Automatic Smoothing Spline Projection Pursuit *

Charles B. Roosen
Department of Statistics
Stanford University

Trevor J. Hastie
Statistics and Data Analysis Research Department
AT&T Bell Laboratories

December 12, 1993

Abstract

A highly flexible nonparametric regression model for predicting a response y given covariates $\{x_k\}_{k=1}^d$ is the projection pursuit regression (PPR) model $\hat{y} = h(\mathbf{x}) = \beta_0 + \sum_j \beta_j f_j(\boldsymbol{\alpha}_j^T \boldsymbol{x})$, where the f_j are general smooth functions with mean zero and norm one, and $\sum_{k=1}^d \alpha_{kj}^2 = 1$. The standard PPR algorithm of Friedman and Stuetzle (1981) estimates the smooth functions f_j using the supersmoother nonparametric scatterplot smoother. Friedman's algorithm constructs a model with M_{max} linear combinations, then prunes back to a simpler model of size $M \leq M_{max}$, where M and M_{max} are specified by the user. This paper discusses an alternative algorithm in which the smooth functions are estimated using smoothing splines. The direction coefficients $\boldsymbol{\alpha}_j$, the amount of smoothing in each direction, and the number of terms M and M_{max} are determined to optimize a single generalized cross-validation measure.

*To appear as: Roosen, C. B. and Hastie, T. J. (1994). Automatic Smoothing Spline Projection Pursuit. *Journal of Computational and Graphical Statistics*, **3**, 235–248.

1 Introduction

A great deal of interest has arisen in the statistical and connectionist learning communities in using highly flexible nonlinear models to predict a response y given covariates $\{x_k\}_{k=1}^d$. A class of models which has the ability to implement a wide variety of functions is the class of models consisting of sums of functions of linear combinations of the predictors. That is, models of the form:

$$\hat{y} = h(\mathbf{x}) = \beta_0 + \sum_{j=1}^{M} \beta_j f_j(\boldsymbol{\alpha}_j^T \boldsymbol{x})$$

In the connectionist learning literature, the functions f_j are usually taken to be fixed functions, with one of the most common being the sigmoidal function $f(z) = \frac{1}{1+e^{-z}}$ (Hertz et al., 1991). In the nonparametric regression literature, one approach is to estimate the functions f_j using scatterplot smoothers, with the constraints $E(f_j) = 0$, $E(f_j^2) = 1$, and $\sum_{k=1}^d \alpha_{kj}^2 = 1$ for identifiability of the model components. This yields a Projection Pursuit Regression (PPR) model (Friedman and Stuetzle, 1981).

The standard PPR algorithm proposed by Friedman and Stuetzle (1981) and refined by Friedman (1984a) estimates the smooth functions f_j using a two-dimensional scatterplot smoother known as the supersmoother (Friedman, 1984b). The supersmoother is a variable span smoother based on local linear fits, in which local cross-validation is used to estimate the optimal span as a function of abscissa value. The algorithm proceeds by finding a model with M_{max} terms (i.e. M_{max} linear combinations and smooth functions), and then pruning the model back to a total of $M \leq M_{max}$ terms. The user need not specify smoothing parameters, but must set the values of M and M_{max}.

Friedman et al. (1983) propose a regression spline PPR algorithm which they call the Multidimensional Additive Spline Approximation (MASA). In this algorithm, the user specifies the number of terms M to add, and the number of knots to use in each regression spline. Another PPR algorithm is that of Hwang et al. (1993). They estimate the smooth functions using high degree polynomials (in the examples they present, they use polynomials of degree 7 or 9). The same degree polynomial is used for each term in the fit. Again, the user specifies M and M_{max}.

This paper presents an algorithm in which the functions are estimated by smoothing splines. Hwang et al. note that two desirable qualities of polyno-

mials are that they provide a fast and accurate derivative calculation, and they provide a smooth interpolation (1993, p. 11). Smoothing splines also have these properties, and in addition can be used to choose the smoothing parameters and the number of terms automatically. The local smoothness of the fit provided by smoothing splines will in many cases make them more reliable than the supersmoother. The automatic selection of smoothing parameters and number of terms will hopefully guard against model misspecification that may be present under any of the algorithms which require user-specified parameters. This algorithm is a direct descendant of the non-adaptive smoothing spline PPR algorithm described in Roosen and Hastie (1993).

This paper first discusses the generalized cross-validation (GCV) based techniques used to select the smoothing parameters and number of terms. It then describes the structure of the Automatic Smoothing Spline Projection Pursuit (ASP) algorithm. Finally it presents examples comparing ASP with the standard supersmoother-based PPR algorithm of Friedman.

