

Contents

| | | |
|----------|--|-----------|
| 1 | Projection Pursuit Regression and Neural Networks | 6 |
| | <i>S. Klinké, and J. Grassmann</i> | |
| 1.1 | Introduction | 6 |
| 1.2 | Projection Pursuit Regression | 7 |
| 1.3 | Neural Networks | 27 |
| 1.4 | Optimization Methods | 34 |
| | References | 39 |
| | Appendix | 44 |
| A.1 | Test functions for Projection Pursuit Regression | 44 |
| | Index | 46 |

Contributors

S. KLINKE, Humboldt-University of Berlin, Department of Economics, Institute of Statistics and Econometrics, Spandauer Strasse 1, D-10178 Berlin, Germany

J. GRASSMANN, Stanford University, Department of Statistics, Margret Jacks Hall, California 94305, USA.

1 Projection Pursuit Regression and Neural Networks

S. KLINKE[‡], and J. GRASSMANN[†]

[‡] Humboldt-University of Berlin, Department of Economics, Institute of Statistics and Econometrics, Spandauer Strasse 1, D-10178 Berlin, Germany

e-mail: sigbert@wiwi.hu-berlin.de

WWW: <http://www.wiwi.hu-berlin.de/~sigbert>

[†] Stanford University, Department of Statistics, Margret Jacks Hall, California 94305, USA

e-mail: jg@stat.stanford.edu

1.1 INTRODUCTION

In statistics several estimation methods are available to model a response function

$$E(Y|X = x) = m(x_1, \dots, x_p) + \epsilon$$

with (X, Y) a pair of random variables with $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}$.

They range from parametric methods, e.g. linear regression, to nonparametric methods like k -nearest neighbour methods.

All methods, parametric and nonparametric have advantages and disadvantages. The parametric models fix the model structure completely. In terms of “interpretation” this is an advantage. In the case that the parametric model does not match the underlying true model we estimate the “best” approximation in the space of the chosen function class. However, it can be quite far away from the true model and the conclusion from the estimation may be wrong.

The first problem in nonparametric estimation arises in the estimation of model parameters, e.g. smoothing parameters. These parameters have to be estimated from the data, mostly in computer-intensive routines. The

next problem arises from the multivariate space itself. The so called “curse of dimensionality” describes the property of data to become sparse if the dimension of the space grows. For a fixed number n of observations and the volume $V(p) \approx d^p$ depending on the diameter d it follows that the average distance d_p between two datapoints is

$$\lim_{p \rightarrow \infty} d_p = \lim_{p \rightarrow \infty} 1/\sqrt[p]{n} = 1.$$

How severe this problem is can be seen for estimation in Silverman (1986, Table 4.2 on page 94) and for visualization in Asimov (1985, Table 1).

The most severe problem is that the interpretation of the fit of a non-parametric model is very difficult. The local averaging property provides no general structure of the data. A possible approach is to introduce structure in the nonparametric models and to construct semiparametric models. Two methods will be examined in detail: Projection Pursuit Regression (PPR) and Feedforward Neural Networks (FFN).

The technique of PPR is rather nonparametric than the FFN, because non-parametric estimators are used in the estimation process. Once the number of hidden neurons in a FFN is fixed the model is parametric. The choice of the number of the hidden neurons makes a FFN regression a nonparametric technique.

The idea of “Projection Pursuit” has been introduced by Kruskal (1969; 1972) for exploratory data analysis. The term “projection” implies that we are looking at projected data and the term “pursuit” means to find a “good” projection for the purpose of regression.

The approach has been successfully implemented for exploratory purposes by many authors (Friedman and Tukey, 1974; Jee, 1985; Huber, 1985; Jones and Sibson, 1987; Friedman, 1987; Hall, 1989a; Cook and Cabrera, 1992; Cook, Buja and Cabrera, 1993; Posse, 1995; Nason, 1995). The idea has been applied to location and symmetry estimation (Blough, 1989; Maller, 1989), density estimation (Friedman, Stuetzle and Schroeder, 1984; Chen, 1987; Rejtó and Walter, 1990; Rejtó and Walter, 1992), classification (Friedman and Stuetzle, 1981a; Portier, Dippon and Hetrick, 1987; Flick, Jones, Priest and Herman, 1990) and discriminant analysis (Posse, 1992; Ahn and Rhee, 1992; Polzehl, 1995). Good references about projection pursuit are Jones and Sibson (1987) and Huber (1985).

