate sample sizes, $50 \leq N \leq 1000$, and moderate to high dimension, $3 \leq n \leq 20$. It can be viewed as either a generalization of the recursive partitioning regression strategy (Morgan and Sonquist, 1963, and Breiman, Friedman, Olshen, and Stone, 1984), or as a generalization of the additive modeling approach of Friedman and Silverman (1989). Its immediate ancestor is discussed in Friedman (1988). Although the procedure described here is somewhat different from that in Friedman (1988), the two procedures have a lot in common and much of the associated discussion of that earlier procedure is directly relevant to the one described here. Some of this common discussion material is therefore repeated in this paper for completeness.

2.0. Existing Methodology

This section provides a brief overview of some existing methodology for multivariate regression modeling. The intent here is to highlight some of the difficulties associated with each of the methods when applied in high dimensional settings in order to motivate the new procedure described later. It should be borne in mind however that many of these methods have met with considerable success in a variety of applications.

2.1. Global Parametric Modeling.

Function approximation in high dimensional settings has (in the past) been pursued mainly in statistics. The principal approach has been to fit (usually a simple) parametric function $g(\mathbf{x} \mid \{a_j\}_1^p)$ to the training data most often by least-squares. That is

$$\hat{f}(\mathbf{x}) = g(\mathbf{x} \mid \{\hat{a}_j\}_1^p) \tag{4}$$

where the parameter estimates are given by

$$\{\hat{a}_j\}_1^p = \underset{\{a_j\}_1^p}{\operatorname{argmin}} \sum_{i=1}^{N} [y_i - g(\mathbf{x} \mid \{a_j\}_1^p)]^2.$$

The most commonly used parametrization is the linear function

$$g(\mathbf{x} \mid \{a_j\}_0^p) = a_0 + \sum_{i=1}^{p} a_i x_i, \quad p \leq n. \tag{5}$$

Sometimes additional terms, that are preselected functions of the original variables (such as polynomials) are also included in the model. This parametric approach has limited flexibility and is likely to produce accurate approximations only when the form of the true underlying function $f(\mathbf{x})$ (1) is close to the prespecified parametric one (4). On the other hand, simple parametric models have the virtue of requiring relatively few data points, they are easy to interpret, and rapidly computable. If the stochastic component ϵ (1) is large compared to $f(\mathbf{x})$, then the systematic error associated with model misspecification may not be the most serious problem.

2.2. Nonparametric Modeling.

In low dimensional settings ($n \lesssim 2$) global parametric modeling has been successfully generalized using three (related) paradigms – piecewise and local parametric fitting and roughness penalty methods. The basic idea of piecewise parametric fitting is to approximate f by several simple parametric functions (usually low order polynomials) each defined over a different subregion of the domain D. The approximation is constrained to be everywhere continuous, and sometimes have continuous low order derivatives as well. The tradeoff between smoothness and flexibility of the approximation $\hat{f}$ is controlled by the number of subregions (knots) and the lowest order derivative allowed to be discontinuous at subregion boundaries. The most popular piecewise polynomial fitting procedures are based on splines, where the parametric functions are taken to be polynomials of degree q and derivatives to order $q-1$ are required to be continuous ($q = 3$ is the most popular choice). The procedure is implemented by constructing a set of (globally defined) basis functions that span the space of qth order spline approximations, and fitting the coefficients of the basis function expansion to the data by ordinary least-squares. For example, in the univariate case ($n = 1$) with $K+1$ regions delineated by K points on the real line ("knots"), one such basis is represented by the functions

$$1,\ \{x^j\}_1^q,\quad \{(x-t_k)_+^q\}_1^K \tag{6}$$

where $\{t_k\}_1^K$ are the knot locations. (Here the subscript "+" indicates a value of zero for negative values of the argument.) This is known as the "truncated power" basis and is one of many that span the space of q-degree spline functions of dimension $K+q+1$. [See deBoor (1978) for a general review of splines and Shumacker (1976), (1984) for reviews of some two-dimensional ($n = 2$) extensions.]

The direct extension of piecewise parametric modeling to higher dimensions ($n > 2$) is straightforward in principle but difficult in practice. These difficulties are related to the so called "curse-of-dimensionality," a phrase coined by Bellman (1961) to express the fact that exponentially increasing numbers of (data) points are needed to densely populate Euclidean spaces of increasing dimension. In the case of spline approximations the subregions are usually constructed as tensor products of $K+1$ intervals (defined by K knots) over the n variables. The corresponding global basis is the tensor product over the $K+q+1$ basis functions associated with each variable (6). This gives rise to $(K+q+1)^n$ coefficients to be estimated from the data. Even with a very coarse grid (small K), a very large data sample is required.

Local parametric approximations ("smoothers") take the form

$$\hat{f}(\mathbf{x}) = g(\mathbf{x} \mid \{\hat{a}_j(\mathbf{x})\}_1^p)$$

where g is a simple parameric function (4). Unlike global parametric approximations, here the parameter values are generally different at each evaluation point $\mathbf{x}$ and are obtained by locally weighted least-squares fitting

$$\{\hat{a}_j(\mathbf{x})\}_1^p = \underset{\{a_j\}_1^p}{\operatorname{argmin}} \sum_{i=1}^{N} w(\mathbf{x}, \mathbf{x}_i)[y_i - g(\mathbf{x}_i \mid \{a_j\}_1^p)]^2. \tag{7}$$

The weight function $w(\mathbf{x}, \mathbf{x}')$ (of $2n$ variables) is chosen to place the dominant mass on points $\mathbf{x}'$ "close" to $\mathbf{x}$. The properties of the approximation are mostly determined by the choice of w and to a lesser extent by the particular parametric function g used. The most commonly studied g is the simple constant $g(\mathbf{x} \mid a) = a$ [Parzen (1962), Shepard (1964), Bonzzini and Lenarduzzi (1985)]. Cleveland (1979) suggested that local linear fitting (5) produces superior results, especially near the edges, and Cleveland and Devlin (1988) suggest local fitting of quadratic functions. Stone (1977) shows that, asymptotically, higher order polynomials can have superior convergence rates when used with simple weight functions (see below), depending on the continuity of properties of f (1).

The difficulty with applying local parametric methods in higher dimensions lies with the choice of an appropriate weight function w (7) for the problem at hand. This strongly depends on f (1) and thus is generally unknown. Asymptotically any weight function that places dominant mass in a (shrinking) convex region centered at $\mathbf{x}$ will work. This motivates the most common choice

$$w(\mathbf{x}, \mathbf{x}') = K(|\mathbf{x} - \mathbf{x}'|/s(\mathbf{x})) \tag{8}$$

with $|\mathbf{x} - \mathbf{x}'|$ being a (possibly) weighted distance between $\mathbf{x}$ and $\mathbf{x}'$, $s(\mathbf{x})$ is a scale factor ("bandwidth"), and K is a ("kernel") function of a single argument.The kernel is usually chosen so that its absolute value decreases with increasing value of its argument. Commonly used scale functions are a constant $s(\mathbf{x}) = s_0$ ("kernel" smoothing) or $s(\mathbf{x}) = s_0/\hat{p}(\mathbf{x})$ ("near neighbor" smoothing), where $\hat{p}(\mathbf{x})$ is some estimate of the local density of the design points. In low dimensional ($n \lesssim 2$) settings, this approximation of the weight function w of $2n$ variables by a function K of a single variable (8), controlled by a single parameter (s_0), is generally not too serious since asymptotic conditions can be realized without requiring gargantuan sample sizes. This is not the case in higher dimensions. The problem with a kernel based on interpoint distance (8) is that the volume of the corresponding sphere in n-space grows as its radius to the nth power. Therefore to ensure that w (8) places adequate mass on enough data points to control the variance of $\hat{f}(\mathbf{x})$, the bandwidth $s(\mathbf{x})$ will necessarily have to be very large, incurring high bias.

Roughness penalty approximations are defined by

$$\hat{f}(\mathbf{x}) = \underset{g}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} [y_i - g(\mathbf{x}_i)]^2 + \lambda R(g) \right\}.$$

Here $R(g)$ is a functional that increases with increasing "roughness" of the function $g(\mathbf{x})$. The minimization is performed over all g for which $R(g)$ is defined. The parameter λ regulates the tradeoff between the roughness of g and its fidelity to the data. The most studied roughness penalty is the integrated squared Laplacian

$$R(g) = \sum_{k=1}^{n} \sum_{\ell=1}^{n} \int \left| \frac{\partial^2 g}{\partial x_k \partial x_\ell} \right|^2 d\mathbf{x} \tag{9}$$

leading to Laplacian smoothing ("thin-plate") spline approximations for $n \leq 3$. For $n > 3$ the general thin-plate spline penalty has a more complex form involving derivatives of higher order than

two. [See Wahba (1990), Section 2.4).] The properties of roughness penalty methods are similar to those of kernel methods (7) (8), using an appropriate kernel function K (8) with λ regulating the bandwidth $s(\mathbf{x})$. They therefore encounter the same basic limitations in high dimensional settings.

2.3. Low Dimensional Expansions.

The ability of the nonparametric methods to often adequately approximate functions of a low dimensional argument, coupled with their corresponding inability in higher dimensions, has motivated approximations that take the form of expansions in low dimensional functions

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{J} \hat{g}_j(\mathbf{z}_j). \tag{10}$$

Here each $\mathbf{z}_j$ is comprised of a small (different) preselected subset of $\{x_1, \cdots, x_n\}$. Thus, a function of an n-dimensional argument is approximated by J functions, each of a low ($\lesssim 2$) dimensional argument. [Note that any (original) variable may appear in more than one subset $\mathbf{z}_j$. Extra conditions (such as orthogonality) can be imposed to resolve any identifiability problems.] After selecting the variable subsets $\{\mathbf{z}_j\}_1^J$, the corresponding functions estimates $\{\hat{g}_j(\mathbf{z}_j)\}_1^J$ are obtained by nonparametric methods, for example, using least-squares

$$\{\hat{g}_j(\mathbf{z}_j)\}_1^J = \underset{\{g_j\}}{\operatorname{argmin}} \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{J} g_j(\mathbf{z}_{ij}) \right]^2 \tag{11}$$

with smoothness constraints imposed on the $\hat{g}_j$ through the particular nonparametric method used to estimate them.

In the case of piecewise polynomials (splines) a corresponding basis is constructed for each individual $\mathbf{z}_j$ and the solution is obtained as a global least-squares fit of the response y on the union of all such basis functions [Stone and Koo (1985)]. With roughness penalty methods the formulation becomes

$$\hat{f}(\mathbf{x}) = \underset{\{g_j\}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{J} g_j(\mathbf{z}_{ij})) \right]^2 + \sum_{j=1}^{J} \lambda_j R(g_j) \right\}. \tag{12}$$

These are referred to as "interaction splines" [Barry (1986), Wahba (1986), Gu, Bates, Chen and Wahba (1990), Gu and Wahba (1988), and Chen, Gu and Wahba (1989)].

Any low dimensional nonparametric function estimator can be used in conjunction with the "backfitting" algorithm to solve (11) [Friedman and Stuetzle (1981), Breiman and Friedman (1985), and Buja, Hastie and Tibshirani (1989)]. The procedure iteratively reestimates $\hat{g}_j(\mathbf{z}_j)$ by

$$\hat{g}_j(\mathbf{z}_j) \leftarrow \underset{g_j}{\operatorname{argmin}} \sum_{i=1}^{N} \left[\left(y_i - \sum_{k \neq j} g_k(\mathbf{z}_{ik}) \right) - g_j(\mathbf{z}_{ij}) \right]^2$$

until convergence. Smoothness is imposed on the $\hat{g}_j$ by the particular estimator employed. For example, if each iterated function estimate is obtained using Laplacian smoothing splines (9) with parameter λ_j (12), then the backfitting algorithm produces the solutions to (12). [See Buja, Hastie and Tibshirani (1989)]. Hastie and Tibshirani (1986) generalize the backfitting algorithm to obtain solutions for criteria other than squared-error loss.

The most extensively studied low dimensional expansion has been the additive model

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{n} \hat{g}_j(x_j) \tag{13}$$

since nonadaptive smoothers work best for one dimensional functions and there are only n of them (at most) that can enter. Also, in many applications the true underlying function f (1) can be approximated fairly well by an additive function.

Nonparametric function estimation based on low dimensional expansions is an important step forward and (especially for additive modeling) has met with considerable practical success (see references above). As a general method for estimating functions of many variables, this approach has some limitations. The abilities of nonadaptive nonparametric smoothers generally limit the expansion functions to low dimensionality. Performance (and computational) considerations limit the number of low dimensional functions to a small subset of all those that could potentially be entered. For example, there are $n(n+1)/2$ possible univariate and bivariate functions. A good subset will depend on the true underlying function f (1) and is often unknown. Also, each expansion function has a corresponding smoothing parameter causing the entire procedure to be defined by many such parameters. A good set of values for all these parameters is seldom known for any particular application since they depend on f (1). Automatic selection based on minimizing a model selection criterion generally requires a multiparameter numerical optimization which is inherently difficult and computationally consuming. Also the properties of estimates based on the simultaneous estimation of a large number of smoothing parameters are largely unknown, although progress is being made [see Gu and Wahba (1988)].

2.4. Adaptive Computation.

Strategies that attempt to approximate general functions in high dimensionality are based on adaptive computation. An adaptive computation is one that dynamically adjusts its strategy to take into account the behavior of the particular problem to be solved, e.g. the behavior of the function to be approximated. Adaptive algorithms have been in long use in numerical quadrature [see Lyness (1970); Friedman and Wright (1981).] In statistics, adaptive algorithms for function approximation have been developed based on two paradigms, recursive partitioning [Morgan and Sonquist (1963), Breiman, et al. (1984)], and projection pursuit [Friedman and Stuetzle (1981), Friedman, Grosse, and Stuetzle (1983), and Friedman, (1985)].

2.4.1. Projection Pursuit Regression.

Projection pursuit uses an approximation of the form

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} f_m \left(\sum_{i=1}^{n} \alpha_{im} x_i \right), \tag{14}$$

that is, additive functions of linear combinations of the variables. The univariate functions, f_m, are required to be smooth but are otherwise arbitrary. These functions, and the corresponding coefficients of the linear combinations appearing in their arguments, are jointly optimized to produce a good fit to the data based on some distance (between functions) criterion – usually squared-error loss. Projection pursuit regression can be viewed as a low dimensional expansion method where the (one dimensional) arguments are not prespecified, but instead are adjusted to best fit the data. It can be shown [see Diaconis and Shahshahani (1984)] that any smooth function of n variables can be represented by (14) for large enough M. The effectiveness of the approach lies in the fact that even for small to moderate M, many classes of functions can be closely fit by approximations of this form [see Donoho and Johnstone (1989).] Another advantage of projection pursuit approximations is affine equivariance. That is, the solution is invariant under any nonsingular affine transformation (rotation and scaling) of the original explanatory variables. It is the only general method suggested for practical use that seems to possess this property. Projection pursuit solutions have some interpretive value (for small M) in that one can inspect the solution functions f_m and the corresponding loadings in the linear combination vectors. Evaluation of the resulting approximation is computationally fast. Disadvantages of the projection pursuit approach are that there exist some simple functions that require large M for good approximation [see Huber (1985)], it is difficult to separate the additive from the interaction effects associated with the variable dependencies, interpretation is difficult for large M, and the approximation is computationally time consuming to construct.