2 Automatic Parameter Selection

2.1 Smoothness Parameters

2.1.1 Estimating a Single Smoothing Parameter

Each time ASP computes a smooth term in a given direction, it needs to decide how much smoothing should be done. We first review this selection for simple bivariate smoothing using smoothing splines.

The penalized least squares fitting criterion for a smoothing spline is:

$$\sum_{i=1}^{n}\{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt \qquad (1)$$

The smoothing spline is fit to minimize this criterion. The second component is a roughness penalty, which penalizes for large curvature, and hence for changes in slope. Overall the criterion trades off fidelity to the data with smoothness, via the parameter λ. For small values of λ more weight is given to fitting the squared error portion of the criterion, and a rough fit results. As λ increases, more weight is given to keeping the curvature small, and

a smoother fit results. For a fixed value of λ, the solution to (1) is an n-dimensional natural cubic spline, and computing the fit reduces to a large generalized ridge-regression problem.

The ASP algorithm is implemented in S using the convenient smoothing spline algorithm `smooth.spline()`. This algorithm fits a cubic smoothing spline, with computations linear in the number of data points n. For $n < 50$ a knot is placed at every distinct data point, while for larger n the number of knots is chosen judiciously to keep the computation time manageable. This algorithm is based on the FORTRAN code of Finbarr O'Sullivan, with subsequent modifications by the second author (Chambers and Hastie, 1992, pp. 574–576). `Smooth.spline()` minimizes the GCV statistic to automatically select λ. GCV asymptotically minimizes mean squared error for estimating f (Craven and Wahba, 1979; Silverman, 1985), and is defined as:

$$\mathrm{GCV}(\lambda) = \frac{\mathrm{RSS}(\lambda)/n}{(1 - df_\lambda/n)^2}$$

where RSS is the residual sum of squares, n is the sample size, and df_λ is the *effective number of parameters* or *degrees of freedom* used in the fit. For a rough, near interpolating, fit RSS will be small; but then so will the denominator, since a rough fit will likely require a large df_λ. The reverse occurs for an oversmooth fit, and hence GCV trades off smoothness with fidelity to the data. If we write the smooth as $\hat{y} = S_\lambda \cdot y$, then df_λ is defined by: $df_\lambda = tr(S_\lambda)$. This definition is one of several that generalize the usual notion of degrees of freedom of a linear model (Hastie and Tibshirani, 1990). Note that when $\lambda = \infty$, the fit is linear, and $df_\infty = 2$; likewise $df_0 = n$.

2.1.2 Estimating Multiple Smoothing Parameters

As we are not estimating a single smooth, but rather a number of smooths (with differing smoothing parameters), we need to take this into account when calculating the GCV score. To determine the appropriate formulation of GCV, first suppose the directions and smoothing parameters were given. The procedure would then be a linear procedure, and hence we could use a GCV criterion analogous to that for a single smoothing spline:

$$\mathrm{GCV}(\gamma) = \frac{\mathrm{RSS}(\gamma)/n}{(1 - df_\gamma/n)^2} \tag{2}$$

where γ indexes the set of directions and smoothing parameters, and $df_\gamma = tr(S_\gamma)$ with S_γ the ASP operator matrix using these parameters. In principle one could try to optimize this criterion over all directions and smoothing parameters, which leads to a messy nonlinear optimization problem. We approach this optimization by approximating df_γ, and performing the optimization in a cyclic manner. This approach was used by the second author (1989) in the similar context of additive models.

As a simplifying approximation, we use $df_\gamma = M \cdot d + \sum_{j=1}^{M} df_\lambda^j$, where df_λ^j is the trace of the smoothing matrix for smooth j. For each term, we roughly charge $d-1$ degrees of freedom for the linear combinations $\boldsymbol{\alpha}_j$, and 1 degree of freedom for the scale coefficient β_j. Our implementation of the algorithm allows the user to modify the charge per term, but preliminary studies suggest that the charge of $d + df_\lambda^j$ degrees of freedom per term is appropriate.