In the area of neural networks we have an enormous amount of literature. Good references for this topic are Bishop (1995) and Ripley (1996).

1.2 PROJECTION PURSUIT REGRESSION

1.2.1 The basic algorithm

8 PROJECTION PURSUIT REGRESSION AND NEURAL NETWORKS

A pure nonparametric approach can lead to a strong oversmoothing, since the sparseness of the space requires to include a lot of space and observations to do a local averaging for a reliable estimate. To estimate the response function $m(x)$ from the data $\{(x_{1,1}, \dots, x_{1,p}, y_1), \dots, (x_{n,1}, \dots, x_{n,p}, y_n)\}$, Friedman and Stuetzle (1981b) have suggested the following algorithm:

1. Set $r_i^{[0]} = y_i$.
2. For $j = 1, \dots$ maximize

$$R_{[j]}^2 = 1 - \frac{\sum_{i=1}^n \left(r_i^{[j-1]} - \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i) \right)^2}{\sum_{i=1}^n \left(r_i^{[j-1]} \right)^2}$$

by varying over the parameters $\hat{\alpha}_{[j]} \in \mathbb{R}^p$ ($\|\alpha_{[j]}\| = 1$) and a univariate regression function $\hat{m}_{[j]}$.

3. Define $r_i^{[j]} = r_i^{[j-1]} - \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i)$ and repeat step 2 until $R_{[j]}$ becomes small. A small $R_{[j]}$ implies that $\hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i)$ is approximately the zero function and we will not find any other useful direction.

This algorithm leads to an estimation of the response function by

$$\hat{m}_M(x) = \sum_{j=1}^M \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x). \quad (1.1)$$

The advantages of estimating the response function are:

- We use univariate regression functions instead of their multivariate analogues and avoid the “curse of dimensionality”.
- Univariate regressions are easily and quick to calculate.
- In contrast to generalized additive models (GAM) PPR is able to approximate a much richer class of functions.
- In comparison to local averaging methods, e.g. k -nn estimator, we are able to ignore variables of no or small information about m .

Of course we also have some disadvantages with this model:

- We have to examine a p -dimensional parameter space to estimate $\hat{\alpha}_{[j]}$.
- We have to solve the problem of selecting a smoothing parameter, if we use nonparametric smoothers for $\hat{m}_{[j]}$.

- The interpretation of a single term may not be easy.

For simplicity let us assume that

$$E(X_i) = 0, Var(X_i) = 1, E(Y) = 0, Var(Y) = 1.$$

Friedman and Stuetzle (1981b) constructed a special smoother for estimating the unknown regression function \hat{m}_M , similar to the supersmoothen (Friedman, 1984b).

Moreover, they suggested to use backfitting to improve the quality of the estimate. Particularly, it holds that

$$E \left(Y - \sum_{\substack{j=1 \\ j \neq k}}^M m_{[j]}(\alpha_{[j]}^T X) \middle| \alpha_{[k]}^T X \right) = m_{[k]}(\alpha_{[k]}^T X).$$

Cycle now through $k = 1, \dots, M, 1, \dots, M, \dots$ and update either $\alpha_{[k]}$ or $m_{[k]}$ or both of them by

$$\hat{m}_{[k]}(\hat{\alpha}_{[k]}^T x_i) = y_i - \sum_{\substack{j=1 \\ j \neq k}}^M \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i).$$

1.2.2 How good is the approximation ?

We have described how to do PPR. But which kind of function can be picked up by PPR easily and which not ?

Donoho and Johnstone (1989) examined the approximation of bivariate polynomials by a PPR (in comparison to kernel based methods). For their examinations they chose

$$f(x, y) = \Re(z^8) = x^8 - 28x^6y^2 + 70x^4y^4 - 28x^2y^6 + y^8$$

evaluated on $[-1, 1] \times [-1, 1]$.

Figure 1.1 shows the general problem of estimating a multivariate function: the corners contain the structure. We need either a lot of datapoints within the whole area or we need datapoints at the "right" place. In practice either situation is difficult to get.

Moreover Donoho and Johnstone (1989) showed that radial functions (e.g. $f(x, y) = g(\sqrt{x^2 + y^2})$) can be approximated better by PPR than functions where the oscillation is averaged out locally.