2.4.2 Recursive Partitioning Regression

The recursive partitioning regression model takes the form

$$\text{if } \mathbf{x} \in R_m, \text{ then } \hat{f}(\mathbf{x}) = g_m(\mathbf{x}|\{a_j\}_1^p). \tag{15}$$

Here $\{R_m\}_1^M$ are disjoint subregions representing a partition of D. The functions g_m are generally taken to be of quite simple parametric form. The most common is a constant function

$$g_m(\mathbf{x}|a_m) = a_m \tag{16}$$

[Morgan and Sunquist (1963) and Breiman, et al. (1984)]. Linear functions (5) have also been proposed [Breiman and Meisel (1976) and Friedman (1979)], but they have not seen much use (see below). The goal is to use the data to simultaneously estimate a good set of subregions and the parameters associated with the separate functions in each subregion. Continuity at subregion boundaries is not enforced.

The partitioning is accomplished through the recursive splitting of previous subregions. The starting region is the entire domain D. At each stage of the partitioning all existing subregions are each optimally split into two (daughter) subregions. The eligible splits of a region R into two daughter regions R_ℓ and R_r take the form

$$\begin{aligned}&\text{if } \mathbf{x} \in R \text{ then}\\ &\quad \text{if } x_v \leq t \text{ then } \mathbf{x} \in R_\ell\\ &\quad\quad \text{else } \mathbf{x} \in R_r\\ &\text{end if.}\end{aligned}$$

Here v labels one of the covariates and t is a value on that variable. The split is jointly optimized over $1 \leq v \leq n$ and $-\infty \leq t \leq \infty$ using a goodness-of-fit criterion on the resulting approximation (15). This procedure generates hyperrectangular axis oriented subregions. The recursive subdivision is continued until a large number of subregions are generated. The subregions are then recombined in a reverse manner until an optimal set is reached, based on a criterion that penalizes both for lack-of-fit and increasing number of regions (see Breiman et al., 1984).

Recursive partitioning is a powerful paradigm, especially if the simple piecewise constant approximation (16) is used. It has the ability to exploit low "local" dimensionality of functions. That is, even though the function f (1) may strongly depend on a large number of variables globally, in any local region the dependence is strong on only a few of them. These few variables may be different in different regions. This ability comes from the recursive nature of the partitioning which causes it to become more and more local as the splitting proceeds. Variables that locally have less influence on the response are less likely to be used for splitting. This gives rise to a local variable subset selection. Global variable subset selection emerges as a natural consequence. Recursive partitioning (15) based on linear functions (5) basically lacks this (local) variable subset selection feature. This tends to limit its power (and interpretability) and is probably the main reason contributing to its lack of popularity.

Another property that recursive partitioning regression exploits is the marginal consequences of interaction effects. That is, a local intrinsic dependence on several variables, when best approximated by an additive function (13), does not lead to a constant model. This is nearly always the case.

Recursive partitioning models using piecewise constant approximations (15) (16) are fairly interpretable owing to the fact that they are very simple and can be represented by a binary tree. [See Breiman et al. (1984) and Section 3.1 below.] They are also fairly rapid to construct and especially rapid to evaluate.

Although recursive partitioning is the most adaptive of the methods for multivariate function approximation it suffers from some fairly severe restrictions that limit its effectiveness. Foremost among these is that the approximating function is discontinuous at the subregion boundaries. This is more than a cosmetic problem. It severely limits the accuracy of the approximation, especially when the true underlying function is continuous. Even imposing continuity only of the function

(as opposed to derivatives of low order) is usually enough to dramatically increase approximation accuracy.

Another problem with recursive partitioning is that certain types of simple functions are difficult to approximate. These include linear functions with more than a few nonzero coefficients [with the piecewise constant approximation (16)] and additive functions (13) in more than a few variables (piecewise constant or piecewise linear approximation). More generally, it has difficulty when the dominant interactions involve a small fraction of the total number of variables. In addition, one cannot discern from the representation of the model whether the approximating function is close to a simple one, such as linear or additive, or whether it involves complex interactions among the variables.

3.0. Adaptive Regression Splines

This section describes the multivariate adaptive regression spline (MARS) approach to multivariate nonparametric regression. The goal of this procedure is to overcome some of the limitations associated with existing methodology outlined above. It is most easily understood through its connections with recursive partitioning regression. It will therefore be developed here as a series of generalizations to that procedure.

3.1. Recursive Partitioning Regression Revisited.

Recursive partitioning regression is generally viewed as a geometrical procedure. This framework provides the best intuitive insight into its properties, and was the point of view adopted in Section 2.4.2. It can however also be viewed in a more conventional light as a stepwise regression procedure. The idea is to produce an equivalent model to (15) (16) by replacing the geometrical concepts of regions and splitting with the arithmetic notions of adding and multiplying.

The starting point is to cast the approximation (15) (16) in the form of an expansion in a set of basis functions

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} a_m B_m(\mathbf{x}). \tag{17}$$

The basis functions B_m take the form

$$B_m(\mathbf{x}) = I[\mathbf{x} \in R_m] \tag{18}$$

where I is an indicator function having the value one if its argument is true and zero otherwise. The $\{a_m\}_1^M$ are the coefficients of the expansion whose values are jointly adjusted to give the best fit to the data. The $\{R_m\}_1^M$ are the same subregions of the covariate space as in (15) (16). Since these regions are disjoint only one basis function is nonzero for any point $\mathbf{x}$ so that (17) (18) is equivalent to (15) (16).

The aim of recursive partitioning is not only to adjust the coefficient values to best fit the data, but also to derive a good set of basis functions (subregions) based on the data at hand. Let $H[\eta]$ be a step function indicating a positive argument

$$H[\eta] = \begin{cases} 1 & \text{if } \eta \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

and let $LOF(g)$ be a procedure that computes the lack-of-fit of a function $g(\mathbf{x})$ to the data. Then the forward stepwise regression procedure presented in Algorithm 1 is equivalent to the recursive partitioning strategy outlined in Section 2.4.2.

Algorithm 1: (recursive partitioning)

$B_1(\mathbf{x}) \leftarrow 1$
For $M = 2$ to $M_{\max}$ do: $lof^* \leftarrow \infty$
  For $m = 1$ to $M - 1$ do:
    For $v = 1$ to n do:
      For $t \in \{x_{vj} | B_m(\mathbf{x}_j) > 0\}$
        $g \leftarrow \sum_{i \neq m} a_i B_i(\mathbf{x}) + a_m B_m(\mathbf{x}) H[+(x_v - t)] + a_M B_m(\mathbf{x}) H[-(x_v - t)]$
        $lof \leftarrow \min_{a_1, \ldots, a_M} LOF(g)$
        if $lof < lof^*$ then $lof^* \leftarrow lof$; $m^* \leftarrow m$; $v^* \leftarrow v$; $t^* \leftarrow t$ end if
      end for
    end for
  end for
  $B_M(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x}) H[-(x_{v^*} - t^*)]$
  $B_{m^*}(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x}) H[+(x_{v^*} - t^*)]$
end for
end algorithm

The first line in Algorithm 1 is equivalent to setting the initial region to the entire domain. The first For-loop iterates the "splitting" procedure with $M_{\max}$ being the final number of regions (basis functions). The next three (nested) loops perform an optimization to select a basis function B_{m^*} (already in the model), a predictor variable x_{v^*} and a "split point" t^*. The quantity being minimized is the lack-of-fit of a model with B_{m^*} being replaced by its product with the step function $H[+(x_{v^*} - t^*)]$, and with the addition of a new basis function which is the product of B_{m^*} and the reflected step function $H[-(x_{v^*} - t^*)]$. This is equivalent to splitting the corresponding region R_{m^*} on variable v^* at split point t^*. Note that the minimization of $LOF(g)$ with respect to the expansion coefficients (line 7) is a linear regression of the response on the current basis function set.

The basis functions produced by Algorithm 1 have the form

$$B_m(\mathbf{x}) = \prod_{k=1}^{K_m} H[s_{km} \cdot (x_{v(k,m)} - t_{km})]. \tag{20}$$

The quantity K_m is the number of "splits" that gave rise to B_m, whereas the arguments of the step functions contain the parameters associated with each of these splits. The quantities s_{km} in (20) take on values ± 1, and indicate the (right/left) sense of the associated step function. The $v(k, m)$ label the predictor variables and the t_{km} represent values on the corresponding variables. Owing to the forward stepwise (recursive) nature of the procedure the parameters for all the basis functions

can be represented on a binary tree that reflects the partitioning history (see Breiman et al., 1984). Figure 1 shows a possible result of running Algorithm 1 in this binary tree representation, along with the corresponding basis functions. The internal nodes of the binary tree represent the step functions and the terminal nodes represent the final basis functions. Below each internal node are listed the variable v and location t associated with the step function represented by that node. The sense of the step function s is indicated by descending either left or right from the node. Each basis function (20) is the product of the step functions encountered in a traversal of the tree starting at the root and ending at its corresponding terminal node.

With most forward stepwise regression procedures it makes sense to follow them by a backwards stepwise procedure to remove basis functions that no longer contribute sufficiently to the accuracy of the fit. This is especially true in the case of recursive partitioning. In fact the strategy here is to deliberately overfit the data with an excessively large model, and then to trim it back to proper size with a backwards stepwise strategy (see Breiman, et al., 1984).

In the case of recursive partitioning the usual straightforward "one at a time" stepwise term (basis function) deletion strategy does not work. Each basis function represents a disjoint subregion and removing it leaves a hole in the predictor variable space within which the model will predict a zero response value. Therefore it is unlikely that any term (basis function) can be removed without seriously degrading the quality of the fit. To overcome this a backward stepwise strategy for recursive partitioning models must delete (sibling) regions in adjacent pairs by merging them into a single (parent) region in roughly the inverse splitting order. One must delete splits rather than regions (basis functions) in the backwards stepwise strategy. One method for doing this is the optimal complexity tree pruning algorithm described in Breiman et al. (1984).

3.2. Continuity.

As noted in Section 2.4.2, a fundamental limitation of recursive partitioning models is lack of continuity. The models produced by (15) (16) are piecewise constant and sharply discontinuous at subregion boundaries. This lack of continuity severely limits the accuracy of the approximation. It is possible, however, to make a minor modification to Algorithm 1 which will cause it to produce continuous models with continuous derivatives.

The only aspect of Algorithm 1 that introduces discontinuity into the model is the use of the step function (19) as its central ingredient. If the step function were replaced by a continuous function of the same argument everywhere it appears (lines 6, 12, and 13), Algorithm 1 would produce continuous models. The choice for a continuous function to replace the step function (19) is guided by the fact that the step function as used in Algorithm 1 is a special case of a spline basis function (6).

The one-sided truncated power basis functions for representing qth order splines are

$$b_q(x-t) = (x-t)_+^q$$

where t is the knot location, q is the order of the spline, and the subscript indicates the positive part of the argument. For $q > 0$ the spline approximation is continuous and has $q-1$ continuous

derivatives. A two-sided truncated power basis is a mixture of functions of the form

$$b_q^{\pm}(x-t) = [\pm(x-t)]_+^q. \tag{21}$$

The step functions appearing in Algorithm 1 are seen to be two-sided truncated power basis functions for $q = 0$ splines.

The usual method for generalizing spline fitting to higher dimensions is to employ basis functions that are tensor products of univariate spline functions (see Section 2.2). Using the two-sided truncated power basis for the univariate functions, these multivariate spline basis functions take the form

$$B_m^{(q)}(\mathbf{x}) = \prod_{k=1}^{K_m} [s_{km} \cdot (x_{v(k,m)} - t_{km})]_+^q, \tag{22}$$

along with products involving the truncated power functions with polynomials of lower order than q. (Note that $s_{km} = \pm 1$.) Comparing (20) with (22) we see that the basis functions (20) produced by recursive partitioning are a subset of a complete tensor product ($q = 0$) spline basis with knots at every (distinct) marginal data point value. Thus, recursive partitioning can be viewed as a forward/backward stepwise regression procedure for selecting a (relatively very small) subset of regressor functions from this (very large) complete basis.

Although replacing the step function (19) by a $q > 0$ truncated power spline basis function (21) in Algorithm 1 will produce continuous models (with $q - 1$ continuous derivatives), the resulting basis will not reduce to a set of tensor product spline basis functions (as was the case for $q = 0$). Algorithm 1 permits multiple splitting on the same variable along a single path of the binary tree (see Fig. 1). Therefore the final basis functions can each have several factors involving the same variable in their product. For $q > 0$ this gives rise to dependencies of higher power than q on individual variables. Products of univariate spline functions on the same variable do not give rise to a (univariate) spline function of the same order, except for the special case of $q = 0$ (21). Each factor in a tensor product spline basis function must involve a different variable and thereby cannot produce dependencies on individual variables of power greater than q. Owing to the many desirable properties of splines for function approximation (de Boor, 1978) it would be nice for a continuous analog of recursive partitioning to also produce them. Since permitting repeated (nested) splits on the same variable is an essential aspect contributing to the power of recursive partitioning, we cannot simply prohibit it in a continuous generalization. A natural resolution to this dilemma emerges from the considerations in Section 3.3.

3.3. A Further Generalization.

Besides lack of continuity, another problem that plagues recursive partitioning regression models is their inability to provide good approximations to certain classes of simple often occurring functions. These are functions that either have no strong interaction effects, or strong interactions each involving at most a few of the predictor variables. Linear (5) and additive (13) functions are among those in this class. From the geometric point of view, this can be regarded as a limitation

of the axis-oriented hyperrectangular shape of the generated regions. These difficult functions (for recursive partitioning) have isopleths that tend to be oriented at oblique angles to the coordinate axes, thereby requiring a great many axis oriented hyperrectangular regions to capture the functional dependence.

One can also understand this phenomenon by viewing recursive partitioning as a stepwise regression procedure (Algorithm 1). It is in this framework that a natural solution to this problem emerges. The goal, in this context, is to find a good set of basis functions for the approximation. The final model is then obtained by projecting the data onto this basis. The bias associated with this procedure is just the average distrance of the true underlying function f (1) from its projection onto the space spanned by the derived basis functions. The variance of the model estimate is directly proportional to the dimensionality of this space, namely the number of basis functions used. In order to achieve good accuracy (small bias and variance) one must derive a small set of basis functions that are close to the true underlying function in the above sense (small bias).

The problem with the basis derived through recursive partitioning (20), or the continuous analog (22), is that it tends to mostly involve functions of more than a few variables (higher order interactions). Each execution of the outer loop in Algorithm 1 (split) removes a basis function of lower interaction order, and replaces it by two functions, each with interaction order one level higher, unless it happens to split on a variable already in the product. Thus, as the partitioning proceeds the average interaction level of the basis function set steadily increases. One simple consequence is that recursive partitioning cannot produce an additive model in more than one variable. The overriding effect is that such a basis involving high order interactions among the variables cannot provide a good approximation to functions with at most low order interactions, unless a large number of basis functions are used. This is the regression analog of trying to approximate with rectangular regions, functions that have isopleths oblique to the axes.

As noted in Section 3.2, recursive partitioning ($q = 0$) can be regarded as a stepwise procedure for selecting a small subset of basis functions from a very large complete tensor product spline basis. The problem is that all members of this complete basis are not eligible for selection, namely many of those that involve only a few of the variables. The problem can be remedied by enlarging the eligible set to include all members of the complete tensor product basis. This in turn can be accomplished by a simple modification to Algorithm 1, or its continuous analog (Section 3.2).