With this assumption regarding df_γ, we perform the optimization over directions and smoothing parameters by cycling through each direction, freezing the parameters for all but one term j and minimizing the partial residuals for the remaining term. Let $r_{ij} = y_i - \beta_0 - \sum_{k \neq j} \beta_k f_k(\boldsymbol{\alpha}_k^T \boldsymbol{x}_i)$ denote the partial residual excluding term j from the fit, and $df_{\text{offset}} = M \cdot d + \sum_{k \neq j} df_\lambda^k$ denote the degrees of freedom for all model components except the smooth f_j under consideration. We may reformulate (2) as:

$$\text{GCV}(\gamma) = \frac{\sum_i \{r_{ij} - f_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_i)\}^2 / n}{\{1 - (df_\lambda^j + df_{\text{offset}})/n\}^2}$$

where $f_j = S_\lambda^j \cdot r_j$ and $df_\lambda^j = tr(S_\lambda^j)$, absorbing the scale coefficient β_j into the smooth function. Note that r_{ij} and df_{offset} are constant with respect to the smoothing parameter λ used in f_j. This formulation of the criterion leads to the simple univariate smoothing spline GCV criterion with a small modification to the degrees of freedom. Hence the estimation of f_j may be performed using a standard univariate spline GCV procedure which has been slightly modified to adjust the degrees of freedom for the constant df_{offset}. When fitting each individual smooth, the GCV criterion accounts for all other parameters in the model. This differs from the supersmoother based algorithm, in which univariate smooths are performed without taking total model complexity into consideration.

2.1.3 Detecting and Correcting for Failure of GCV

Hastie and Tibshirani (1990, p. 50) note that GCV tends to undersmooth, and that it is not unusual for it to fail dramatically by interpreting short trends in the plot of Y against X as high-frequency structure. Although this behavior is the exception rather than the norm, it is a concern in the ASP algorithm. As new fits are performed on the working residuals from previous terms, correlation in the residuals may be interpreted as structure. To prevent catastrophic overfitting, it is necessary to determine which fits dramatically undersmooth, and replace them with smoother fits. The approach we take here is to consider a wildly fluctuating function as having too much sharp local structure. We count the local minima, and if a fit has more than a specified number of local minima (with a default of 10), then we replace the fit with a $df_\lambda = 4$ smooth. (A different small value of df_λ could be used equally well, and the allowable number of local minima should scale with the sample size.) The idea is to fit a relatively gentle term to remove the structure confusing the GCV criterion. If more degrees of freedom are truly needed in the specified direction, additional terms may be added along this direction later. Often the $df_\lambda = 4$ term is useful for facilitating the discovery of other directions, but is itself dropped during a pruning step.

An alternative approach would be to threshold the value of df_λ at some maximum. This approach has two flaws. One is that a large df_λ value does not necessarily correspond to a rough function. It may alternately correspond to a very smooth function which is not easily represented by piecewise polynomials (e.g. $\tanh(x)$). Hence we cannot look at df_λ alone as a measure of roughness. Secondly, when GCV grossly overfits, it is a sign of the failure of the criterion on the data under consideration, rather than a sign that a relatively large df_λ value is appropriate. Again, if a large df_λ is indeed appropriate, this may be achieved by the inclusion of additional terms along the same direction at a later step.

2.2 Number of Terms

The user may specify M and M_{max} if desired. Otherwise, we also monitor GCV in order to determine how many terms to add and which size model to return. The goal in adding terms is to add a few more terms than the final number needed. The rule of thumb used by Hwang et al. is to set

$M_{max} = M + 2$. This appears to be a useful heuristic. As Friedman notes in the SMART paper (1984a, p. 5), adding more terms than necessary and using backwards stepwise model selection helps to avoid local minima.

In the forward stepping, terms are added until two consecutive increases in GCV are obtained. A local minimum often occurs in which a single increase is followed by a decrease, but the convention of considering two consecutive increases as denoting having passed the minimum seems to work well. This also agrees with going two terms past the desired final number of terms.

After the maximum term is added, we then prune the model back, dropping terms until we reach the model of size one term less than the model at which the first local minimum in GCV was observed during the forward procedure. In testing the algorithm, additional pruning provided no benefit. This gives a sequence of models obtained during pruning. The model with the lowest GCV at this point is then deemed the best model. Note that during pruning the GCV values will change, so the model with minimum GCV after pruning may have a different number of terms than were in the model with minimum GCV during the forward procedure.

3 The ASP Algorithm

This section presents a smoothing spline version of the PPR algorithm. The general approach is to fit the terms in a stepwise manner, with a good deal of backfitting between the addition of terms. The overall algorithm structure is presented, after which the components of the algorithm are described.

3.1 Algorithm Structure

INITIALIZE TERMS

$\beta_0 \leftarrow \bar{y}$, $\beta_j \leftarrow 1$, $f_j \leftarrow 0$

ADD TERMS

While (not yet two consecutive increases in GCV) {

Calculate partial residual $r \leftarrow y - \beta_0 - \sum_{m=1}^{j-1} \beta_m f_m$.