To overcome the last problem we may use a local PPR algorithm. The idea is to find a point $\hat{\mu}_{[j]}$ and use a neighbourhood of it to estimate a local regression function

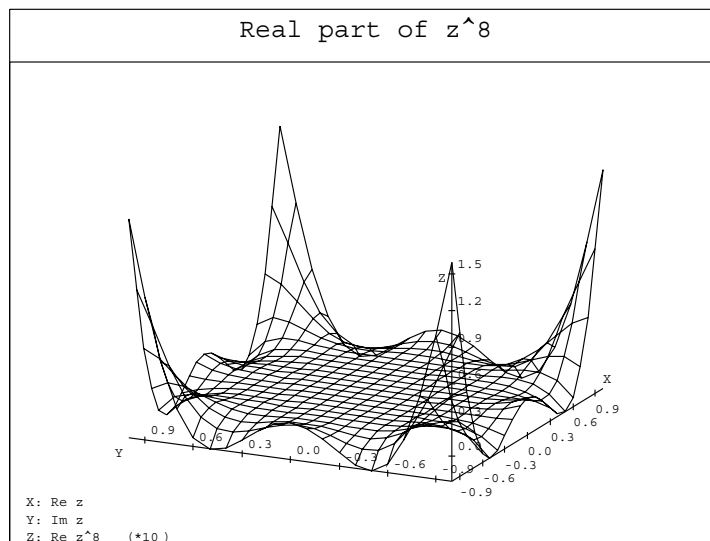


Fig. 1.1 Real part of the complex polynomial z^8 plotted in $[-1, 1] \times [-1, 1]$.

$$\hat{m}_{[j]}(x) = \hat{m}_{[j]}(\hat{\mu}_{[j]} + \hat{\alpha}_{[j]}^T x).$$

For a unique location estimate we may choose $|\hat{\mu}_{[j]}|$ to be minimal.

Since $\hat{\mu}_{[j]} + \hat{\alpha}_{[j]}^T x$ defines a line in \mathbb{R}^p , we can compute the distances from the line for each observation. The distance can be used to define a weight of an observation for the regression.

This approach has some serious disadvantages: the interpretation of the location parameters $\hat{\mu}_{[j]}$ and the projection vectors $\hat{\alpha}_{[j]}$ will be difficult. The optimization involves a search within a much larger space than \mathbb{R}^{p-1} .

Another way to handle the problem might be a different method of decomposition of the space orthogonal to $\hat{\alpha}_{[j]}$. Let us assume that the model for the data in Donoho and Johnstone (1989) is

$$m(x_1, x_2) = \sum_{i=1}^M G_{[j]}(f_{[j,1]}(x_1) + f_{[j,2]}(x_2))$$

with $G_{[j]}$ a nonparametric estimate and $f_{[j,\bullet]}$ a parametric transformation of the x -coordinates (e.g. polynomials of fixed degree). If G is a known link function then we have a generalized additive model (GAM). As in generalized partial linear models (GPLM) we choose another metric instead of the euclidean.

1.2.3 How many terms to choose ?

The last section shows that it depends strongly on the underlying regression function how many terms we have to choose. In the case of $\Re(z^8)$ we need a lot of projections (at least eight) to model the regression function appropriately.

A method similar to the choice of the number of components in principal component analysis can be used in PPR too. Computing $\beta_{[j]} = \sqrt{\text{var}(r_{[j]})}$ we get

$$\text{var}(r_{[j]}/\beta_{[j]}) = \text{var}(r_{[j]})/\beta_{[j]}^2 = 1.$$

We can construct a plot of $(j, \beta_{[j]})$ and stop if the decrease of the $\beta_{[j]}$ becomes too small.

Other criteria, e.g. standard model selection criteria like generalized cross validation (GCV), have been used in Hwang, Lay, Maechler, Martin and Schimert (1994) and Roosen and Hastie (1994). Friedman (1984a) has remarked that adding more terms than necessary and using a backward stepwise model selection helps to avoid getting stuck in local maxima. In the spline setting of Roosen and Hastie (1994) the authors added terms until two consecutive increases in the GCV have been noticed. Then, they pruned back until the number of terms is one less than the number of terms in the model with the first minimum of the GCV.

1.2.4 Interpretable Projection Pursuit Regression

Morton (1989) tried to find a more general approach to increase the interpretability of a PPR-model. The first question is: What are interpretable models ?