The central operation in Algorithm 1 (lines 6, 12, 13) is to delete an existing (parent) basis function and replace it by both its product with a univariate truncated power spline basis function and the corresponding reflected truncated power function. The modification proposed here involves simply *not* removing the parent basis function. That is, the number of basis functions increases by two as a result of each iteration of the outer loop (split). All basis functions (parent and daughters) are eligible for further splitting. Note that this includes $B_1(\mathbf{x}) = 1$ (line 1). Basis functions involving only one variable (additive terms) can be produced by choosing $B_1(\mathbf{x})$ as the parent. Two-variable basis functions are produced by choosing a single variable basis function as the parent, and so on. Since no restrictions are placed on the choice of a parent term, the modified

procedure is able to produce models involving either high or low order interactions or both. It can produce purely additive models (13) by always choosing $B_1(\mathbf{x})$ as the parent.

This strategy of not removing a parent basis function, after it has been selected for splitting, also resolves the dilemma presented in the last paragraph of Section 3.2. A prohibition against more than one split on the same variable along the path leading to a single basis function can now be enforced without limiting the power of the procedure. Repeated splitting on the same variable is used by ($q = 0$) recursive partitioning to attempt to approximate local additive dependencies. This can now be directly accomplished by repeated selection of the same parent for splitting (on the same variable) thereby introducing additional terms but not increasing the depth of the splitting. There is no longer a need for repeated factors associated with the same variable in a single basis function.

Combining the considerations of this and the preceding section leads to a generalization of recursive partitioning regression involving the following modifications to Algorithm 1:

(a) replacing the step function $H[\pm(x - t)]$ by a truncated power spline function $[\pm(x - t)]_+^q$.

(b) not removing the parent basis function $B_{m^*}(\mathbf{x})$ after it is split, thereby making it and both its daughters eligible for further splitting.

(c) restricting the product associated with each basis function to factors involving distinct predictor variables.

An important consideration in this generalization of recursive partitioning is the degree-of-continuity to impose on the solution; that is, the choice of q (21) (22). There are both statistical and computational trade-offs. These are discussed in Sections 3.7 and 3.9, where it is argued that only continuity of the approximating function and its first derivative should be imposed. Furthermore, the proposed implementing strategy is to employ $q = 1$ splines in the analog of Algorithm 1, and then to use the resulting solution (with discontinuous derivatives) to derive a continuous derivative solution. The detailed discussion of this is deferred to Section 3.7.

3.4. MARS Algorithm.

Algorithm 2 implements the forward stepwise part of the MARS strategy by incorporating the modificatons to recursive partitioning (Algorithm 1) outlined above. Truncated power basis functions ($q = 1$) are substituted for step functions in lines 6, 12, and 13. The parent basis function is included in the modified model in line 6, and remains in the updated model through the logic of lines 12–14. Basis function products are constrained to contain factors involving distinct variables by the control loop over the variables in line 4 [see (20), (22)]. This algorithm produces $M_{\mathbf{max}}$ $q = 1$ tensor product (truncated power) spline basis functions that are a subset of the complete tensor product basis with knots located at all distinct marginal data values. As with recursive partitioning, this basis set is then subjected to a backwards stepwise deletion strategy to produce a final set of basis functions. The knot locations associated with this approximation are then used to derive a piecewise cubic basis, with continuous first derivatives (Section 3.7), thereby producing the final (continuous derivative) model.

Algorithm 2 (MARS – forward stepwise)

$B_1(\mathbf{x}) \leftarrow 1$; $M \leftarrow 2$
Loop until $M > M_{\max}$: $lof^* \leftarrow \infty$
  For $m = 1$ to $M - 1$ do:
    For $v \notin \{v(k,m)|\ 1 \leq k \leq K_m\}$
      For $t \in \{x_{vj}|B_m(\mathbf{x}_j) > 0\}$
        $g \leftarrow \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_M B_m(\mathbf{x})[+(x_v - t)]_+ + a_{M+1} B_m(\mathbf{x})[-(x_v - t)]_+$
        $lof \leftarrow \min_{a_1,\cdots,a_{M+1}} LOF(g)$
        if $lof < lof^*$ then $lof^* \leftarrow lof$; $m^* \leftarrow m$; $v^* \leftarrow v$; $t^* \leftarrow t$ end if
      end for
    end for
  end for
  $B_M(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x})[+(x_{v^*} - t^*)]_+$
  $B_{M+1}(\mathbf{x}) \leftarrow B_{m^*}(\mathbf{x})[-(x_{v^*} - t^*)]_+$
  $M \leftarrow M + 2$
end loop
end algorithm

Unlike recursive partitioning, the basis functions produced by Algorithm 2 do not have zero pairwise product expectations; that is, the corresponding "regions" are not disjoint but overlap. Removing a basis function does not produce a "hole" in the predictor space (so long as the constant basis function B_1 is never removed). As a consequence, it is not necessary to employ a special "two at a time" backward stepwise deletion strategy based on sibling pairs. A usual "one at a time" backward stepwise procedure of the kind ordinarily employed with regression subset selection can be used. Algorithm 3 presents such a procedure for use in the MARS context.

Algorithm 3 (MARS – backwards stepwise)

$J^* = \{1, 2, \cdots, M_{\max}\}$; $K^* \leftarrow J^*$
$lof^* \leftarrow \min_{\{a_j|j\in J^*\}} LOF(\sum_{j\in J^*} a_j B_j(\mathbf{x}))$
For $M = M_{\max}$ to 2 do: $b \leftarrow \infty$; $L \leftarrow K^*$
  For $m = 2$ to M do: $K \leftarrow L - \{m\}$
    $lof \leftarrow \min_{\{a_k|k\in K\}} LOF(\sum_{k\in K} a_k B_k(\mathbf{x}))$
    if $lof < b$ then $b \leftarrow lof$; $K^* \leftarrow K$ end if
    if $lof < lof^*$ then $lof^* \leftarrow lof$; $J^* \leftarrow K$ end if
  end for
end for
end Algorithm

Initially (line 1) the model is comprised of the entire basis function set J^* derived from Algorithm 2. Each iteration of the outer For-loop of Algorithm 3 causes one basis function to be deleted. The inner For-loop chooses which one. It is the one whose removal either improves the fit the most or degrades it the least. Note that the constant basis function $B_1(\mathbf{x}) = 1$ is never eligible

for removal. Algorithm 3 constructs a sequence of $M_{\max} - 1$ models, each one having one less basis function than the previous one in the sequence. The best model in this sequence is returned (in J^*) upon termination.

3.5. ANOVA Decomposition.

The result of applying Algorithms 2 and 3 is a model of the form

$$\hat{f}(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{k=1}^{K_m} [s_{km} \cdot (x_{v(k,m)} - t_{km})]_+. \tag{23}$$

Here a_0 is the coefficient of the constant basis function B_1, and the sum is over the basis functions B_m (22) produced by Algorithm 2 that survive the backwards deletion strategy of Algorithm 3 and $s_{km} = \pm 1$. This (constructive) representation of the model does not provide very much insight into the nature of the approximation. By simply rearranging the terms, however, one can cast the model into a form that reveals considerable information about the predictive relationship between the response y and the covariates $\mathbf{x}$. The idea is to collect together all basis functions that involve identical predictor variable sets.

The MARS model (23) can be recast into the form

$$\begin{aligned} \hat{f}(\mathbf{x}) = a_0 &+ \sum_{K_m=1} f_i(x_i) + \sum_{K_m=2} f_{ij}(x_i, x_j) \\ &+ \sum_{K_m=3} f_{ijk}(x_i, x_j, x_k) + \cdots. \end{aligned} \tag{24}$$

The first sum is over all basis functions that involve only a single variable. The second sum is over all basis functions that involve exactly two variables, representing (if present) two-variable interactions. Similarly, the third sum represents (if present) the contributions from three-variable interactions, and so on.

Let $V(m) = \{v(k,m)\}_1^{K_m}$ be the variable set associated with the mth basis function B_m (23). Then each function in the first sum of (24) can be expressed as

$$f_i(x_i) = \sum_{\substack{K_m=1 \\ i \in V(m)}} a_m B_m(x_i). \tag{25}$$

This is a sum over all single variable basis functions involving only x_i, and is a $q = 1$ spline representation of a univariate function. Each bivariate function in the second sum of (24) can be expressed as

$$f_{ij}(x_i, x_j) = \sum_{\substack{K_m=2 \\ (i,j) \in V(m)}} a_m B_m(x_i, x_j), \tag{26}$$

which is a sum over all two-variable basis functions involving the particular pair of variables x_i and x_j. Adding this to the corresponding univariate contributions (25) (if present)

$$f_{ij}^*(x_i, x_j) = f_i(x_i) + f_j(x_j) + f_{ij}(x_i, x_j) \tag{27}$$

gives a $q = 1$ bivariate tensor product spline approximation representing the joint bivariate contribution of x_i and x_j to the model. Similarly, each trivariate function in the third sum can be obtained by collecting together all basis functions involving the particular variable triples

$$f_{ijk}(x_i, x_j, x_k) = \sum_{\substack{K_m=3 \\ (i,j,k)\in V(m)}} a_m B_m(x_i, x_j, x_k). \tag{28}$$

Adding this to the corresponding univariate and bivariate functions (25) (26) involving x_i, x_j and x_k, provides the joint contribution of these three variables to the model. Terms involving more variables (if present) can be collected together and represented similarly. Owing to its similarity to decompositions provided by the analysis of variance for contingency tables, we refer to (24) as the ANOVA decomposition of the MARS model.

Interpretation of the MARS model is greatly facilitated through its ANOVA decomposition (24). This representation identifies the particular variables that enter into the model, whether they enter purely additively or are involved in interactions with other variables, the level of the interactions, and the other variables that participate in them. Interpretation is further enhanced by representing the ANOVA decomposition graphically. The additive terms (25) can be viewed by plotting $f_i(x_i)$ against x_i as one does in additive modeling. The two-variable contributions can be visualized by plotting $f^*_{ij}(x_i, x_j)$ (27) against x_i and x_j using either contour or perspective mesh plots. Models involving higher level interactions can be (roughly) visualized by viewing plots on variable pairs for several (fixed) values of the other (complementary) variables (see Section 4.7).

3.6. Model Selection

Several aspects of the MARS procedure (Algorithms 2 and 3) have yet to be addressed. Among these are the lack-of-fit criterion LOF (Algorithm 2, line 7, and Algorithm 3, lines 2 and 5), and the maximum number of basis functions $M_{\max}$ (Algorithm 2, line 2, and Algorithm 3, lines 1 and 3). The lack-of-fit criterion used with the algorithm depends on the distance (loss) function Δ specified with the integral (2) or expected (3) error. The most often specified distance is squared-error loss

$$\Delta[\hat{f}(\mathbf{x}), f(\mathbf{x})] = [\hat{f}(\mathbf{x}) - f(\mathbf{x})]^2 \tag{29}$$

because its minimization leads to algorithms with attractive computational properties. As will be seen in Section 3.9, this aspect is very impotant in the context of Algorithm 2, and so squared-error loss is adopted here as well. The goal of a lack-of-fit criterion is to provide a data based estimate of future prediction error (2) (3) which is then minimized with respect to the parameters of the procedure.

As in Friedman and Silverman (1989) and Friedman (1988) we use a modified form of the generalized cross-validation criterion originally proposed by Craven and Wahba (1979)

$$LOF(\hat{f}_M) = GCV(M) = \frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{f}_M(\mathbf{x}_i)]^2 \Big/ \left[1 - \frac{C(M)}{N}\right]^2. \tag{30}$$

Here the dependencies of $\hat{f}$ (23), and the criterion, on the number of (nonconstant) basis functions M is explicitly indicated. The GCV criterion is the average-squared residual of the fit to the data (numerator) times a penalty (inverse denominator) to account for the increased variance associated with increasing model complexity (number of basis functions M).

If the values of the basis function parameters (number of factors K_m, variables $v(k,m)$, knot locations t_{km} and signs s_{km}) associated with the MARS model were determined independently of the data response values $(y_1, \cdots, y_N)$, then only the coefficients $(a_0, \cdots, a_M)$ are being fit to the data. Consequently the complexity cost function is

$$C(M) = \text{trace}(\mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T) + 1 \tag{31}$$

where $\mathbf{B}$ is the $M \times N$ "data"matrix of the M (nonconstant) basis functions ($B_{ij} = B_i(\mathbf{x}_j)$). This is equal to the number of linearly independent basis functions in (23) and therefore $C(M)$ here (31) is just the number of parameters being fit. Using (31) in (30) leads to the GCV criterion proposed by Craven and Wahba (1979).

The MARS procedure (like recursive partitioning) makes heavy use of the response values to construct a basis function set. This is how it achieves its power and flexibility. This (usually dramatically) reduces the bias of model estimates, but at the same time increases the variance since additional parameters (of the basis functions) are being adjusted to help better fit the data at hand. The reduction in bias is directly reflected in reduced (expected) average squared residual (numerator (30)). The (inverse) denominator (30) (31) is, however, no longer reflective of the (increased) variance owing to the additional number of (basis function) parameters as well as their nonlinear nature.

Friedman and Silverman (1989) suggested using (30) as a lack-of-fit criterion in these circumstances, but with an increased cost complexity function $\tilde{C}(M)$ to reflect the additional (basis function) parameters that, along with the expansion coefficients $(a_0, \cdots, a_M)$, are being fit to the data. Such a cost complexity function can be expressed as

$$\tilde{C}(M) = C(M) + d \cdot M. \tag{32}$$

Here $C(M)$ is given by (31) and M is the number of nonconstant basis functions in the MARS model, being proportional to the number of (nonlinear) basis function parameters. The quantity d in (32) represents a cost for each basis function optimization and is a (smoothing) parameter of the procedure. Larger values for d will lead to fewer knots being placed and thereby smoother function estimates.

In the case of additive modeling, Friedman and Silverman (1989) gave an argument for choosing the value $d = 2$, based on the expected decrease in the average-squared residual by adding a single knot to make a piecewise-linear model. The MARS procedure can be forced to produce an additive model by simply modifying the upper limit of the outer For-loop in Algorithm 2 (line 3) to always have the value one. With this modification only the constant basis function B_1 is eligible for

"splitting" and the resulting model is a sum of functions each of a single variable ($K_m = 1$) which, after the corresponding ANOVA decomposition (24), assumes the form of an additive model (13). Restricting MARS in this manner leads to a palindromically invariant version of the Friedman and Silverman (1989) procedure. This type of additive modeling is a restricted version of general MARS modeling. A higher degree of optimization (over m) is being performed by the latter causing the data at hand to be fit more closely, thereby increasing variance. In order for the GCV criterion (30) (31) (32) to reflect this, an even larger value for d is appropriate.

One method for choosing a value for d in any given situation would be to simply regard it as a parameter of the procedure that can be used to control the degree of smoothness imposed on the solution. Alternatively it could be estimated through a standard sample reuse technique such as bootstrapping (Efron, 1983) or cross-validation (Stone, 1974). In a fairly wide variety of simulation studies (a subset of which are presented in Section 4) the resulting model and its accuracy (2) (3) are seen to be fairly independent of the value chosen for the parameter d (32). These simulation studies indicate:

(1) The optimal cost complexity function $\tilde{C}(M)$ to be used in the GCV criterion (30) (in the context of MARS modeling) is a monotonically increasing function with decreasing slope as M increases.

(2) The approximation (32), with $d = 3$, is fairly effective, if somewhat crude.

(3) The best value for d in any given situation depends (weakly) on M, N, n and the distribution of the covariate values in the predictor space.