Initialize Direction coefficients $\{\alpha_{kj}\}$.

Update Term j with response r.

Backfit Terms 1 to j.

}

PRUNE TERMS

For (j from max term added to one less than first local min in GCV) {

Save current model.

Drop Term which yields smallest GCV upon removal.

Backfit Terms 1 to (j-1).

}

CLEAN UP

Select the model with smallest GCV as the best model.

Adjust f_j's to have mean zero, changing β_0 accordingly.

3.2 Basic PPR Updates

At the single-term level, the approach of a PPR algorithm is to alternate between adjusting the function f_j with the linear combination coefficients $\{\alpha_{kj}\}_{k=1}^{d}$ fixed, and adjusting the linear combination coefficients with the function fixed. Detailed derivations of the PPR updates are presented by Friedman (1984a) and Hwang et al. (1993). A brief description of the main updates is given here.

Consider a single term PPR model, so that $\hat{y} = f(\boldsymbol{\alpha}^T \boldsymbol{x})$. Our goal is to find the f and α_k's which minimize RSS. With the α_k's fixed, the problem is merely a smoothing problem, where we smooth the response y on the predictor $\boldsymbol{\alpha}^T \boldsymbol{x}$. When we introduce β's to get a model of the form $\hat{y} = \beta_0 + \beta_1 f(\boldsymbol{\alpha}^T \boldsymbol{x})$, we obtain f by smoothing $(y - \beta_0)/\beta_1$ on $\boldsymbol{\alpha}^T \boldsymbol{x}$.

The update to adjust the α_k's given f is slightly more complicated, as the parameters no longer enter into the RSS quadratically. Let α_k^0 denote the current value of α_k and α_k^1 the new value. The Gauss-Newton update for the change $\delta_k = \alpha_k^1 - \alpha_k^0$ is obtained by regressing $y - \hat{y}$ on $\{x_k f'\}_{k=1}^{d}$, where $\hat{y}$ is the current fit and f' is the estimated derivative at the current value of f. The regression coefficient for $x_k f'$ is the value of δ_k. When we add β's into the model, the value of β_1 affects the derivative calculation, and the predictors become $\{x_k \beta_1 f'\}_{k=1}^{d}$.

Note that in order to get the α_k updates, we need to have derivative estimates at the current values of the linear combinations. In the supersmoother-based application, a modified version of first-differences is used to estimate the derivatives. As cubic smoothing splines are produced with piecewise-cubic basis functions, it is easy to obtain derivatives based on the fitted coefficients and the derivatives of the basis functions. With cubic smoothing splines, the first derivatives are in fact continuous. The S function `predict.smooth.spline()` will return derivative estimates of the desired order. As we can use exact derivative values rather than estimates, we expect the weight updates using smoothing splines to be more stable than those obtained with the supersmoother.

3.3 Initializing Direction Coefficients

Due to the highly nonlinear nature of the problem, the particular initial direction coefficients $\{\alpha_{kj}\}$ used can have a dramatic effect on the results of the algorithm. Hence it is useful to run the algorithm at a variety of starting values in order to explore a number of fits. The user may specify a matrix of starting directions. If no such direction matrix is provided, each new set of direction coefficients is obtained by regressing the current residual r on the x_k's. Anecdotally, the GCV criterion appears to undersmooth more often using regression-based starting directions. It is highly recommended that the user try a number of random initial directions in addition to the regression-based initial directions used by default.

3.4 Update Term

The Update Term routine is the primary fitting component in the algorithm. It fits a single term PPR model to the partial residual r using the updates described in Section 3.2. If an initial f_j has not been fit, the first step is to fit a smooth function to r with predictor $\boldsymbol{\alpha}_j^T \boldsymbol{x}$. The routine then alternates between changing $\boldsymbol{\alpha}_j$ and changing f_j, until the percent change in error is less than a specified tolerance ξ. (The default value for ξ is 0.005.)

Note that the $\boldsymbol{\alpha}$ update is guaranteed to be in a descent direction, but may be too large a change. One approach is to perform a line search for the appropriate step size. However, for each new value of $\boldsymbol{\alpha}$ we have new

predictors $\boldsymbol{\alpha}^T\boldsymbol{x}$, and hence must reevaluate the function f_j in order to calculate the error. Rather than putting this effort into a line search, we put effort into a backfitting routine. If the step $\boldsymbol{\delta}$ results in an increase in error after refitting f_j, we discard the step and stop updating this term. Due to the backfitting, we will have the opportunity to update the term later with a different response.