A simple answer is to choose $\alpha_j = e_j$, which allows a quite easy interpretation. But then we are fitting an additive model to the data. Assume that the number of terms M is fixed. Then we have projection vectors $\hat{\alpha}_{[1]}, \dots, \hat{\alpha}_{[M]}$. A goal could be to get as many zeros as possible in each vector $\hat{\alpha}_{[j]} = (\hat{\alpha}_{[j]1}, \dots, \hat{\alpha}_{[j]p})$ such that only some variables contribute to the current fit.

The vector

$$\eta_i = \sum_{j=1}^M \hat{\alpha}_{[j]i}^2$$

contains the sum of each term's relative contribution to the i -th variable ($i = 1, \dots, p$). Now, we can define an index

$$S_g(\eta) = \frac{p}{p-1} \sum_{i=1}^p \left(\frac{\eta_i}{M} - \frac{1}{p} \right)^2$$

which measures the individual interpretability of each combination $\hat{\alpha}_{[1]}, \dots, \hat{\alpha}_{[M]}$.

A large value of the index S_j means that the different variables contribute to different projection vectors.

The second question: Can we use this index to choose M in practice? Morton gave the following example:

$$\hat{m}_M(x_1, x_2) = \begin{cases} \hat{m}_{[1]} \left(\frac{x_1+x_2}{\sqrt{2}} \right) & (M = 1) \\ \hat{m}_{[1]}(x_1) + \hat{m}_{[2]}(x_2) & (M = 2). \end{cases}$$

Which model should be chosen? She constructed three different indices, but none of them satisfied her completely. A solution would be to return a sequence of models and let the user with special knowledge about the underlying topic decide which model suits best.

Some practical experience shows that the interpretability drops down, when we improve the approximation. A possibility might be to use the index as a penalization during the search for $\hat{\alpha}_{[j]}$.

1.2.5 Convergence rates

Hall (1988) stated that the convergence rate, the bias and the variance are affected by two issues: the estimation of α_j and the estimation of m_j . He showed that the estimation of α_j can be done with the order $n^{-1/2}$ whereas the nonparametric estimation of m_j has an order worse than $n^{-1/2}$. Since the standard algorithm can not achieve an order $n^{-1/2}$ he proposes a two stage estimator:

1. Use an undersmoothed regression estimator to find $\hat{\alpha}_{[j]}$ and
2. Apply the obtained $\hat{\alpha}_{[j]}$ in the regression smoother with a correct amount of smoothing.

Hall (1988) considered the case of projection pursuit density estimation whereas in Hall (1989b) he worked on the PPR case. He proved that

$$\sqrt{nh}(\hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x) - m_j(\alpha_j^T x)) = \tau(x)N(0, 1) + \sqrt{nh}h^r(\beta_1 + \beta_2 + \beta_3) + o_p(1)$$

with $r > 1$ the number of smooth derivatives of $m_{[j]}$. Thus, the bias is determined by β_1 , β_2 and β_3 . The parameters β_1 and β_2 are the contribution of the bias by maximizing the direction, whereas β_3 results from the the bias of the nonparametric estimator, e.g. the Nadaraya-Watson estimator. If we choose the bandwidth (window) proportional to $n^{-1/3}$ to $n^{-1/4}$ then we achieve a parametric rate and get

$$\sqrt{nh}(\hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x) - m_{[j]}(\alpha_{[j]}^T x)) = \tau(x)N(0, 1) + \sqrt{nh}h^2\beta_3 + o_p(1)$$

under the assumption that the smoothing parameter $h \sim n^{-1/5}$.

Of course, an undersmoothed estimate causes a wiggly error function with a lot of local minima.

Huber (1985) has established a weak L_2 -convergence for PPR. Jones proved the norm convergence and Jones (1992) established a $O(1/\sqrt{M})$ non-sampling convergence rate. He used a modification of the PPR algorithm which he called relaxed PPR. Here we choose

$$\hat{m}_{M+1}(x) = \lambda_{[M]} \left\{ \hat{m}_{M+1}(\hat{\alpha}_{[M+1]}^T x) - \hat{m}_M(x) \right\}$$

such that $0 < \lambda_{[M]} < 1$ and $\hat{\alpha}_{[M+1]}$ minimizes the norm of

$$|(1 - \lambda)\hat{m}_M(x) + \lambda\hat{m}_M(\hat{\alpha}_M^T x) - m(x)|.$$

He gave some examples to estimate the error

$$e_M = |\hat{m}_M(x) - m(x)|,$$

e.g. by

$$e_M^2 \leq \frac{\bar{M}^2}{M + \bar{M}^2/e_0^2}.$$

e_0 and \bar{M} can be estimated from the data if the d -dimensional Fouriertransform of $m(x)$ exists and has a finite $L_1(\mathbb{R}^p)$ norm. Obviously, the dimension of the search space will be increased by one to p . This applies also to feed-forward networks with one hidden layer and sigmoidal or squashed activation functions.