(4) Over all situations studied, the best value for d is in the range $2 \leq d \leq 4$.

(5) The actual accuracy in terms of either integral (2) (29) or expected (3) (29) squared error is fairly insensitive to the value of d in this range.

(6) The value of the GCV criterion for the final MARS model does exhibit a moderate dependence on the value chosen for d.

A consequence of (5) and (6) is that while how well one is doing with the MARS approach is fairly independent of d, how well one thinks one is doing (based on the optimized GCV score) does depend somewhat on its value. Therefore, a sample reuse technique might be used to obtain an additional estimate of the goodness-of-fit of the final model if it needs to be known fairly precisely.

The strategy in recursive partitioning regression (Breiman et al., 1984) is to let the forward stepwise procedure produce a fairly large number of regions (basis functions) and then have the backwards stepwise procedure trim the model back to an appropriate size. The arguments in favor of this apply equally well to the MARS approach. Therefore, the value chosen for $M_{\max}$ in Algorithms 2 and 3 should be considerably larger than the optimal (minimal GCV) model size M^*. Typically choosing $M_{\max} = 2M^*$ is sufficient.

3.7. Degree-of-Continuity.

One of the central ideas leading to the MARS generalization of recursive partitioning is to replace the step function implicit in the latter with a truncated power spline basis function (21). This leads to an approximation in the form of an expansion in tensor product spline basis functions.

The continuity properties of this approximation are governed by the order q (21) chosen for the univariate spline functions comprising the tensor products; derivatives exist to order q.

If the intent is accurate estimation of the function (as opposed to its derivatives of various orders) then there is little to be gained by imposing continuity beyond that of the function itself. If the true underlying function nowhere has a very large local second derivative then a small additional increase in accuracy can be achieved by imposing continuous first derivatives. Also, continuous (first) derivative approximations have considerably more cosmetic appeal. There is, however, little to be gained and in fact much to lose, by imposing continuity beyond that of the first derivative, especially in high dimensional settings.

The difficulty with higher order regression splines centers on so called "end effects." The largest contribution to the average approximation error (2) (3) emanates from locations $\mathbf{x}$ near the boundaries of the domain. This phenomenon is well known even in univariate smoothing ($n = 1$), and is especially severe in higher dimensions. As the dimension of the covariate space increases, the fraction of the data points near a boundary increases rapidly. Fitting high degree polynomials (associated with high degree regression splines) in these regions leads to very high variance of the function estimate there. This is mainly due to the lack of constraints on the fit at the boundaries.

One approach that has been suggested (Stone and Koo, 1985) is to modify the spline basis functions so that near the ends of the data interval (on each variable) they smoothly join a linear function. This can substantially help moderate the bad end effects of (unmodified) regression splines in the case of smoothing ($n = 1$) and additive modeling (13), although the approximating basis functions can still have very large slope near the boundaries. A computationally simpler way to ensure a linear approximation near the boundaries is to make a piecewise-linear approximation everywhere by using $q = 1$ tensor product splines. This is accomplished in the MARS approach by using $q = 1$ truncated power (univariate) spline basis functions (21) in Algorithm 2 (lines 6, 12, and 13).

A piecewise-linear approximation, of course, does not possess continuous derivatives. The lowest order spline approximation with continuous derivatives involves $q = 2$ univariate spline basis functions. Their use, however, leads to the problems cited above. Motivated by the approach of Stone and Koo (1985) we fit with a modified basis set. These functions resemble $q = 1$ splines, but have continuous derivatives.

The model (23) produced by Algorithms 2 and 3 involves a sum of products of functions of the form

$$b(x|s,t) = [s(x-t)]_+. \tag{33}$$

Our strategy for producing a model with continuous derivatives is to replace each such function by

a corresponding truncated cubic function of the form

$$C(x|s=+1,t_-,t,t_+) = \begin{cases} 0 & x \le t_- \\ p_+(x-t_-)^2 + r_+(x-t_-)^3 & t_- < x < t_+ \\ x - t & x \ge t_+ \end{cases}$$
$$C(x|s=-1,t_-,t,t_+) = \begin{cases} -(x-t) & x \le t_- \\ p_-(x-t_+)^2 + r_-(x-t_+)^3 & t_- < x < t_+ \\ 0 & x \ge t_+ \end{cases} \tag{34}$$

with $t_- < t < t_+$. Setting

$$\begin{aligned} p_+ &= (2t_+ + t_- - 3t)/(t_+ - t_-)^2 \\ r_+ &= (2t - t_+ - t_-)/(t_+ - t_-)^3 \\ p_- &= (3t - 2t_- - t_+)/(t_- - t_+)^2 \\ r_- &= (t_- + t_+ - 2t)/(t_- - t_+)^3 \end{aligned} \tag{35}$$

causes $C(x|s,t_-,t,t_+)$ to be continuous and have continuous first derivatives. There are second derivative discontinuities at $x = t_\pm$. Each truncated linear function (33) is characterized by a single knot location t, whereas each corresponding truncated cubic function (34) (35) is characterized by three knots; these are a central knot t and upper/lower side knots $t_\pm$. Figure 2a compares the two functions.

Each factor (33) in every basis function in the approximation produced by Algorithms 2 and 3 (23) is replaced by a corresponding truncated cubic factor (34) (35). The central knot t (34) is placed at the same location as the (single) knot for its associated truncated linear function (33). The side knots $t_\pm$, $t_- < t < t_+$, are located so as to reduce the number of second derivative discontinuities. This is accomplished through the ANOVA decomposition of the MARS model (24) (25) (26) (28).

Each basis function m in (23) has a knot set $\{t_{km}\}_1^{K_m}$. The ANOVA decomposition collects together all basis functions corresponding to exactly the same variable set $\{v(k,m)\}_1^{K_m}$. Thus, the knot sets associated with each ANOVA function (25) (26) (28) can be viewed as a set of points (multivariate knots) in the same K_m-dimensional space. The projections of these points onto each of the respective K_m axes, $v(k,m)$, gives the knot locations of the factors that correspond to that variable. These are the central knot locations for the piecewise cubic factors. The side knots $t_\pm$ for each cubic factor are placed at the midpoints between its central knot and the two adjacent central knots in the same projection. The lower/upper side knots for the corresponding smallest/largest projected central knot value are placed midway between the central knot and the smallest/largest data value on that variable. Figure 2b illustrates this process for one and two dimensional ANOVA functions.

The final model is obtained by fitting the resulting piecewise cubic basis function set to the data. It will have continuous first (but not second) derivatives. The contribution to the fitted model of each basis function far from its central knot location will be the same as its corresponding piecewise linear basis function. Therefore this continuous derivative model will tend to have the same highly desirable boundary properties as the piecewise linear model produced by Algorithms

2 and 3. The important ingredient is that the slope of each univariate basis factor never exceeds a value of one.

3.8. Knot Optimization.

The MARS procedure (Algorithm 2), as well as recursive partitioning (Algorithm 1), can be viewed as a technique for developing a multivariate model, based on sums of products of univariate functions (17) (20) (23), through the use of a univariate smoother. This can be seen by casting the (MARS) model in the form

$$\hat{f}(\mathbf{x}) = \hat{f}_{\setminus mv}(\mathbf{x}) + B_m(\mathbf{x})\varphi_{mv}(x_v), \tag{36}$$

with

$$\hat{f}_{\setminus mv}(\mathbf{x}) = \sum_{\substack{V(i)\neq V(m) \\ V(i)\neq V(m)+\{v\}}} a_i B_i(\mathbf{x}). \tag{37}$$

The function $\varphi_{mv}(x_v)$ has the form

$$\varphi_{mv}(x_v) = c_0 + \sum_{j=1}^{J} c_j[s_j(x_v - t_j)]_+ \tag{38}$$

which is just a ($q = 1$) piecewise linear spline representation of a univariate function. (In the case of recursive partitioning it would be a $q = 0$ piecewise constant representation.) The second term in (36) isolates the contributions to the model of the variable set $V(m)$ of the mth basis function, and the set including the variable v with $V(m)$. The first term (37) represents the contributions of the variables from the other basis functions. Minimization of the lack-of-fit criterion (30) in Algorithm 2 (line 7) performs a joint optimization of the current model with respect to the coefficients $(c_0, \cdots, c_J)$ of the univariate function φ_{mv} (38) and those of the other basis functions $\{a_i\}$ (37).

Expressing the current model in the form given by (36) and letting

$$R_{\setminus mv} = y - \hat{f}_{\setminus mv} \tag{39}$$

this optimization can be written in the form

$$E[R_{\setminus mv} - B_m\varphi_{mv}(x_v)]^2 = \min. \tag{40}$$

Here the dependence of the respective quantities on the multivariate argument $\mathbf{x}$ has been surpressed. If one fixes the values of the coefficients $\{a_i\}$ (37) then (40) has the general solution

$$\varphi_{mv}(x_v) = E\left[B_m^2\left(\frac{R_{\setminus mv}}{B_m}\right)\middle| x_v\right] / E[B_m^2|x_v], \tag{41}$$

which can be estimated by a weighted smooth of $R_{\setminus mv}/B_m$ on x_v, with weights given by B_m^2. Letting this smooth take the form of a piecewise linear spline approximation (38) gives rise (in an indirect way) to the MARS approach.

This framework provides the connection between MARS and the smoothing (and additive modeling) method (TURBO) suggested by Friedman an Silverman (1989). They presented a forward stepwise strategy for knot placement in a simple piecewise linear smoother. The inner For-loop of Algorithm 2 can be viewed as an application of this strategy for choosing the best location for the next knot t_J in $\varphi_{mv}(x_v)$ (38) (41) in the more general context of MARS modeling. In fact, in the univariate case ($n = 1$) the MARS algorithm simply represents a (palindromically invariant) version of TURBO. As noted above, restricting the upper limit of the outer For-loop to always have the value one (Algorithm 2) gives rise (for $n > 1$) to the TURBO method of additive modeling.

Both recursive partitioning (Algorithm 1, line 5) and MARS (Algorithm 2, line 5) make every distinct (nonzero weighted) marginal data value eligible for knot placement. As pointed out by Friedman and Silverman (1989), this has the effect of permitting the corresponding piecewise linear smoother (38) (41) to achieve a local minimum span of one observation. In noisy settings this can lead to locally high variance of the function estimate. There is no way that a smoother, along with its lack-of-fit criterion, can distinguish between sharp structure in the true underlying function f (1), and a run of either positive or negative error values ϵ (1). If one assumes (as one must) that the underlying function is smooth compared to the noise, then it is reasonable to impose a minimum span on the smoother that makes it resistant to runs in the noise of length likely to be encountered in the errors. In the context of piecewise linear smoothing this translates into a minimal number L of (nonzero weighted) observations between each knot. Assuming a symmetric error distribution, Friedman and Silverman (1989) use a coin tossing argument to propose choosing $L = L^*/2.5$ with L^* being the solution to

$$\Pr(L^*) = \alpha. \tag{42}$$

Here $\Pr(L^*)$ is the probability of observing a (positive or negative) run of length L^* or longer in nN_m tosses of a fair coin, and α is a small number (say $\alpha = 0.05$ or 0.01). The quantity N_m is the number of observations for which $B_m > 0$ (Algorithm 2, line 5). The relevant number of tosses is nN_m since there are that many potential locations for each new knot (inner two For-loops in Algorithm 2) for each basis function B_m (outer For-loop).

For $nN_m \geq 10$ and $\alpha < 0.1$ a good approximation to L^* (42) is

$$L^* = -\log_2\left[-\frac{1}{nN_m}\ell n(1-\alpha)\right],$$

so that a reasonable number of counts between knots is given by

$$L(\alpha) = -\log_2\left[-\frac{1}{nN_m}\ell n(1-\alpha)\right]/2.5. \tag{43}$$

The denominator in (43) arises from the fact that a piecewise linear smoother must place between two and three knots in the interval of the run to respond to it, and not degrade the fit anywhere else. Using (43) gives the procedure resistance, with probability $1 - \alpha$, to a run of positive or negative error values in the interior of the interval.

The arguments that lead to (43) do not apply to the ends of the interval; that is, $L(\alpha)$ refers to the number of counts between knots but not to the number of counts between the extreme knot locations and the corresponding ends of the interval defined by the data. As discussed in Section 3.7, it is essential that end effects be handled well for the procedure to be successful. An argument analogous to the one that leads to $L(\alpha)$ (43) for the interior, can be advanced for the ends. The probability of a run of length L^* or longer of positive or negative error values at the beginning or end of the data interval is 2^{-L^*+3}. There are n such intervals, corresponding to the n predictor variables, so that the total probability of encountering an end run is

$$\Pr(L^*) = n2^{-L^*+3}. \tag{44}$$

Therefore requiring at least

$$Le(\alpha) = 3 - \log_2(\alpha/n) \tag{45}$$

observations between the extreme knots and the corresponding ends of the interval provides resistance (with probability $1 - \alpha$) to runs at the ends of the data intervals.

The quantity α in (43) (45) can be regarded as another smoothing parameter of the procedure. Both $L(\alpha)$ (43) and $Le(\alpha)$ (45) are, however, fairly insensitive to the value of α. The differences $L(.01)$–$L(.05)$ and $Le(.01)$–$Le(.05)$ are both approximately equal to 2.3 observations. In any case both expressions can only be regarded as approximate since they only consider the signs and ignore the magnitudes of the errors in the run.

It can be noted that (36), (37), (39), and (41) could be used to develop a generalized backfitting algorithm based on a (univariate) local averaging smoother, in direct analogy to the backfitting algorithm for additive modeling (Friedman and Stuetzle, 1981, Breiman and Friedman, 1985, and Buja, Hastie and Tibshirani, 1989). Like MARS, this generalized backfitting algorithm could be used to fit models involving sums of products of univariate curve estimates. It would, however, lack the flexibility of the MARS procedure (especially in high dimensions) owing mainly to the latter's close relation to the recursive partitioning approach (local variable subset selection – see Sections 2.4.2 and 6). It would also tend to be computationally far more expensive.

3.9. Computational Considerations

In Sections 3.2 and 3.3 the MARS procedure was motivated as a series of conceptually simple extensions to recursive partitioning regression. In terms of implementation, however, these extensions produce a dramatic change in the algorithm. The usual implementations of recursive partitioning regression [AID (Morgan and Sonquist, 1963) and CART (Breiman, et al., 1984)] take strong advantage of the special nature of step functions, along with the fact that the resulting basis functions have disjoint support, to dramatically reduce the computation associated with the middle and inner For-loops of Algorithm 1 (lines 4 and 5). In the case of least-squares fitting, very simple updating formulae can be employed to reduce the computation for the associated linear (least-squares) fit (line 7) from $O(NM^2 + M^3)$ to $O(1)$. The total computation can therefore

be made proportional to $nN\ M_{\max}$, after sorting. Unfortunately, these same tricks cannot be applied to the implementation of the MARS procedure. In order to make it computationally feasible, different updating formulae must be derived for the MARS algorithm.