Some bookkeeping details are that after getting the new $\boldsymbol{\alpha}_j$, we normalize $\boldsymbol{\alpha}_j$ to norm one before recalculating $\boldsymbol{\alpha}_j^T\boldsymbol{x}$. After fitting f_j, we adjust the fitted values to have norm one, changing the β_j scale coefficient to keep the overall fit the same. At this point we also recalculate our derivative estimates f_j'.

3.5 Backfit Terms

The Backfit Terms algorithm uses a backfitting procedure to adjust the j terms previously fitted. As in Update Term, the procedure is repeated until the percent change in error is less than the tolerance ξ. The first step in backfitting is to recalculate the β's by regressing y on $\{f_m\}_{m=1}^{j}$. The second step is to loop over the j terms, where for the m^{th} term we call Update Term with response r the partial residual $r = y - \beta_0 - \sum_{j \neq m} \beta_j f_j$. Note that the first step cannot increase the error, as the current β's are a valid choice for the fitted parameters, and the second step cannot increase the error, as the Update Term routine does not change a term if doing so would increase the error.

3.6 Drop Term

The standard measure of the importance of a term used by Friedman (1984a) and Hwang et al. (1993) is the magnitude of its coefficient β_j. Note that this does not take into account any correlation structure between the directions. It is conceivable that two terms running in roughly the same direction could have large β_j's, yet a single one of the two terms might be sufficient to capture most of the activity in that direction. With m terms, we fix the values of the f_j's, and perform m linear regressions, each excluding one of the terms. Using the values of RSS obtained, we then calculate GCV values with df's reflecting the terms retained in each of the m models. The term whose exclusion leaves the model with the smallest GCV is then dropped.

3.7 The Ridge Option

In the linear regression steps in the algorithm, we provide the option of using ridge regression rather than regular linear regression. As some of the function values f_j may be highly correlated, this may add some stability in the calculation of the β_k's. In the α update, we assume δ is small when relying on the Gauss-Newton stepping, and ridge regression encourages small values of δ. The use of ridge regression in this context is still very much at the experimental stage.

4 Simulation Examples

In the following examples we compare the ASP algorithm to the SMART algorithm (Friedman, 1984a) which is the standard PPR algorithm. We first compare the algorithms on a number of bivariate predictor examples used in previous PPR papers, and then present some higher dimensional examples from the PPR and MARS literature.

We use simulated examples so that we may compare the fits on a sizable test set. Our metric for comparison is the fraction of variance unexplained (FVU), which is defined as

$$\text{FVU} = \frac{E\{\hat{g}(\mathbf{x}_i) - g(\mathbf{x}_i)\}^2}{E\{g(\mathbf{x}_i) - \bar{g}\}^2}$$

where $g(\mathbf{x}_i)$ is the true value of the function and $\hat{g}(\mathbf{x}_i)$ is the fitted value. We evaluate the FVU for a fit by replacing the expectations with averages over a set of 10,000 test set values. For the bivariate predictor examples, this set is a 100 by 100 grid on the unit square. In the higher dimensional examples, the test set is composed of random uniform values over the domain of the predictors. Note that the FVU is the mean squared error for the estimate $\hat{g}(\mathbf{x})$ scaled by the variance of the true function $g(\mathbf{x})$. When reporting FVU for the training set, we take the FVU over the training sample, treating the observed y values as $g(\mathbf{x})$.

We fit the SMART models using the `ppreg()` function in S-Plus. To select the number of terms in the model, we first run the algorithm with $M = 1$ and $M_{max} = 10$. Then we plot the FVU on the training set and pick the model size for which additional terms appear to add little extra

information. This is the procedure recommended by Friedman (1984a). To avoid giving SMART a disadvantage due to improper selection of model size, we fit the two models most likely to be optimal based on this criterion, and compare both of these with the ASP fit. The actual FVU values used in model selection are presented as FVU train.

A standard disclaimer is that a few runs of the algorithms on a small set of functions should not be overinterpreted. The main aim of this section is to demonstrate that ASP is at least competitive with SMART in many situations, and for some problems ASP appears to yield superior performance.

4.1 Bivariate Predictors

The first set of functions is used by Friedman et al. (1983) to demonstrate the Multidimensional Adaptive Spline Approximation algorithm. The functions are drawn from a paper by Franke (1979) in which a number of interpolatory surface approximation schemes are compared.