Rejtő and Walter (1992) generalized PPR for a Hilbert space. Assuming that in each step the regression function is chosen as

$$|m_{[j]}(\alpha_{[j]}^T x)| \geq \rho \sup_{\alpha} |m_{[j]}(\alpha^T x)|$$

with $0 < \rho < 1$ they showed strong convergence for this algorithm. Even though they claimed that a search will be finished earlier, if we are close to the global maximum, an implementation seems rather difficult to us.

1.2.6 Modifications

1.2.6.1 Other smoothing methods. Generally, other smoothing methods are available. Linear ($k = 1$) and polynomial regression (of order $k > 1$) would lead to an approximation of the unknown regression surface by polynomials of order k

$$\hat{m}_M(x) = \sum_{j_1, \dots, j_p} c_{j_1 \dots j_p} x_1^{j_1} \dots x_p^{j_p}$$

used by Hwang et al. (1994). In the study of Roosen and Hastie (1994) three different smoothers (smoothing spline, supersmoother and polynomial)

are compared for known functions m in terms of the fraction of variance unexplained (FVU)

$$FVU_M = \frac{\sum_{i=1}^n (\hat{m}_M(x_i) - m(x_i))^2}{\sum_{i=1}^n (m(x_i) - \bar{m}(x_i))^2}$$

with $\bar{m}(x_i) = 1/n \sum_{i=1}^n m(x_i)$. They used the same bivariate functions as in Hwang et al. (1994): a simple interaction function, a radial function, a harmonic function, an additive function and a complicated interaction function (see appendix A.1). With noiseless data the polynomials perform best for the simple interaction function, for the harmonic function and for the complicated interaction, whereas the smoothing splines perform better for the complicated interaction function and slightly better on the radial function. On noisy data the smoothing splines perform better on all functions except of the harmonic.

Roosen and Hastie (1994) used a modified GCV criterion to determine the amount of smoothing in their spline approach. Let

$$r_{ij} = y_i - \sum_{\substack{k=1 \\ k \neq j}}^M \hat{m}_{[k]}(\hat{\alpha}_{[k]}^T x_i)$$

be the partial residual and

$$d_j(\lambda) = Mp + \sum_{j=1}^M \text{trace}(S_{[j]}(\lambda))$$

with λ the smoothing parameter and $S_{[j]}(\lambda)$ the smoother matrix. Define

$$GCV_j(\lambda) = \frac{\sum_{i=1}^n (r_{ij} - \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i))^2 / n}{(1 - d_j(\lambda)/n)^2}$$

and find the “optimal” directions and smoothing parameters by cycling through $j = 1, \dots, M, 1, \dots, M, \dots$. Hastie and Tibshirani (1990) stated that the GCV tends to undersmooth. Because we fit residuals in each step of PPR the autocorrelation may be interpreted as structure and may lead to catastrophic overfitting. They counted the number of local maxima. If the number was above some threshold (here: 10), then $d(\lambda)$ is replaced by $4 + Mp$ to remove the structure that confuses the GCV criterion.

1.2.6.2 Optimization criteria. As a modification one can apply another optimization criterion. In fact, $R_{[j]}^2$ can be viewed as the well known R^2 criterion in linear regression. From linear regression we know other criteria, e.g. the minimization of the sum of the absolute deviation

$$SAD_{[j]} = \sum_{i=1}^n |r^{[j-1]} - \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i)|$$

or the median absolute deviation

$$MAD_j = \text{median}(|r^{[j-1]} - \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i)|)$$

for robust estimation. Obviously, all these criteria can be reformulated for weighted observations.

For a large number of variables, e.g. in speech or image recognition which typically consists of several thousands, Intrator (1990) suggested to add not only a smoothness criterion, but also a projection pursuit index from exploratory projection pursuit to the error function. So, he can decouple the search for an interesting projection from the smoothing part.