The minimization of the lack-of-fit criterion (30) in Algorithm 2 (line 7) is a linear least-squares fit of the response y on the current basis function set (line 6). There are a variety of techniques for numerically performing this fit. The most popular, owing to its superior numerical properties, is based on the QR decomposition (see Golub and Van Loan, 1983) of the basis "data" matrix $\mathbf{B}$,

$$B_{mi} = B_m(\mathbf{x}_i). \tag{46}$$

As noted above, however, computational speed is of paramount importance since this fit must be repeated many times in the course of running the algorithm. A particular concern is keeping the computation linear in the number of observations N, since this is the largest parameter of the problem. This rules out the QR decomposition technique in favor of an approach based on using the Cholesky decomposition to solve the normal equations

$$\mathbf{B}^T\mathbf{B}\mathbf{a} = \mathbf{B}^T\mathbf{y} \tag{47}$$

for the vector of basis coefficients $\mathbf{a}$ (line 6). Here $\mathbf{y}$ is the (length N) vector of response values. This approach is known to be less numerically stable than the QR decomposition technique. Also, the truncated power basis (21) is the least numerically stable representation of a spline approximation. Therefore, a great deal of care is required in the numerical aspects of an implementation of this approach. This is discussed below.

If the basis functions are centered to have zero mean then the matrix product $\mathbf{B}^T\mathbf{B}$ is proportional to the covariance matrix of the current basis function set. The normal equations (47) can be written

$$\mathbf{V}\mathbf{a} = \mathbf{c} \tag{48}$$

with

$$\begin{aligned} V_{ij} &= \sum_{k=1}^{N} B_j(\mathbf{x}_k)[B_i(\mathbf{x}_k) - \bar{B}_i], \\ c_i &= \sum_{k=1}^{N} (y_k - \bar{y})B_i(\mathbf{x}_k), \end{aligned} \tag{49}$$

and $\bar{B}_i$ and $\bar{y}$ the corresponding averages over the data. These equations (48) (49) must be resolved for every eligible knot location t, for every variable v, for all current basis functions m, and for all iterations M of Algorithm 2 (lines 2, 3, 4, and 5). If carried out in a straightforward manner this would require computation proportional to

$$C \sim nNM_{\max}^4(\alpha N + \beta M_{\max})/L \tag{50}$$

with α and β constants of proportionality and L given by (43). This computational burden would be prohibitive except for very small problems or very large computers. Although it is not possible

to achieve the dramatic reduction in computation for MARS as can be done for recursive partitioning regression, one can reduce the computation enough, so that moderate sized problems can be conveniently run on small computers. Following Friedman and Silverman (1989), the idea is to make use of the special properties of the $q = 1$ truncated power spline basis functions to develop rapid updating formulae for the quantities that enter into the normal equations (48) (49), as well as to take advantage of the rapid updating properties of the Cholesky decomposition (see Golub and Van Loan, 1983).

The most important special property of the truncated power basis used here is that each (univariate) basis function is characterized by a single knot. Changing a knot location changes only one basis function, leaving the rest of the basis unchanged. Other bases for representing spline approximations, such as the minimal support B-splines, have superior numerical properties but lack this important computational aspect. Updating formulae for B-splines are therefore more complex giving rise to slower computation.

The current model (Algorithm 2, line 6) can be reexpressed as

$$g' \leftarrow \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_M B_m(\mathbf{x}) x_v + a_{M+1} B_m(\mathbf{x})(x_v - t)_+. \tag{51}$$

The inner For-loop (line 5) minimizes the GCV criterion (30) jointly with respect to the knot location t and the coefficients $a_1, \cdots, a_{M+1}$. Using g' (51) in place of g (line 6) yields an equivalent solution with the same optimizing GCV criterion lof^* (line 8) and knot location t^* (line 8). (The solution coefficient values will be different.) The advantage of using g' (51) is that only one basis function is changing as the knot location t changes.

Friedman and Silverman (1989) developed updating formulae for least-squares fitting of $q = 1$ splines by visiting the eligible knot locations in decreasing order and taking advantage of that fact for $t \leq u$

$$(x - t)_+ - (x - u)_+ = \begin{cases} 0 & x \leq t \\ x - t & t < x < u \\ u - t & x \geq u. \end{cases}$$

The Friedman and Silverman (1989) updating formulae can be extended in a straightforward man-

ner to the more general MARS setting, giving ($t \leq u$)

$$
\begin{aligned}
c_{M+1}(t) &= c_{M+1}(u) + \sum_{t \leq x_{vk} < u} (y_k - \bar{y}) B_{mk}(x_{vk} - t) \\
&\quad + (u - t) \sum_{x_{vk} \geq u} (y_k - \bar{y}) B_{mk}, \\
V_{i,M+1}(t) &= V_{i,M+1}(u) + \sum_{t \leq x_{vk} < u} (B_{ik} - \bar{B}_i) B_{mk}(x_{vk} - t) \\
&\quad + (u - t) \sum_{x_{vk} \geq u} (B_{ik} - \bar{B}_i) B_{mk}, \qquad (1 \leq i \leq M), \\
V_{M+1,M+1}(t) &= V_{M+1,M+1}(u) + \sum_{t \leq x_{vk} < u} B_{mk}^2 (x_{vk} - t)^2 \\
&\quad + (u - t) \sum_{x_{vk} \geq u} B_{mk}^2 (2x_{vk} - t - u) + (s^2(u) - s^2(t))/N,
\end{aligned}
\tag{52}
$$

with $s(t) = \sum_{x_{vk} \geq t} B_{mk}(x_{vk} - t)$. In (52) B_{ik} and B_{mk} are elements of the basis function "data" matrix (46), the x_{vk} are elements of the original data matrix, and y_k are the data response values.

These updating formulae (52) can be used to obtain the last $(M + 1)$st row (and column) of the basis covariance matrix $\mathbf{V}$ and last element of the vector $\mathbf{c}$ at all eligible knot locations t with computation proportional to $(M + 2)N_m$. Here N_m is the number of observations for which $B_m(\mathbf{x}) > 0$ (line 5). Note that all the other elements of $\mathbf{V}$ and $\mathbf{c}$ do not change as the knot location t changes. This permits the use of updating formulae for the Cholesky decomposition to reduce its computation from $O(M^3)$ to $O(M^2)$ (in solving the normal equations (48)) at each eligible knot location. Therefore the computation required for the inner For-loop (lines 5–9) is proportional to $\alpha M N_m + \beta M^2 N_m / L$. This gives an upper bound on total computation for Algorithm 2 as being proportional to

$$
C^* \sim n N M_{\max}^3 (\alpha + \beta M_{\max}/L). \tag{53}
$$

Thus, comparing with (50) the use of updating formulae is seen to reduce the computation roughly by a factor of $N M_{\max}/L$. For typical values of $N = 200$, $M_{\max} = 30$, and $L = 5$ this reduces the required computation by roughly a factor of 1000.

Table 1 shows the total computation time (sec.) of the MARS procedure as a function of $M_{\max}$ for one of the examples (AC circuit impedance, Section 4.4.1) discussed below. These times were obtained on a SUN Microsystems Model 3/260 (with floating point accelerator). For this example $n = 4$ and $N = 200$. The computation scales linearly in both n and N with the MARS algorithm.

Three timing sequences are shown in Table 1, corresponding to different constraints being placed on the final MARS model. The first row ($mi = 1$) corresponds to an additive model where interactions among the variables are prohibited. As mentioned above, this is accomplished by suppressing the outer For-loop in Algorithm 2 (line 3) and only allowing the constant basis function $B_1(\mathbf{x}) = 1$ to appear in the products with the univariate spline basis functions. This, of course, reduces the total computation roughly by a factor proportional to $M_{\max}$. The second row

($mi = 2$) only allows two-variable interactions to appear in the model. This reduces computation a little since only previous basis functions involving one variable are permitted to appear in the products. The last row in Table 1 ($mi = n$) shows the times for the fully unconstrained MARS model.

It should be noted that in all the examples discussed in this paper (some of which, like this one, involve fairly complex functions) the optimal number of basis functions was between 10 and 15, so that $M_{\max}$ values around 20 to 30 were appropriate. Setting $M_{\max} = 50$ permits the procedure to use from 125 to 200 degrees of freedom to fit the final model, if required, thereby allowing it to approximate very complex functions. Clearly though, for very large problems (say $n > 20$ and $N > 1000$) either long execution times or fast computers (compared to the one used here) would be required. It can also be noted that the MARS algorithm admits a high degree of parallelization so that it could run very fast on computers with parallel architectures.

Updating formulae for higher order ($q > 1$) truncated power spline basis functions (21) can be developed in analogy to (52). They would, however, be far more complex than those for $q = 1$ leading to much slower execution of the algorithm. Also, their corresponding numerical properties would be very much worse.

At any point during the execution of Algorithm 2 the current basis function set need not be linearly independent. (The basis functions set comprising the final model is, however, always linearly independent.) Therefore, the covariance matrix $\mathbf{V}$ appearing in the normal equations (48) may be singular. This presents no fundamental problem since they can be solved by applying pivoting in the Cholesky decomposition (see Dongarra, Moler, Bunch and Stewart, 1979). A better strategy from the point of view of MARS modeling is, however, to slightly modify the normal equations via

$$(\mathbf{V} + \epsilon \mathbf{D})\mathbf{a} = \mathbf{c} \tag{54}$$

where $\mathbf{D}$ is a diagonal $(M + 1) \times (M + 1)$ matrix comprised of the diagonal elements of $\mathbf{V}$. The coefficients for the basis function set $\mathbf{a}$ are then taken to be the solution derived from (54). The average-squared-residual

$$ASR(\mathbf{a}) = \frac{1}{N}\left[\sum_{k=1}^{N}(y_k - \bar{y})^2 - \sum_{i=1}^{M+1} a_i(c_i + \delta D_{ii}a_i)\right] \tag{55}$$

is still used as the numerator of the GCV criterion (30). The value for δ is taken to be a small number just large enough to maintain numerical stability.

The principal advantage of this "ridge regression" approach (54) is that it eliminates the need for pivoting in the Cholesky decomposition update, thereby increasing execution speed. Additional advantages are that it increases numerical stability to help compensate for the bad numerical properties of the truncated power spline basis representation, and it applies a small overall shrinkage to the solution coefficients to help compensate for the selection bias inherent in stepwise regression procedures (see Copas, 1983).

The updating formulae (52) are not necessarily numerically stable. Widely different locations and scales for the predictor variables can cause instabilities that adversely effect the quality of the final model. The MARS procedure is (except for numerics) invariant to the locations and scales of the predictor variables. It is therefore reasonable to perform a transformation that causes the resulting locations and scales to be most favorable from the point of view of numerical stability. Standardizing them to each have zero location and unit scale provides good numerical properties.

4.0. Simulation Studies and Examples

In the following sections we present the results of applying the MARS procedure to a series of simulated and real data sets. The goal is to try to gain some understanding of its properties and to learn in what situations one might expect it to provide better performance than existing methodology. In all the examples the smoothing parameter d (32) was taken to be $d = 3$. The software automatically reduces it to $2d/3$ $(= 2)$ for additive modeling. The minimum number of observations between knots was determined by (43) and the number between the extreme knots and the edges was determined by (45), both with $\alpha = 0.05$. In all examples the explanatory variables were standardized to aid in numerical stability (see Section 3.9). In all simulation studies the covariate vectors were independently drawn (from the same sampling distribution) for each replication of the experiment. Therefore, nonidentical (random) designs were realized for each of the 100 replications. All results reported are for the continuous derivative (piecewise cubic) model (see Section 3.7) unless otherwise noted.

4.1. Modeling Pure Noise

With a modeling procedure as flexible as MARS, a reasonable concern is that it might find considerable spurious structure in data for which the signal to noise ratio is small. This false structure would reflect the sampling fluctuations in the noise ϵ (1) and would provide a misleading indication of the association between the response and predictor variables. One would expect this effect to be especially severe for small samples in high dimensions. Our first simulation study indicates that this tends not to be the case for the MARS procedure.

Tables 2a and 2b summarize the results of applying MARS to pure noise $f(\mathbf{x}) = 0$ (1). Results are presented for two dimensionalities $(n = 5, 10)$ and three sample sizes $(N = 50, 100, 200)$. The summary consists of the percent points of the lower half of the distribution of the optimizing GCV score (30) for the MARS model, scaled by that for the corresponding constant model $\hat{f}(\mathbf{x}) = \bar{y}$. These distributions were obtained by applying MARS to 100 data sets for which the response values were randomly generated from a normal distribution and the covariate vectors were randomly generated from a uniform distribution in R^n. As in Table 1, Tables 2a and 2b show the results for three types of constraints being placed on the model. These constraints are controlled by the parameter mi, which is the maximum number of variables allowed to appear in any basis function, $K_m \leq mi$ (23), thereby controlling the number of variables that can participate in interaction effects. For $mi = 1$ the model is restricted to be additive in the variables, whereas for $mi = 2$, interactions are limited to those involving (at most) two variables. Setting $mi = n$ places no

constraint on the number of variables that can enter into interactions.

This simulation study represents one test of the lack-of-fit criterion based on the GCV score (30) using the cost complexity criterion $\tilde{C}(M)$ (31) (32). Tables 2a and 2b show that the MARS procedure in this situation seldom claims to produce a model that fits the data markedly better than the response mean. Over half of the time (as reflected by the median) it claims to provide no better fit than the constant model at all dimensionalities and sample sizes shown. Even the best MARS fit over the 100 trials (1% point) does not produce a distinctly superior GCV value to the constant (no structure) fit on the same data. This is especially noteworthy given the small sample sizes for these dimensionalities.

4.2. MARS Modeling on Additive Data.

A related concern to that of the previous section concerns what happens when MARS is applied in situations where the true underlying function f (1) is additive (13) in the predictor variables. It might be expected that given the ability of MARS to introduce a large number of complex interactions into its models, that it might be somewhat at a disadvantage in these situations when compared to procedures that restrict the model to be additive. This section presents a simulation study that indicates that this is not the case.

We use for this study an example presented in Friedman and Silverman (1989)

$$f(\mathbf{x}) = 0.1e^{4x_1} + 4/[1 + e^{-20(x_2-1/2)}] + 3x_3 + 2x_4 + x_5 + 0 \cdot \sum_{i=6}^{10} x_i. \tag{56}$$

This function has a nonlinear additive dependence on the first two variables, a linear dependence on the next three, and is independent of the last five (pure noise) variables. A simulation study was performed consisting of 100 replications of the following experiment. First $N(= 50, 100, 200)$ ten dimensional ($n = 10$) covariate vectors were generated in the unit hypercube. Then corresponding response values were assigned according to

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \qquad 1 \leq i \leq N \tag{57}$$

with the ϵ_i randomly generated from a standard normal and $f(\mathbf{x})$ given by (56). Here the signal to noise ratio is 3.28 so that the true underlying function (56) accounts for 91% of the variance of the response (57).

A reasonable strategy to be used with MARS modeling is to fit both an additive model ($mi = 1$) and one that permits interactions ($mi = 2$ or $mi = n$). The respective GCV scores of both models can then be compared and the one corresponding to the lowest score chosen. With this strategy a model involving interactions is only used if it claims (through its GCV score) to do better than an additive model on the same data.

For each of the 100 replications (at each sample size N) the MARS procedure was applied with $mi = 1, 2$, and n. The first value ($mi = 1$) corresponds to additive modeling, the second ($mi = 2$) permits interactions in at most two variables, whereas the last ($mi = n = 10$) fits an

unconstrained MARS model. For the last two mi values the corresponding interaction fit was only used if it produced a smaller GCV score than the additive model on the same data.

Table 3 compares the average accuracy of using this strategy to that of additive modeling on the purely additive data (56) (57). The principal measure of accuracy is the (scaled) integral-squared-error ISE (2) (29)

$$ISE = \int_D [\hat{f}(\mathbf{x}) - f(\mathbf{x})]^2 d^n x / \mathrm{Var}_{\mathbf{x}\in D} f(\mathbf{x}), \tag{58}$$

where here $n = 10$ and D is the ten-dimensional unit hypercube. For each replication of the simulation study, the ISE (58) was estimated by Monte Carlo integration using 5000 points randomly generated from a uniform distribution over D.