The function in Franke's first example is

$$\begin{aligned} g(x_1, x_2) &= 0.75 \exp\left\{-\frac{(9x_1-2)^2+(9x_2-2)^2}{4}\right\} + 0.75 \exp\left\{-\frac{(9x_1+1)^2}{49} - \frac{9x_2+1}{10}\right\} \\ &\quad +0.5 \exp\left\{-\frac{(9x_1-7)^2+(9x_2-3)^2}{4}\right\} + 0.2 \exp\{-(9x_1-4)^2-(9x_2-7)^2\}. \end{aligned}$$

The function in Franke's second example is

$$g(x_1, x_2) = \frac{1}{9}\{\tanh(9x_2 - 9x_1) + 1\}.$$

The training samples for the two functions each consisted of 100 pairs of predictors drawn uniformly from the unit square. For the noisy examples, error was added such that the ratio of the standard error of the signal to the standard error of the noise was roughly 4. For the first function $y_i = g(\mathbf{x}_i) + 0.065\varepsilon_i$, while for the second function $y_i = g(\mathbf{x}_i) + 0.025\varepsilon_i$, where the ε's are i.i.d. $N(0,1)$.

Table 1 presents the number of terms M, the FVU on the training set, and the FVU on the test set for each example. On both of Franke's interpolation examples (noiseless data), ASP does orders of magnitude better than SMART. On the noisy data, the results are comparable. In both cases

		Noiseless Data			Noisy Data		
Function	Method	M	FVU train	FVU test	M	FVU train	FVU test
Franke 1	ASP	4	0.00271	0.00839	3	0.04365	0.07610
	SMART	3	0.02526	0.06582	4	0.03685	0.07741
	SMART	4	0.00516	0.02093	5	0.03106	0.08228
Franke 2	ASP	1	5.3626e-12	1.7399e-09	1	0.07334	0.01067
	SMART	1	9.8904e-05	0.00015	1	0.07277	0.01059
	SMART	2	9.2477e-06	1.7495e-05	2	0.07269	0.01386

Table 1: FVU for Franke Examples

ASP correctly chose a single term for the second example. (The reason the training error is lower than the test error in the noiseless case is that the procedure favors those points in predictor space where data occurred, and the predictor values differ in the training and test sets.)

The second set of functions is used by Hwang et al. (1993) to compare their Hermite polynomial PPR algorithm with the SMART algorithm. They use 225 pairs of predictors drawn uniformly from the unit square. In the noiseless examples, the response is simply the value of the function $g(x_{1i}, x_{2i})$, while in the noisy examples the response is $g(x_{1i}, x_{2i}) + 0.25\varepsilon_i$, where the errors ε_i are i.i.d. $N(0,1)$. The functions are

- *Simple Interaction Function*

$$g^{(1)}(x_1, x_2) = 10.391\{(x_1 - 0.4) \cdot (x_2 - 0.6) + 0.36\}$$

- *Radial Function*

$$g^{(2)}(x_1, x_2) = 24.234\{r^2(0.75 - r^2)\}, \qquad r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$$

- *Harmonic Function*

$$g^{(3)}(x_1, x_2) = 42.659\{(2 + x_1)/20 + Re(z^5)\}$$

where $z = x_1 + ix_2 - 0.5(1+i)$, or equivalently, with $\tilde{x}_1 = x_1 - 0.5$, $\tilde{x}_2 = x_2 - 0.5$,

$$g^{(3)}(x_1, x_2) = 42.659\{0.1 + \tilde{x}_1(0.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\tilde{x}_2^2 + 5\tilde{x}_2^4)\}$$

- *Additive Function*

$$\begin{aligned} g^{(4)}(x_1, x_2) &= 1.3356[1.5(1 - x_1) + e^{2x_1 - 1}\sin\{3\pi(x_1 - 0.6)^2\} \\ &\quad + e^{3(x_2 - 0.5)}\sin\{4\pi(x_2 - 0.9)^2\}] \end{aligned}$$

- *Complicated Interaction Function*

$$g^{(5)}(x_1, x_2) = 1.9[1.35 + e^{x_1}\sin\{13(x_1 - 0.6)^2\}e^{-x_2}\sin(7x_2)]$$

These functions are scaled and translated to have a standard deviation of 1 and a nonnegative range.