1.2.6.3 PPR for moderate non-linearities. In practice we often have the situation that the data do not have strong non-linearities. Thus, Aldrin, Bølviken and Schweder (1993) suggested to use the ordinary least squares estimator as an estimator for the direction and to estimate the function non-parametrically. Under the condition that the regressors have an elliptically contoured distribution and the function is approximately monotonous it follows that

$$\alpha_{[j],OLS} = E(\alpha_{[j]}^T X m_{[j]}(\alpha_{[j]}^T X)) \alpha_{[j]}. \quad (1.2)$$

A more complicated method can be used if the expectation in 1.2 is near zero, e.g. if m_j is a symmetric quadratic function. It holds that

$$\begin{aligned} \alpha_{[j],OLS} &= \lambda \alpha_{[j]} + B \\ \lambda &= E(\alpha_{[j]}^T X m_{[j]}(\alpha_{[j]}^T X)) \\ B &= \Sigma_x^{-1} E(m_{[j]}(\alpha_{[j]}^T X) (X - (\Sigma_x^{-1} \alpha_{[j]} \alpha_{[j]}^T X))) \end{aligned}$$

with Σ_x the covariance matrix of the regressors. A sample version is obtained by

$$\begin{aligned} \hat{\alpha}_{[j],OLS} &= \hat{\lambda} \hat{\alpha}_{[j]} + \hat{B} + \hat{\Sigma}_x^{-1} \hat{\Sigma}_{x\epsilon} \\ \hat{\lambda} &= \frac{1}{n} \sum_{i=1}^n \hat{\alpha}_{[j]}^T X \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T X) \end{aligned}$$

$$\hat{B} = \hat{\Sigma}_x^{-1} \frac{1}{n} \sum_{i=1}^n \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i) (x_i - (\hat{\Sigma}_x^{-1} \hat{\alpha}_{[j]}) \hat{\alpha}_{[j]}^T x_i)$$

with $\hat{\Sigma}_x$ the sample covariance matrix and $\hat{\Sigma}_{x\epsilon}$ the sample cross covariance matrix of x and the residuals.

The iterative method to find a direction will be

1. Estimate $\hat{\alpha}_{[j,0]}$ via OLS and compute a smooth $\hat{m}_{[j,0]}$
2. Compute

$$\hat{\alpha}_{[j,k+1]} = \frac{\hat{\alpha}_{[j,k]} - \hat{B}}{\hat{\lambda}}$$

and standardize by $\hat{\alpha}_{[j,k+1]}^T \hat{\Sigma}_x \hat{\alpha}_{[j,k+1]} = 1$.

3. Smooth again and obtain a new $\hat{m}_{[j,k+1]}$.
4. Repeat the steps 2 and 3 until

$$D_k = \left| \frac{\hat{\alpha}_{[j,k]}}{\sqrt{\hat{\alpha}_{[j,k]}^T \hat{\Sigma}_x \hat{\alpha}_{[j,k]}}} - \frac{\hat{\alpha}_{[j,k+1]}}{\sqrt{\hat{\alpha}_{[j,k+1]}^T \hat{\Sigma}_x \hat{\alpha}_{[j,k+1]}}} \right|$$

becomes small.

The authors applied this technique to the Boston housing data (see section 1.2.8.1).

1.2.6.4 PPR and Sliced Inverse Regression. Li (1991) proposed a new technique for dimension reduction and regression analysis called “sliced inverse regression”.

This technique can be used to determine an interesting direction for PPR. Assume that the dependent variable Y will be sliced in $S = n^q$ ($0.5 \leq q < 1$) non-overlapping slices, so that we can define

$$h(y) = \begin{cases} c_k & c + k(d - c)/S \leq y < c + (k + 1)(d - c)/S \quad (k = 0, \dots, S - 1) \\ 0 & \text{otherwise} \end{cases}$$

with c and d constants with $c < \min(y_i)$ and $d > \max(y_i)$ and $c_k = \text{mean}_{y_i}$ in the slice $k(y_i)$. Choose a projection vector $\hat{\alpha}_{[j,0]}$ and minimize the c_k 's by

$$\frac{\sum_{i=1}^n (h_{[0]}(y_i) - \hat{\alpha}_{[j,0]}^T X_i)^2}{n \text{var}(h_{[0]}(y_i))} \quad (1.3)$$

with the constraint $\text{var}(h_{[j]}(y_i)) > v$ to avoid to fit a constant transformation h . The next step is a linear regression to compute a projection vector

$\hat{\alpha}_{[j,l]}$. Then we compute again a function $h_{[l]}$, a projection vector $\hat{\alpha}_{[j,l+1]}$ and so on.