A closely related quantity of interest is the (scaled) predictive-squared-error

$$PSE = E[y - \hat{f}(\mathbf{x})]^2 / \mathrm{Var}\ \ y \tag{59}$$

which is related to the ISE (58) by

$$PSE = (ISE \cdot\ \mathrm{Var}_{\mathbf{x}\in D} f(\mathbf{x}) +\ \mathrm{Var}\ \epsilon)/\mathrm{Var}\ \ y \tag{60}$$

with ϵ being the error component (1). If the response values $y_1, \cdots, y_N$ are standardized to have unit variance then the GCV criterion (30) (31) (32) is intended as an estimate of the PSE (59) (60), and the ratio GCV/PSE provides an estimate of how well the optimized GCV criterion for the model is estimating the true PSE.

Shown in Table 3 are the average ISE (58), PSE (59) and GCV/PSE, along with the corresponding standard deviations (in parentheses) over the 100 replications at each sample size N $(= 50, 100, 200)$. (Note that the standard deviations of the averages shown in the table are one tenth the corresponding standard deviations shown.) Comparing the first row $(mi = 1)$ to the next two $(mi = 2, n)$ shows that there is little sacrifice in accuracy using interaction models in the context of the above strategy, even though the true underlying function (56) involves no interactions. Table 3 also shows that the optimized GCV score produced by the MARS fit slightly overestimates (on average) the actual predictive-squared-error for this problem at all sample sizes studied. This effect is most pronounced for the smallest sample size $(N = 50)$. (Note that the variability of this ratio as reflected by its standard deviation over the 100 replications if fairly high.)

The above strategy chooses the additive model over that involving interactions only if the former produces a GCV score no worse than the latter. The first column of Table 4 shows the number of times the additive model was chosen in the 100 replications of this simulation experiment. As can be seen, the additive model was being chosen most of the time. This is why the loss in accuracy was so slight. A more conservative strategy would be to accept the additive model if its GCV score is only slightly (say 5 or 10 percent) worse. The second and third columns of Table 4 show the number of times the additive model is chosen under these two slightly more conservative scenarios. In these cases the fit involving interactions is seen to be almost never chosen.

The results of this simulation study indicate that on data for which the true underlying function f (1) is additive, the MARS procedure does not produce fits involving interactions that appear distinctly superior to an additive model. This basically is another test of the lack-of-fit criterion (30) (31) (32), and especially the choice of $d = 3$ (32) for general MARS modeling and $d = 2$ for additive modeling with MARS (Friedman and Silverman, 1989).

4.3. A Simple Function of Ten Variables.

The previous examples (Sections 4.1 and 4.2) tested the ability of MARS to avoid finding structure when it is not present. It is at least equally important that it find structure when it does exist. The next several examples examine the ability of MARS to uncover interaction effects that are present in data. The first test example is taken from Friedman, Grosse and Stuetzle (1983). They considered trying to model the function

$$f(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 \tag{61}$$

in the $n = 6$ dimensional unit hypercube using $N = 200$ points. The covariates were randomly generated from a uniform distribution and the responses were assigned using (57) with $f(\mathbf{x})$ given by (61) and with ϵ being a standard normal deviate.

We consider here this same function (61) but in a more difficult setting. First we reduce the sample size to $N = 100$. In addition we increase the dimensionality of the covariate space to $n = 10$, so that instead of one noise variable, there are now five such variables that are independent of $f(\mathbf{x})$. For this study however the MARS procedure has no prior knowledge of the nature of the dependence of $f(\mathbf{x})$ on any of the variables. The signal to noise ratio for this example is high (4.8/1); the true underlying function accounts for 96% of the variance of the response. On the other hand, the dimension of the covariate space is high ($n = 10$), the sample is small ($N = 100$) and the function (61) is fairly highly structured.

Table 5a summarizes the MARS model derived from a data set generated from the above prescription. The data response values were standardized so that the resulting GCV scores are an estimate of the PSE (59). The first two lines in Table 5a give the optimizing GCV scores for the corresponding piecewise linear (23) and piecewise cubic (Section 3.7) fits. The third line gives the total number of (nonconstant) basis functions M in the final model, whereas the fourth line gives $\tilde{C}(M)$ (31) (32), the estimated number of linear degrees-of-freedom used in the fit. The ANOVA decomposition is summarized by one row for each ANOVA function. The columns represent summary quantities for each one. The first column lists the function number. The second gives the standard deviation of the function. This gives one indication of its (relative) importance to the overall model and can be interpreted in a manner similar to a standardized regression coefficient in a linear model. The third column provides another indication of the importance of the corresponding ANOVA function, by listing the GCV score for a model with all of the basis functions corresponding to that particular ANOVA function removed. This can be used to judge whether this ANOVA function is making an important contribution to the model, or whether it just slightly helps to

improve the global GCV score. The fourth column gives the number of basis functions comprising the ANOVA function while the fifth column provides an estimate of the additional number of linear degrees-of-freedom used by including it. The last column gives the particular predictor variables associated with the ANOVA function.

The MARS fit is seen in Table 5a to have produced seven ANOVA functions, the first five involving only one variable ($K_m = 1$), and two comprised of two variables ($K_m = 2$). Judging from the second and (especially) the third columns, the last ANOVA function (involving an interaction between variables 2 and 6) is not very important to the fit. Its standard deviation is much smaller than that of the other ANOVA functions and removing it from the fit degrades the GCV score for the entire fit imperceptibly (see line 1). All other ANOVA functions seem important to the model in that the removal of any of them substantially degrades the quality of the fit.

After removing the unneeded (seventh) ANOVA function, the resulting MARS model is seen to be additive in variables 3, 4, and 5, and involves a two- variable interaction effect between variables 1 and 2. Note that this model shows no indication of a dependence of the response on the last five (pure noise) variables. Figure 3 shows a graphical representation of these ANOVA functions. The three additive contributions $f_4(x_4)$, $f_5(x_5)$ and $f_3(x_3)$ (25) are plotted in the first three frames. The joint contribution of the first two variables $f^*_{1,2}(x_1, x_2)$ (27) is presented in three different views of a perspective mesh plot of this function (last three frames). Low values of the corresponding variables are indicated by the position of the "0", whereas higher values are in the direction of the axis label. Note that the plotting routine does not show the bivariate function outside the convex hull of the projected data points. As discussed in Section 3.7, the MARS procedure basically extrapolates linearly beyond the boundaries of the data. It is important to note that this surface does not represent a smooth of y on x_1 and x_2, but rather it shows the contribution of x_1 and x_2 to a smooth of y on the *ten* variables $x_1, \cdots, x_{10}$.

Comparing the results of the MARS fit to these data (Table 5a and Figure 3) with the true underlying function $f(\mathbf{x})$ (61), shows that the resulting model provides a fairly accurate and interpretable description. This is especially noteworthy given the high dimensionality ($n = 10$) and the small sample size ($N = 100$).

Table 5a (and Figure 3) show results for one realization of a data set with $N = 100$. In order to assess the general ability of MARS to model data of this kind, it must be applied to a large number of realizations of this situation. Table 5b summarizes the results of running MARS on 100 replications of this example at three sample sizes ($N = 50$, 100, and 200), in the same format as Table 3. (Here the strategy of choosing the additive fit if it produced a better GCV score was not used for the $mi = 2, 10$ models.) For the very smallest sample size ($N = 50$) the additive model ($mi = 1$) is seen to actually produce more accurate fits than those involving interactions ($mi = 2, 10$), even though there are strong interaction effects (involving x_1 and x_2) in the generated data. This is due to the bias-variance tradeoff. Even though the additive model is highly biased, its lower variance leads to lower estimation errors. When the sample size is increased, however, this is no longer the case and the models involving interactions produce vastly superior accuracy.

As in the additive case (Table 3) the optimized GCV criterion is seen to overestimate the actual PSE on average (again with fairly high sample to sample variability). It is interesting to note that even though the true underlying function (61) exhibits interactions involving only two variables, the totally unconstrained MARS fit ($mi = 10$) produces models nearly as accurate as when the fit is constrained to have at most two-variable interactions ($mi = 2$).

4.4. Alternating Current Series Circuit.

Figure 4a shows a schematic diagram of a simple alternating current series circuit involving a resistor R, inductor L, and capacitor C. Also in the circuit is a generator that places a voltage

$$V_{ab} = V_o \sin \omega t \tag{62a}$$

across the terminals a and b. Here ω is the angular frequency which is related to the cyclic frequency f by

$$\omega = 2\pi f. \tag{62b}$$

The electric current I_{ab} that flows through the circuit is also sinusoidal with the same frequency,

$$I_{ab} = (V_o/Z) \sin(\omega t - \phi). \tag{62c}$$

Its amplitude is governed by the impedance Z of the circuit and there is a phase shift ϕ, both depending on the components in the circuit:

$$Z = Z(R, \omega, L, C),$$
$$\phi = \phi(R, \omega, L, C).$$

From elementary physics one knows that

$$Z(R, \omega, L, C) = [R^2 + (\omega L - 1/\omega C)^2]^{1/2}, \tag{63a}$$

$$\phi(R, \omega, L, C) = \tan^{-1}\left[\frac{\omega L - 1/\omega C}{R}\right]. \tag{63b}$$

The purpose of this exercise is to see to what extent the MARS procedure can approximate these functions and perhaps yield some insight into the variable relationships, in the range

$$\begin{aligned} &0 \leq R \leq 100 \text{ ohms} \\ &20 \leq f \leq 280 \text{ hertz} \\ &0 \leq L \leq 1 \text{ henries} \\ &1 \leq C \leq 11 \text{ micro farads.} \end{aligned} \tag{64}$$

Two hundred four-dimensional uniform covariate vectors were generated in the ranges (64). For each one, two responses were generated by adding normal noise to (63a) and (63b). The variance of the noise was chosen to give a 3 to 1 signal to noise ratio for both Z (63a) and ϕ (63b), thereby

causing the true underlying function to account for 90% of the variance in both cases. As with the previous example, the data response values were standardized.

4.4.1. Impedance Z

Applying MARS to the impedance data (63a) (64) (with 3/1 signal to noise) gave an optimizing GCV score of 0.19. The corresponding GCV Scores for an additive model ($mi = 1$) was 0.56, whereas that for $mi = 2$ was 0.19. The additive model is seen (not surprisingly from the known truth) to be inadequate. Perhaps more surprising is the fact that even though the true underlying function (63a) has interactions to all orders, an approximation involving at most two-variable interactions gives just as good a fit to these data. Owing to its increased interpretability we select the $mi = 2$ model.

Table 6a summarizes the ($mi = 2$) MARS fit. There are five ANOVA functions all of which, except for the last, seem important to the model. There is an additive contribution from R, and interactions between ωC and ωL. Figure 4b displays a graphical representation of the ANOVA decomposition. The upper left frame shows the additive contribution $f_R(R)$ (25), the upper right shows the joint contribution of ω and C, $f^*_{\omega C}(\omega, C)$ (27), while the bottom two frames show $f^*_{\omega L}(\omega, L)$ (27) from two different perspectives.

The dependence of the impedance Z (62) on the resistance R of the circuit is seen to be roughly linear. Its joint dependence on ω and C is seen to be fairly mild except when they both achieve simultaneously very low values, in which case the impedance increases very sharply. For low frequencies ω, the impedance Z is seen to be high and independent of the inductance L. For high ω, Z has a monotonically increasing dependence on L. For low L, Z monotonically decreases with increasing ω, whereas for high L values, the impedance is seen to achieve a minimum for moderate ω. These interpretations are based on visual examination of the graphical representation of the ANOVA decomposition of the MARS model, based on a sample of size $N = 200$. Since the data are in this case generated from known truth, one can examine the generating equation (63a) to verify their general correctness.

Table 6b summarizes the results of a simulation study based on 100 replications of AC circuit impedance data (63a) (64) (3/1 signal to noise) at three sample sizes ($N = 100$, 200, and 400). Additive modeling ($mi = 1$) is seen to perform badly at all sample sizes. The accuracy of the models involving interactions improves sharply with increasing sample size. The $mi = 2$ models offer slightly higher accuracy in most situations. Unlike the previous examples the GCV score is seen to underestimate the true predictive squared error PSE (59) a little on average.

4.4.2. Phase Angle φ

The MARS procedure applied to the phase angle data (63b) (64) (3/1 signal to noise) with $mi = 1, 2, 4$ gave optimizing GCV scores of 0.30, 0.22, and 0.22, respectively. Here the additive model, while still being less accurate, is more competitive with those involving interactions. The model limited to two-variable interactions ($mi = 2$) is again seen to fit the data as well as the general ($mi = 4$) MARS model. Table 7a summarizes the $mi = 2$ model. It involves nine ANOVA

functions, two of which are clearly unimportant (6 and 7), and three more that are of marginal importance (5, 8, and 9). Figure 4c shows perspective mesh plots of all six bivariate functions f^* (27) associated with the four variables. The dependence of the phase angle φ on all of the variables is seen to be more gentle and more nearly additive than the impedance Z (Figure 4b).

Table 7b gives the results of applying MARS to 100 replications of the phase angle data at $N = 100$, 200, and 400. At the smallest sample size additive fitting is seen to be almost as accurate as with interactions. This is, however, no longer true at the larger sample sizes. The optimizing GCV score is seen to slightly underestimate the true PSE (on average) for most of the situations, but as with the previous examples, the variance of the ratio (GCV/PSE) dominates this small bias.

4.5. Portuguese Olive Oil.

For this example MARS is applied to data from analytical chemistry. The observations consist of 417 samples of olive oil from Portugal (Forina, et al., 1983). On each sample, measurements were made on the concentrations of 10 acids and three sterols (see Table 8). Also recorded was the location where the sample originated. The purpose was to see if there is a relation between the chemical composition and geographical origin. Of particular interest was the extent to which olive oil from northeastern Portugal (Douro Valley – 90 samples) differed from that of the rest of Portugal (327 samples). One way to address this question is to examine the results of trying to model the probability that a sample originates from the Douro Valley given its measured chemical composition (Table 8). The response variable y in this case takes on only two values: 1 = Duoro Valley, 0 = rest of Portugal. Since $\Pr(y = 1 \mid \mathbf{x}) = E(y \mid \mathbf{x})$ one can estimate this probability through regression techniques.

Linear logistic regression (Cox, 1970) is often used when the response variable assumes only two values. The model takes the form

$$\log[p/(1-p)] = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

where p is the probability that y assumes its larger value. The coefficients $\{\beta_i\}_0^n$ are estimated by (numerically) maximizing the likelihood of the data. Hastie and Tibshirani (1986) extended this approach to additive logistic regression

$$\log[p/(1-p)] = \sum_{i=1}^{n} f_i(x_i).$$

The smooth covariate functions are estimated through their "local scoring" algorithm. The model can be further generalized to involve potential interaction effects by

$$\log[p/(1-p)] = \hat{f}(\mathbf{x}) \tag{65}$$

with $\hat{f}(\mathbf{x})$ taking the form of the MARS approximation. This can be implemented in the MARS algorithm by simply replacing the internal linear least-squares routine (LOF – Algorithm 2, line 7,

and Algorithm 3, lines 2 and 5) by one that does linear logistic regression (given the current set of multivariate spline basis functions). Unless rapid updating formulae can be derived this is likely to be quite computationally intensive. A compromise strategy, however, is likely to provide a good approximation; the multivariate spline basis functions are selected using the MARS squared-error based loss criterion, and the coefficients $\{a_m\}_0^M$ for the final model are fit using a linear logistic regression on this basis set. In this mode one takes advantage of the local variable subset selection aspect of MARS as well as its ability to produce continuous models. The detailed knot placement on the selected variables will, however, be optimally placed for the untransformed response rather than for the logistic fit.