Table 2 presents the ASP and SMART results on these examples. On the noiseless data, ASP again does orders of magnitude better on most of the functions. This suggests that SMART is too conservative in noiseless situations. If ASP did well on the noiseless data but poorly on the noisy data, we could attribute this discrepancy to overfitting by ASP. But this is not the case. ASP is also competitive with SMART in the noisy case, and in 4 of the 5 Hwang examples actually does slightly better. However, it would be imprudent to draw stronger conclusions from these few examples. The trends presented here are consistent with those observed on other runs of these examples which are not reported.

4.2 High Dimensional Predictors

The first example is taken from Friedman et al. (1983). The basic function is

$$g(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5.$$

As was done by Friedman et al., we generate 200 random uniform predictors from the six-dimensional unit hypercube, and take the response to be $g(\mathbf{x}_i) +$

		Noiseless Data			Noisy Data		
Function	Method	M	FVU train	FVU test	M	FVU train	FVU test
Simp Int	ASP	2	8.1e-11	5.2e-07	2	0.06558	0.00769
	SMART	2	0.00010	0.00079	2	0.06414	0.01829
	SMART	3	0.00009	0.00088	3	0.06253	0.02348
Radial	ASP	4	0.00001	0.00247	3	0.05430	0.03699
	SMART	2	0.02497	0.03231	3	0.06545	0.02345
	SMART	3	0.01164	0.01970	4	0.04758	0.01617
Harmonic	ASP	6	0.00400	0.05274	4	0.08781	0.09064
	SMART	6	0.04157	0.20551	4	0.11403	0.16105
	SMART	7	0.04215	0.19331	5	0.11439	0.16252
Additive	ASP	2	3.2e-10	2.5e-08	2	0.05034	0.00651
	SMART	3	0.00131	0.00196	2	0.04865	0.00860
	SMART	4	0.00162	0.00336	3	0.04706	0.00930
Comp Int	ASP	5	0.00427	0.01282	4	0.05160	0.03112
	SMART	4	0.03931	0.08328	5	0.05864	0.04903
	SMART	5	0.02068	0.05272	6	0.05000	0.05313

Table 2: FVU for Hwang Examples

ε_i where the ε_i's are i.i.d. $N(0,1)$. Note we include one spurious predictor which does not influence the response.

Friedman uses this function again in his MARS paper (1991). This time a ten-dimensional predictor is used, with 5 of the predictors being spurious. To make matters worse, the sample size is reduced to 100.

A final example is from Friedman's Fast MARS paper (1993). A hypothetical robot arm has two segments connected by a joint. One end of the arm is tethered at the origin. The location of the other end of the arm in three-space (x, y, z) may be described by three angles $(\theta_1, \theta_2, \phi)$ through the equations

$$x = \ell_1 \cos\theta_1 - \ell_2 \cos(\theta_1 + \theta_2)\cos\phi$$

$$y = \ell_1 sin\theta_1 - \ell_2 \sin(\theta_1 + \theta_2)\cos\phi$$

$$z = \ell_2 \sin\theta_2 \sin\phi$$

where ℓ_1 and ℓ_2 are the lengths of the two arm segments. The distance of the tip of the arm from the origin is then

$$d = (x^2 + y^2 + z^2)^{(1/2)}.$$

Our goal is to predict the distance d given $(\ell_1, \ell_2, \theta_1, \theta_2, \phi)$ using a PPR model. A training set of size 400 was generated uniformly over the complete range of the angles $\theta_1 \in [0, 2\pi]$, $\theta_2 \in [0, 2\pi]$, $\phi \in [-\pi/2, \pi/2]$, and arm lengths in the range $\ell_1 \in [0, 1]$ and $\ell_2 \in [0, 1]$.

Table 3 presents the results of the high dimensional examples. In the robot arm example, it is hard to select the correct model size for the SMART fit, as the training FVU plot has a number of plateaus. As the data is noiseless, adding extra terms will tend to improve the fit, but without knowledge that the data is noiseless we would probably tend to select a more parsimonious fit. ASP and SMART appear to do roughly the same on this example.

The two spurious predictor examples prove to be quite informative. When there is a single spurious predictor, both algorithms are able to handle the data quite well, with ASP again doing marginally better. With five spurious predictors, the amount of noise in the predictors overwhelms the amount of information. MARS adapts quite well to this problem, as it is fundamentally additive in the predictors. Ridge functions have a harder time as there is no explicit encouragement to ignore the spurious predictors. ASP appears to have been slightly successful at ignoring the spurious predictors, and selects

Function	Method	M	FVU train	FVU test
Spurious Preds, d=6	ASP	5	0.02924	0.02983
	SMART	5	0.03830	0.05186
	SMART	6	0.02620	0.03451
Spurious Preds, d=10	ASP	3	0.04556	0.15680
	SMART	1	0.29204	0.27907
	SMART	3	0.11861	0.60961
	SMART	6	0.01345	0.92303
Robot Arm	ASP	4	0.06874	0.13655
	SMART	4	0.09743	0.15481
	SMART	9	0.03097	0.10646

Table 3: FVU for High Dimensional Predictor Examples

a relatively small model. SMART has difficulty regardless of the number of terms used. Examining training FVU suggests that a 6 term model is appropriate, but this model performs quite poorly on the test data.