The minimization process over c_k is equivalent to solve a set of S quadratic equations in c_0, \dots, c_{S-1} which can be solved by iterative algorithms.

Note that this method has the same problems as sliced inverse regression. For example, it will not provide a good $\hat{\alpha}_{[j]}$ in case that the data are clustered.

In fact, (1.3) is an approximation to $1 - R^2$ and the minimization is equivalent to find the best linear predictor and a non-linear transformation function h . Zhu and Li (1991) showed that this method leads to a consistent estimate for $1 - R^2$.

In case of two slices ($S = 2$), this method may provide a quick way to find a reasonable starting vector $\hat{\alpha}_{[j]}$ rather than starting with the linear regression projection.

1.2.6.5 PP-type regression model. Although PPR is designed to overcome the problem of the ‘‘curse of dimensionality’’, Donoho and Johnstone (1989) showed that the bias of (1.1) on page 8 depends on the dimension of X , if the regression function is harmonic. Chen (1991) proposed a modification of the PPR model which is not as flexible as PPR, but the rate of convergence is not affected by the dimensionality of X .

For a given $M < \dim(X)$ and a given set of projection vectors $\alpha_1, \dots, \alpha_M$ we can solve the (constrained) minimization problem

$$\min \sum_{i=1}^n (y_i - \hat{m}(x_i))^2 1_U(x_i) \tag{1.4}$$

with U an open set which contains the unit ball C and $1_U(x)$ the indicator function. To avoid several fits along the same direction we choose a constant $0 < A \leq \pi/2$, so that the minimal angle between α_i and $\text{span}(\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_M)$ is larger than A . A choice of $A = \pi/2$ would lead to an orthogonal system of projection vectors α_i .

If we use polynomial splines of degree q to approximate m_j , then the minimization problem has a unique solution and we can apply model selection criteria like the final prediction error (FPE) or generalized cross validation (GCV) based on the residual sum of squares (RSS).

$$\begin{aligned} RSS &= \sum_{i=1}^n (y_i - \hat{m}(x_i))^2 1_C(x_i) \\ FPE &= \frac{n_C + \text{trace}(P_n I_C)}{n_C - \text{trace}(P_n I_C)} \frac{RSS}{n_C} \\ GCV &= \left(\frac{n_C}{n_C - \text{trace}(P_n I_C)} \right)^2 \frac{RSS}{n_C} \end{aligned}$$

with n_C the number of observations, P_n the smoother matrix which minimizes (1.4) ($y = P_n x$, $y = (y_1, \dots, y_n)$, $x = (x_1, \dots, x_n)$) and I_C a diagonal matrix with a 1 in the diagonal if $x_i \in C$ and 0 otherwise.

1.2.6.6 Adjoint Projection Pursuit Regression. Duan (1990) proposed an algorithm called “adjoint projection pursuit regression” which is similar to sliced inverse regression. In linear regression the least-squares solution $L(Y | \alpha^T X)$ solves the equation

$$\text{cov}(X, Y - L(Y | \alpha^T X)) = 0. \quad (1.5)$$

In comparison, each step in PPR solves the equivalent nonparametric problem

$$\text{cov}(X, R - E(R | \alpha^T X)) = 0$$

for the residual random variable R instead of Y . From linear algebra, e.g. in Halmos (1958) we have the notation of adjoint linear operators. For the linear least squares regression it follows that α not only solves the equation (1.5), but also the equations

$$\text{cov}(X - L(X | \alpha^T X), Y) = 0 \quad \text{and} \quad (1.6)$$

$$\text{cov}(X - L(X | \alpha^T X), Y - L(Y | \alpha^T X)) = 0. \quad (1.7)$$

Duan (1990) also formulated the “adjoint” problem for the nonparametric case

$$\text{cov}(X - E(X | \alpha^T X), R) = 0.$$

The advantage of estimating $E(X | \alpha^T X)$ rather than $E(R | \alpha^T X)$ is that we can delete or modify outliers in a realization of X . This moves the $E(X | \alpha^T X)$ more to $L(X | \alpha^T X)$ and does not affect the estimation of α whereas a modification of y will affect the estimation of α .