Table 9 gives the results of six different analyses (rows) on the Portuguese olive oil data set. The first column describes the method. The first two rows are for MARS runs on the untransformed 0/1 response using its least-squares criterion. The next two rows give results when a post (linear) logistic regression is applied to the final basis function set as described above. The parameter mi is (as before) the maximum number of variables permitted in any basis function ($mi = 1$ gives an additive model; $mi = 2$ allows two variable interactions). The last two rows are (respectively) ordinary stepwise linear logistic regression and recursive partitioning [CART: Breiman, et al. (1984)]. The second column gives the number of variables that entered each final model; the third column gives the GCV estimate of the PSE (59); the fourth column gives another estimate of this quantity (CV) based on tenfold cross-validation; and the last column gives the cross-validated error rate of a prediction rule that assigns $\hat{y} = 1$ if the estimated conditional probability $\hat{E}(y \mid \mathbf{x})$ is greater than 0.5 and assigns $\hat{y} = 0$ otherwise. In the case of logistic regression the GCV and CV scores were computed using as a (squared-error) loss function

$$[y - 1/(1 + e^{-\hat{f}(\mathbf{x})})]^2,$$

with $\hat{f}(\mathbf{x})$ the corresponding MARS estimate of the log-odds (65).

Table 9 indicates that the (post) logistic transformation improves the fit substantially. The GCV and CV estimates are seen to be fairly close, especially for the (untransformed) least-squares fits. The introduction of interaction effects ($mi = 2$) seems to improve the quality of the fit, especially for the logistic models. (Increasing the value for mi beyond two did not result in further improvement.) CART and stepwise linear logistic regression do not perform as well as the logistic MARS model involving two variable interactions on these data, although (like MARS) they do indicate a strong association between geographical origin (Douro Valley versus rest of Portugal) and the chemical composition (Table 8) of the olive oil samples. This (strong) association can be expressed with a small fraction of the 13 original predictor variables. (When using GCV or CV scores for comparisons it should be remembered that they not only estimate the accuracy of the approximation (2) (3) but also include the irreducible (binomial) error. Therefore their ratios understate the actual accuracy ratios of the methods being compared.)

Table 10 provides a summary of the logistic ($mi = 2$) MARS model. There are four ANOVA functions involving three (of the 13) predictor variables. All four ANOVA functions appear important to the fit, including the last two that involve interactions. Figure 5 shows the joint contribution

(to the log-odds) of the second and twelfth variables $f^*_{2,12}(x_2, x_{12})$ (27) in the upper left frame, and $f^*_{6,12}(x_6, x_{12})$ in the upper right frame. The two lower frames show the same two plots respectively from a different perspective. The MARS model for the log-odds is (in this case) the sum of these two bivariate functions.

4.6. Low Dimensional Modeling

The main advantage of MARS modeling over existing methodology is clearly realized in high dimensional settings. It is, however, (unlike recursive partitioning) competitive in low dimensions ($n \leq 2$) as well. Friedman and Silverman (1989) studied its properties for the nonparametric smoothing problem ($n = 1$) and showed that it can produce superior performance especially in situations involving small samples and low signal to noise. (They also showed that for additive modeling (13) ($n > 1$, $mi = 1$) it is quite competitive with procedures based on the "backfitting" algorithm using local averaging smoothers.) In the univariate case ($n = 1$) the MARS method can be viewed as a modification of an approach first suggested by Smith (1982). In a massive simulation study Breiman and Peters (1988) showed that this was one of the best all around smoothers of those they tested.

In this section we illustrate the use of MARS for two-dimensional nonparametric smoothing ($n = 2$). The first example is taken from Andrews and Herzberg (1985). The data comes from an experiment on the recoiling of guns (Brouncker, 1734). A total of 109 shots were fired at different distances D between the muzzle and the target, and with varying grains of powder in the charge C. The upper left frame of Figure 6 shows the experimental design. Note that there are 40 distinct points in the design so that some points represent more than one firing. The response variable is the (standardized) distance by which the resulting shot missed the target.

The MARS procedure applied to these data resulted in a model with three basis functions and an optimized GCV score of 0.39. The upper right and lower left frames of Figure 6 show two views of the MARS surface smooth. The average shooting error is seen to increase linearly with shooting distance for all powder charges. As might be expected, at the shortest distance the error is very small and independent of the size of the powder charge. As the distance increases, the dependence of the shooting error on charge becomes nonmonotonic with the degree of nonmonotonicity increasing with shooting distance. The optimal (lowest shooting error) charge is seen to increase somewhat as the distance increases. This error is seen to be asymmetric about the minimum with the degree of asymmetry increasing with distance. For moderate to large distances it appears to be much more costly (in terms of average accuracy) to shoot with too small a powder charge than with one that is too large.

Our second example is an artificial one used by Gu, Bates, Chen, and Wahba (1990) to illustrate interaction spline smoothing (see Section 2.3) They generated 300 points (more or less) randomly from a uniform distribution in the unit square and set the response to

$$\begin{aligned} y_i = {} & 40\exp\{8[(x_{1i}-.5)^2+(x_{2i}-.5)^2]\}/\exp\{8[(x_{1i}-.2)^2+(x_{2i}-.7)^2]\} \\ & +\exp\{8[(x_{1i}-.7)^2+(x_{2i}-.2)^2]\}+\epsilon_i, \qquad 1 \leq i \leq 300. \end{aligned} \tag{66}$$

The errors ϵ_i were drawn from a standard normal distribution. Here the signal to noise is 3.15/1 so that the true underlying function accounts for about 91% of the variance of the response. Figure 7a shows a mesh plot of the true underlying function $y - \epsilon$ (66) over the unit square.

The optimized MARS model on these data consisted of 18 basis functions using an estimated 45 linear degrees-of-freedom for the fit. The corresponding surface estimate is shown in Figure 7b. Although this estimate is a bit smoother than the one produced by interaction splines (see Gu et al., 1990, Fig. 4) they both have nearly the same accuracy in terms of expected squared error (3) (29). Since this is a relatively well behaved function and the sample size is quite large ($N = 300$) it is likely that methods based on kernel smoothing (Cleveland and Devlin, 1988) and (nonadaptive) tensor product splines (de Boor, 1978) would also do well in this case.

Although MARS is competitive with other methodologies in low dimensions, what sets it apart is its ability to handle problems involving many variables. Suppose the variables (x_1, x_2) in (66) were two out of ten variables, all of which (jointly) effect the response y in an unknown way, and the goal is to estimate the dependence of the response on x_1, x_2, accounting for the effect of the other eight (nuisance) variables $x_3, \cdots, x_{10}$. To get an idea of how well MARS might do in such a problem, $N = 300$ points were generated in the $n = 10$ dimensional unit hypercube. The dependence of this response $\tilde{y}$ was taken to be

$$\tilde{y}_i = y_i + f(\mathbf{x}_i) \tag{67}$$

with y_i given by (66) and $f(\mathbf{x})$ given by (61) (shifted by two arguments). Thus, the dependence of $\tilde{y}$ on the first two variables is the same as in the previous two-dimensional example, whereas its dependence on the last eight is the same as the first eight variables of the problem studied in Section 4.3. The sample size ($N = 300$) and the pure noise level are the same as in the two-dimensional problem. The apparent noise in the $x_1 x_2$-plane is now however many times greater owing to the variability induced in the response by the many nuisance variables. It is hoped that one can account for the variance associated with the nuisance variables by fitting a (nonparametric) model jointly with respect to them and the variables of interest, thereby obtaining a better estimate of the dependence of the response on x_1, x_2.

Applying MARS to this ten dimensional data resulted in a model with 27 basis functions using an estimated 68.5 linear degrees-of-freedom. Eleven of the basis functions were associated with the dependence on variables x_1 and x_2, accounting for 27.5 linear degrees-of-freedom. Figure 8a shows the resulting surface estimate. Although it is not quite as accurate as the smooth (Figure 7b) produced in the absence of the eight nuisance variables, it still gives a good indication of the nature of the joint dependence of the response on x_1 and x_2. The estimate (Figure 8a) is smoother than that of Figure 7b owing to the fact that it is based on 11 rather than 18 basis functions. Figure 8b shows plots of all of the ANOVA functions produced by the MARS fit to the 10-dimensional data. The estimates corresponding to the (nuisance) variables associated with $f(\mathbf{x})$ (61) (67) are actually better than those obtained in the example of Section 4.3. This is the result of having 300 observations here, whereas only 100 were used in Section 4.3.

4.7. Slicing a Mars Model

In the examples so far presented the dominant interactions involved at most two variables. The resulting MARS models could then be visualized through plots of the contributing ANOVA functions (25) (27). When substantial interaction effects involving more than two variables are present the model becomes more difficult to interpret. This section describes an interpretational tool called "slicing" that can aid in the visualization of such models.

The MARS model (23) (34) is a sum of products of univariate functions

$$\hat{f}(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{k=1}^{K_m} b_{km}(x_{v(k,m)}). \tag{68}$$

Here b_{km} is either a $q = 1$ spline basis function (33) or its cubic analog (34). If the number of factors K_m in any product is greater than two then $\hat{f}(\mathbf{x})$ is more difficult to interpret since the ANOVA decomposition (24) contains functions of more than two variables which are difficult to plot or visualize. It is possible, however, to get a rough idea of the behavior of $\hat{f}(\mathbf{x})$ by reducing the dimensionality of the predictor variable space by repeatedly conditioning on subsets of the variables.

Let $\mathbf{Z}$ represent a (selected) subset of the predictor variables $\{x_1 \cdots x_n\}$, with dimension d ($< n$), and $\tilde{\mathbf{Z}}$ the complement subset of dimension $n - d$. Define a d-dimensional *slice* of the predictor variable space by simultaneously assigning specific values to the selected variables

$$\text{slice} : \{Z_1 = z_1, \cdots, Z_d = z_d\} \equiv \mathbf{Z} = \mathbf{z}. \tag{69}$$

The MARS model along the slice will be a function of the variables $\tilde{\mathbf{Z}}$ complement to those defining the slice

$$\hat{f}(\mathbf{x} \mid \mathbf{Z} = \mathbf{z}) = \hat{f}(\tilde{\mathbf{Z}} \mid \mathbf{z}) = a_0 + \sum_{m=1}^{M} a_m \prod_{k=1}^{K_m} b_{km}(x_{v(k,m)} \mid \mathbf{Z} = \mathbf{z}). \tag{70}$$

The particular form of the MARS model (68) makes the sliced model (70) especially straightforward to compute by decomposing its products into those factors involving variables defining the slice, and those involving the complement variables,

$$\hat{f}(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{x_{v(\ell,m)} \in \mathbf{Z}} b_{\ell m}(x_{v(\ell,m)}) \prod_{x_{v(k,m)} \in \tilde{\mathbf{Z}}} b_{km}(x_{v(k,m)}). \tag{71}$$

For a given slice ($\mathbf{Z} = \mathbf{z}$) the first product in (71) evaluates to a constant (multiplying the coefficient a_m) and the second product gives the dependence on the complement variables. The MARS model along the slice can therefore be represented as

$$\hat{f}(\tilde{\mathbf{Z}} \mid \mathbf{z}) = c_0(\mathbf{z}) + \sum_{m=1}^{\tilde{M}} c_m(\mathbf{z}) \prod_{k=1}^{\tilde{K}_m} b_{km}(\tilde{\mathbf{Z}}_{v(k,m)}) \tag{72a}$$

with

$$c_0(\mathbf{z}) = a_0 + \sum_{\{x_{v(\ell,m)}\} \subseteq \mathbf{Z}} a_m \prod b_{\ell m}(z_{v(\ell,m)}), \tag{72b}$$

$$c_m(\mathbf{z}) = \sum_{\{x_{v(k,m)}\} \subseteq \tilde{Z}} a_m \prod_{x_{v(\ell,m)} \in \mathbf{Z}} b_{\ell m}(z_{v(\ell,m)}). \tag{72c}$$

The sum in (72b) is over all basis functions in (68) (71) that involve only variables defining the slice. The sum in (72c) is over all basis functions that involve exactly the same complement variables $\{\tilde{\mathbf{Z}}_{v(k,m)}\}$.

For a given slice (69) the MARS model along the slice (72a) has the same (constructive) form as any ordinary MARS model (68), and thus has a corresponding ANOVA decomposition that can be interpreted graphically as illustrated in the previous sections. This suggests the following strategy for trying to visualize models that contain interactions involving more than two variables:

(1) Use the ANOVA decomposition of the full MARS model $\hat{f}(\mathbf{x})$ (24) to identify those variables that participate in interactions with more than one other variable.

(2) Choose a variable subset $\mathbf{Z}$ for slicing, such that the MARS model along the slice $\hat{f}(\tilde{\mathbf{Z}} \mid \mathbf{z})$ involves (at most) two-variable interactions.

(3) Examine graphically the ANOVA decomposition (25) (27) of $\hat{f}(\tilde{\mathbf{Z}} \mid \mathbf{z})$ for various values of the slice $(\mathbf{Z} = \mathbf{z})$.

This slicing strategy is illustrated on testing data taken from a semiconductor component. The predictor variables $V_1 \cdots V_4$ are the simultaneous voltages applied to the terminals of a four terminal semiconductor resistor in the ranges $-6 \leq V_1 \leq 6$, $-.75 \leq V_2 \leq 10.75$, and $-.5 \leq (V_3, V_4) \leq 5.5$. The response is the current I into one of the terminals. There were 599 observations taken. Table 11 provides a summary of the results of four MARS runs on these data. The first column gives the maximum number of variables mi allowed to participate in interactions. The second column gives the GCV estimate (30) (32) of the $\sqrt{PSE}$ (59) and the third gives another estimate of this quantity based on 10-fold cross-validation. The last three columns give respectively the median, 75th percentile, and maximum values of the distribuiton of the absolute cross-validated residuals divided by the absolute deviation of the response from its mean value. Table 11 shows results only for piecewise-linear fits; in all cases the corresponding piecewise-cubic models gave rise to much larger values and thus worse fits.

Increasing the permissible interaction level is seen to improve the general quality of the fit. The GCV score appears to rather dramatically underestimate the PSE as estimated from cross-validation. Inspection of the (cross-validated) residual distributions reveals that they are highly skewed toward large values. The GCV score is seen to reflect the size of the typical residuals (median - 75% point) but not the few extremely large ones. Increasing the interaction level beyond two seems to preferentially reduce the larger residuals.

Figure 9a shows a graphical representation of the ANOVA decomposition for the MARS model involving ony two-variable interactions ($mi = 2$). The resistor is seen to be a very nonlinear device. The current appears to be roughly independent of the terminal voltages except when one or more of them take on extreme values where the current changes rapidly.