5 Conclusions

The examples above demonstrate that ASP is at least competitive with SMART on noisy data, and appears to be superior in noiseless situations. The GCV criterion coupled with a check for overfitting seems to effectively determine smoothing parameters and model size.

This paper suggests a number of algorithmic elements which may be of use in PPR modelling. Hopefully these ideas will be of use to other designers of PPR methods, even if this particular smoothing spline implementation is not.

6 Directions for Future Research

The fully automatic nature of the supersmoother was of interest in the simple univariate smoothing case, but its true value was as a building block for

more involved methods in which it was desired to have a fast algorithm free of tuning parameters. Similarly, the ASP algorithm performs well in implementing a single response PPR, but has the additional value of being a component for use in more complicated models. In particular, the authors intend to use it as a component in fitting Generalized Projection Pursuit models as described in Roosen and Hastie (1993).

There are undoubtedly many avenues for improving this version of the ASP algorithm. In particular, the effect of using backfitting as opposed to step cutting is not fully understood. The ramifications of employing ridge regressions have not yet been determined. Also, the speed of the algorithm could undoubtedly be increased dramatically by implementing large portions of it in FORTRAN or C rather than in S. The current version of the ASP algorithm may be obtained from the first author (currently charles@playfair.stanford.edu), who is particularly interested in feedback regarding performance of the algorithm.

Acknowledgments

The authors thank Jerry Friedman for useful discussions regarding this material, the Statistics and Data Analysis Research Department of AT&T Bell Laboratories, and the Stanford University Department of Statistics, for support during the performance of this research.

References

[1] Chambers, J. M. and Hastie, T. J. (1992), Statistical Models in S, Pacific Grove, CA: Wadsworth & Brooks/Cole.

[2] Craven, P. and Wahba, G. (1979), "Smoothing Noisy Data With Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation," *Numerische Mathematik*, 31, 317-403.

[3] Franke, R. (1979), A Critical Comparison of Some Methods for Interpolation of Scattered Data. Naval Postgraduate School report NPS-53-79-003.

[4] Friedman, J. H. (1984a), SMART User's Guide. Dept. of Statistics Technical Report No. 1, Stanford University.

[5] Friedman, J. H. (1984b), A Variable Span Smoother. Dept. of Statistics Technical Report LCS 05, Stanford University.

[6] Friedman, J. H. (1991), "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, 19, 1-141.

[7] Friedman, J. H. (1993), Fast MARS. Dept. of Statistics Technical Report No. 110, Stanford University.

[8] Friedman, J. H., Grosse, E., and Stuetzle, W. (1983), "Multidimensional Additive Spline Approximation," *SIAM Journal of Scientific and Statistical Computing*, 291-301.

[9] Friedman, J. H. and Stuetzle, W. (1981), "Projection Pursuit Regression," *Journal of the American Statistical Association*, 76, 817-823.

[10] Hastie, T. J. (1989), Discussion of "Flexible Parsimonious Smoothing and Additive Modeling" by J. H. Friedman and B. W. Silverman, *Technometrics*, 31, 23-29.

[11] Hastie, T. J. and Tibshirani, R. J. (1990), Generalized Additive Models, London: Chapman and Hall.

[12] Hertz, J., Krogh, A., and Palmer, R. G. (1991), Introduction to the Theory of Neural Computation, New York: Addison-Wesley.

[13] Hwang, J-N, Lay, S-R, Maechler, M., Martin, D. and Schimert, J. (1993), "Regression Modeling in Back-Propagation and Projection Pursuit Learning," *IEEE Transactions on Neural Networks*. In press.

[14] Roosen, C. B. and Hastie, T. J. (1993), Logistic Response Projection Pursuit Regression. Statistics and Data Analysis Research Department, AT&T Bell Laboratories, Document No. BL011214-930806-09TM.

[15] Silverman, B. (1985), "Some Aspects of the Spline Smoothing Approach to Non-parametric Regression Curve Fitting," *Journal of the Royal Statistical Society*, Ser. B, 47, 1-52.