As seen in Table 11 the MARS model involving four-variable interactions provides a subtantially better fit to these data. Figures 9b and 9c explore this function of four variables using the

slicing strategy described above. Figure 9b shows the current as a function of V_1 and V_2 along four slices defined by V_3 and V_4. In this figure the functions are not plotted on the same vertical scale. In order to relate the scales, the maximum value of each function relative to that of the first (upper left frame) is shown above each plot. Since the relative locations of the plots are arbitrary they are each plotted to have a minimum value of zero, so that the maximum value is equal to the range. The (V_3, V_4) slices are taken at the four extreme corners of the $V_3 - V_4$ design. Both the magnitude and shape of the dependence of the current on V_1 and V_2 are seen to depend rather strongly on the values of V_3 and V_4. For simultaneously low values of V_3 and V_4 (upper left) the dependence is seen to be roughly linear, whereas when V_4 takes on its highest value, with V_3 at its lowest value (lower left) the current is seen to vary much less as a function of V_1 and V_2. For high values of V_3 the dependence is similar to that of the lower left frame except for the existence of the dramatic peak for low values of V_2.

Figure 9c shows the dependence of the current (as reflected by the MARS fit) on V_3 and V_4 for four slices on (V_1, V_2). Here the slices do not include the two extreme corners on V_1 for low values of V_2 since they are outside the support of the design. (This can be seen on the $V_1 - V_2$ plots; the functions are plotted only within the convex hull of the bivariate distributions.) In Figure 9c the functions are all plotted on the same scale. As would be expected from the previous results, the dependence of the current on V_3 and V_4 changes substantially with differing values of V_1 and V_2.

Exploring the resulting MARS model in this manner provides some insight into the nature of the cross-validated residual distributions observed in Table 11. The function has very high (and increasing) first and second derivatives very near some of the (joint) boundaries. When extreme observations on these boundaries are deleted during the cross-validation, the resulting slopes are underestimated and the extrapolation to the left out observation gives rise to a large error. This phenomenon also explains why the piecewise-linear models give rise to much better fits. There are clearly small local regions where the second derivatives are very large. By approximating these by infinite second derivatives the piecewise-linear model is able to come closer than the piecewise-cubic fit which tries to moderate these locally very high second derivatives.

Figures 9b and 9c represent a small subset of all possible revealing slices of this four-variable function. In general, slicing is likely to be most effective when performed in an interactive manner. Functional dependencies revealed by inspecting the results of a particular slice will likely suggest further slices to try. The straightforward and rapid calculation of sliced models (72) from the complete MARS model (68) (71) might make feasible the computation of sliced functions in real time on modern workstations. In this case the values defining the slice could be defined (and changed) in a continuous manner through a graphical input device (such as by moving a mouse) and the continuously changing functions along the slice can be viewed in real time.

5.0. Remarks.

This section covers various aspects (extensions, limitations, etc.) of the MARS procedure not discussed in the previous sections.

5.1. Constraints.

The MARS procedure is nonparametric in that it attempts to model arbitrary functions. It is often appropriate, however, to place constraints on the final model, dictated by knowledge of the system under study, outside the specific data at hand. Such constraints will reduce the variance of the model estimates, and if the outside knowledge is fairly accurate, not substantially increase the bias. One type of constraint has already been discussed in Section 4, namely limiting the maximum interaction order (mi) of the model. Setting this maximum to a small value ($mi \lesssim 2$) causes the resulting (restricted) model to be more interpretable since its ANOVA components (Section 3.5) can be directly visualized graphically. With this restriction the MARS model has the same form as a low dimensional expansion (additive model, interaction splines – see Section 2.3). Unlike those methods however, which require the variable subsets to be preselected (in advance), MARS automatically selects them separately for each problem at hand based on the data. It also automatically (and adaptively) selects the (different) degree of smoothing to be applied in estimating the separate functions of each of the variable subsets it produces. In situations where this adaptability is not important one might apply MARS to obtain the low dimensional variable subsets (ANOVA decomposition) and then apply a less adaptive smoothing method (kernel with the backfitting algorithm, or interaction splines (12)) on these subsets to obtain the final function estimates.

One might in addition (or instead) limit the specific variables that can participate in interactions. If it is known a priori that certain variables are not likely to interact with others, then restricting their contributions to be at most additive can improve accuracy. If one further suspects that specific variables can only enter linearly, then placing such a restriction can improve accuracy. The incremental charge d (32) for knots placed under these constraints should be less than that for the unrestricted knot optimization. (The implementing software charges $2d/3$ and $d/3$ respectively, for the additive and linear constraints where d is the charge for unrestricted knot optimization.)

These constraints, as well as far more sophisticated ones, are easily incorporated in the MARS strategy. Before each prospective knot is considered (Algorithm 2, lines 6–8), the parameters of the corresponding two new potential multivariate spline basis functions (v, t, and B_m) can be examined for consistency with the constraints. If they are inconsistent, they can be made ineligible for inclusion in the model by simply skipping lines 6–8 in Algorithm 2.

5.2. Semiparametric Modeling

Another kind of a priori knowledge that is sometimes available has to do with the nature of the dependence of the response on some (or all) the predictor variables. The user may be able to provide a function $g(\mathbf{x})$ that is thought to capture some aspects of the true underlying function $f(\mathbf{x})$. More generally, one may have a set of such functions $\{g_j(\mathbf{x})\}_1^J$, each one of which might capture some special aspect of the functional relationship. A semiparametric model of the form

$$\hat{f}_{sp}(\mathbf{x}) = \sum_{j=1}^{J} c_j g_j(\mathbf{x}) + \hat{f}(\mathbf{x}), \tag{73}$$

where $\hat{f}(\mathbf{x})$ takes the form of the MARS approximation, could then be fit to the data. The coefficients c_j in (73) are jointly fit along with the parameters of the MARS model in Algorithms 2 and 3. To the extent that one or more of the g_j successfully describe attributes of the true underlying function, they will be included with relatively large (absolute) coefficients, and the accuracy of the resulting (combined) model will be improved.

Semiparametric models of this type (73) are easily fit using the MARS strategy. One simply includes $\{g_j(\mathbf{x})\}_1^J$ as J additional predictor variables $(x_{n+1}, \cdots, x_{n+J})$ and constrains their contributions to be linear. One could also, of course, not place this constraint, thereby fitting more complex semiparametric models than (73).

Another strategy that is often employed in this context is to first fit only the parametric component to the data and then apply a nonparametric method (such as MARS) to the residuals of the parametric fit. In general, this strategy is likely to be less successful because the residual function may be more highly structured than the original one (and thus more difficult to approximate) especially if the parametric approximation is not close to the true underlying function. The more general approach (73) allows the fitting procedure to automatically adjust the strength of the parametric components as part of the fitting process.

5.3. Collinearity

Extreme collinearity of the predictor variables is a fundamental problem in the modeling of observational data. Solely in terms of predictive modeling it represents an advantage in that it effectively reduces the dimensionality of the predictor variable space. This is only true provided that the observed collinearity is a property of the population distribution and not an artifact of the sample at hand. Collinearity presents, on the other hand, severe problems for interpreting the resulting model.

This problem is even more serious for (interactive) MARS modeling than for additive or linear modeling. Not only is it difficult to isolate the separate contributions of highly collinear predictor variables to the functional dependence, it is also difficult to separate the additive and interactive contributions among them. A highly nonlinear dependence on one such (highly correlated) variable can be well approximated by a combination of functions of several of them, and/or by interactions among them.

In the context of MARS modeling one strategy to cope with this (added) problem is to fit a sequence of models with increasing maximum interaction order (mi). One first fits an additive model ($mi = 1$), then one that permits at most two variable interactions ($mi = 2$), and so on. The models in this sequence can then be compared by means of their respective optimizing GCV scores. The one with the lowest mi value that gives a (relatively) acceptable fit can then be chosen.

Another (complementary) strategy is to directly resolve the ambiguity by enforcing parsimony on the number of variables that enter the model (Friedman and Silverman, 1989). This will discourage spurious interaction effects caused by collinearity (or concurvity) and, in addition, partially stabilize the function estimates. It will also aid in interpretation in that the resulting approximation will involve fewer variables.

Variable parsimony can be accomplished by introducing a small incremental penalty to the lack-of-fit criterion for choosing factors (knots) that involve introducing a new variable (not already in the model) as part of the forward stepwise procedure (Algorithm 2, line 7):

$$LOF(g) \leftarrow LOF(g)\left[1 + \gamma I\left(v \notin \bigcup_{m=1}^{M-1} \{v(k,m)\}_1^{K_m}\right)\right]. \tag{74}$$

At the Mth iteration there are $M-1$ basis functions currently in the model and the indicator function in (74) will be zero if the vth variable appears in a least one of them; otherwise it will be equal to one. The parameter γ (> 0) regulates the strength of the penalty for entering new variables and can be used to control the lack-of-fit/variable parsimony tradeoff. Note that this penalty (74) is only introduced as part of the (forward stepwise) knot selection process and it is not used to reflect the overall lack-of-fit of the model.

In highly collinear settings the (unmodified) lack-of-fit criterion ($LOF(g)$) has very little preference on which particular variable to enter from a highly correlated set, and a small value for γ will cause the modified criterion (74) to repeatedly enter the same one from that set, without seriously degrading the quality of the approximation. A good value for γ depends on the particular situation (degree of collinearity) and how much goodness-of-fit the user is willing to sacrifice for variable parsimony. This can be judged by examining the resulting fit quality for several (increasing) values of γ as reflected by either the final GCV score (30) or a sample reuse method.

We illustrate these two approaches on data taken from the Places Rated Almanac (Boyer and Savageau, 1986). They rated 329 American cities on the nine criteria listed in Table 12. For this exercise we attempt to model housing cost ($y = x_2$) on the other eight criteria. Table 13 shows the resulting number of variables and GCV estimate (30) of the PSE (59) for running MARS with different values of γ (74). The first three rows are for additive modeling ($mi = 1$) and the second three for models with two variable interactions permitted ($mi = 2$). The models involving interactions are seen to not be distinctly superior to the additive ones, so that using the first strategy (above) one would be inclined to choose the latter. As the value of γ (74) is steadily increased, MARS produces models with progressively fewer variables, as one would expect. For these particular data, however, one is able to reduce the number of variables from (nearly) the full set (at $\gamma = 0$) to only three ($.05 \leq \gamma < .15$) without seriously degrading the quality of the fit as estimated by the solution GCV score. Note that this GCV score (30) does not reflect the additional penalty imposed by setting $\gamma > 0$, so that differences between scores involving larger and smaller values of γ underestimate (on average) their actual differences. Ordinary cross-validation (CV) does account for this increased penalty. For example, the CV estimate (10 replications) for the $\gamma = 0$ ($mi = 1$) model is 0.56 whereas the corresonding score for $\gamma = 0.1$ is 0.52.

Figure 10 shows the graphical ANOVA decomposition for the three variable additive model produced for $0.05 \leq \gamma < .15$. From this analysis it appears that average (increasing) housing costs are most strongly affected by increasingly good climate (especially for the highest values) and are associated to a somewhat lesser degree with economic conditions and access to the arts.

The dependence on climate might be somewhat surprising since in these data housing costs reflect utility bills, which are likely to decrease with good climate, as well as taxes and mortgage payments. Any interpretations, however, must be tempered by the existence of the collinearities present in the design and the fact that the model is estimated to account for only 50% of the variance of the response.

5.4. Robustness

Since the MARS method as described here uses a model selection criterion based on squared-error loss it is not robust against outlying response values. There is nothing fundamental about squared-error loss in the MARS approach. Any criterion can be used to select the multivariate spline basis functions, and construct the final fit, by simply replacing the internal linear least squares fitting routine (LOF – Algorithm 2, line 7, and Algorithm 3, lines 2 and 5) by one that minimizes another loss criterion (given the current set of multivariate spline basis functions). Using robust/resistant linear regression methods would provide resistance to outliers. The only advantage to squared-error loss in the MARS context is computational. It is difficult to see how rapid updating formulae (Section 3.9) could be developed for other types of linear regression.

Gross outliers (in both the response and covariates) that can be detected through a preliminary (exploratory) analysis of the data, should be considered for removal before applying MARS. The MARS procedure is less sensitive than linear regression to covariate outliers owing to the local nature of the fit; sample covariate vectors far from an evaluation point tend to have less rather than more influence on the model estimate. Covariate outliers can have a strong influence on the fit near the corresponding data boundaries. This can be quite helpful if the corresponding response values for the outliers are correctly measured. If not, these outliers will contribute to end effect errors.

Recursive partitioning responds to outlying response values by trying to isolate them. It produces a series of splits so as to place each such outlier in its own region. This localizes the effect of the outlier(s) so that they only distort the fit for covariate values close to that of the outlier(s). The MARS procedure operates similarly. It will also try to isolate outliers through a series of corresponding "splits" producing basis functions that attempt to capture the (apparent) high curvature of the function near each outlier. The outliers will tend to heavily influence the values of the coefficients of their corresponding basis functions, but have much less influence on the rest of the fit. The particular basis functions introduced in this manner by outlying response values, may tend to involve interactions of high order depending on their location in the covariate space. Thus, interpreting the presence of interaction effects can be highly distorted by the existence of outlying response values.

Computationally feasible methods of robustifying the MARS procedure are currently under investigation.

6.0. Conclusion

The aim of the MARS procedure is to combine recursive partitioning and spline fitting in a

way that best retains the positive aspects of both, while being less vulnerable to their unfavorable properties. This has been accomplished, at least to some extent. The greatest strength of recursive partitioning is its adaptability, through its *local* variable subset selection strategy. This makes it a highly dynamic computation (Section 2.4) capable of tracking the dependencies associated with a wide variety of complex functional forms. The two weaknesses of recursive partioning are the lack of continuity of its models, and its inability to capture simple relationships such as linear, additive, or interactions of low order compared to n. Nonadaptive (tensor product) spline fitting produces continuous models with continuous derivatives. It strongly suffers, however, from the "curse-of-dimensionality" in that very large basis function sets are usually required in high dimensions to capture relatively simple functional relationships.

The MARS procedure completely retains the adaptability of recursive partitioning by its close adherence to the recursive splitting paradigm (compare Algorithms 1 and 2). It is in fact much more adaptive because it permits the recursive "splitting" of all basis functions (nodes) in the model and not just those that are currently terminal. This causes it to overcome the second problem (mentioned in the previous paragraph) associated with recursive partitioning. It produces continuous models by replacing the step functions (19) (20) by $q = 1$ truncated power spline basis functions (21) (22). Continuous derivatives are obtained through the strategy outlined in Section 3.7. From the point of view of tensor product spline methods, MARS can be viewed as a hierarchical forward/backward stepwise subset selection procedure for choosing a subbasis appropriate for the problem at hand, from the complete ($q = 1$) truncated power tensor product basis with knots at every (distinct) marginal data value. MARS models have a fair degree of interpretability through the ANOVA decomposition (Section 3.5) that breaks up the approximation into an additive component and into interaction contributions of various orders. Slicing (Section 4.7) can be used to explore the higher dimensional aspects of the models

The implementation of the adaptive regression spline strategy presented here represents a "first attempt" in that particular choices have been made concerning many of the "engineering details" in the absence of a great deal of experience with the procedure. Although incidental to the fundamental ideas, these details can have a strong bearing on performance. As experience is gained it is likely that many of the choices taken here will be seen to be less than optimal and suitable modifications will emerge that improve the performance of the procedure. The attempt here has been to demonstrate that the adaptive regression spline strategy, first introduced by Smith (1982) (in the univariate setting), holds substantial promise as a tool for multivariate function estimation.

A FORTRAN program implementing the MARS procedure is available from the author.

Acknowledgements

Helpful discussions with Terry Therneau are gratefully acknowledged. The author is grateful to Kishore Singhal for providing the semiconductor test data used in Section 4.7.