The disadvantage is that the computation will be more complex than in PPR. To ensure a unique solution for α it is necessary that the true model is a PPR model and moreover that $m_{[j]}$ is strictly monotonous. Donoho and Johnstone (1989) showed that any reasonable function can be approximated in this way if M tends to infinity. The close relationship of PPR to Neural Networks with monotonous activation functions implies that the monotonicity is not a too strong assumption. Neural Networks with one hidden layer can also approximate any reasonable function when the number of hidden units tends to infinity. See more about the relationship between Neural Networks and PPR in section 1.3. Additionally, Duan established a fixpoint theorem under the same conditions.

The estimation process works as follows:

1. Choose an initial estimate $\alpha_{[j,0]}$, e.g. the least squares estimate.
2. Compute the “modified regressor”

$$\tilde{x}_i = L(x \mid \alpha_{[j,i]}^T x) + x - E(x \mid \alpha_{[j,i]}^T x)$$

by linear and nonparametric regression. Duan sliced $\alpha_{[j,i]}^T x$ and took the sample mean over each slice for estimating $E(x \mid \alpha_{[j,i]}^T x)$.

3. Compute the new $\alpha_{j,i+1}$ by solving the linear regression of $r^{(j)}$ on \tilde{x}_i .
4. Repeat the last steps until some convergence criterion is fulfilled, for example, when the angle between $\alpha_{[j,i]}$ and $\alpha_{[j,i+1]}$ falls below some threshold.

1.2.6.7 Generalized Projection Pursuit Regression.

Roosen and Hastie (1993) developed a framework for generalized projection pursuit regression (GPPR). In a classification framework PPR is often used as a regression method which can not take into account the binary structure of the problem.

In analogy of generalized linear models (GLM) in McCullagh and Nelder (1989) and GAM in Hastie and Tibshirani (1990), we fit

$$\sum_{i=1} \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i) = \hat{\eta}_i = g(y_i)$$

with g a known link function and y has a distribution in the exponential family

$$f_Y(y, \theta, \phi) = \exp \left\{ \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\}$$

In the case that g is the identity we get the standard PPR. To compute the solution of the system we have to combine the PPR algorithm with the iteratively reweighted least squares (IRLS) algorithm:

1. Compute y_i from the data, maybe with some adjustment to avoid expressions like $\log(0)$, $\hat{\eta}_{i,0}$ and weights $\hat{w}_{i,0}$.
2. Use a weighted PPR which minimizes

$$\sum_{i=1}^n \hat{w}_{i,k} \left(\hat{\eta}_{i,k} - \sum_{j=1}^M \hat{m}_{[j]}(\hat{\alpha}_{[j]}^T x_i) \right)^2$$

to get a new predictor $\hat{\eta}_{i,k+1}$ (eventually use backfitting). Compute new weights $\hat{w}_{i,k+1}$ from the predictor $\hat{\eta}_{i,k+1}$.

3. Repeat the last step until the convergence criterion

$$\delta(\hat{\eta}_k, \hat{\eta}_{k+1}) = \frac{\|\hat{\eta}_k - \hat{\eta}_{k+1}\|}{\|\hat{\eta}_k\|}$$

is below some threshold.

4. Finally prune some of the $\hat{m}_{[j]}$. Drop the term of least importance and backfit the terms before.

Roosen and Hastie (1993) compared three possibilities of fitting binary data with PPR itself with an identity link function and weights equal to one, a logistic link with a binomial random component (logistic projection pursuit (LPP)) and a logistic link with a gaussian random component (squared error logistic response (SELR)). The five functions in section A.1 are modified by enclosing them in a logistic function with a scale parameter which describes the “pureness” of the data. To have purer data means that the regions of 0 and 1 are better separable.

Roosen and Hastie (1993) notified numerical problems with SELR in the case of pure data. However, LPP seems to perform slightly better than PPR in the case of pure data. Otherwise, PPR seems to be slightly better, but not significantly.

1.2.7 Applications of Projection Pursuit Regression

In the scientific literature we find some applications in chemistry and one in military science (Tian and Rong, 1993).

Beebe and Kowalski (1988) examined the behaviour of an array of Ion-selective electrodes in solutions containing mixtures of Na^+ and K^+ . The following equations are used to compute the potential of the j th electrode

$$E_{ij} = E_j^0 + S_j \log(c_{iNa^+} + K_j c_{iK^+})$$

with E_j^0 a electrode specific potential intercept, S_j the slope and K_j the selectivity coefficient. The logarithmic relationship arises from the theory.

In a first step, Beebe and Kowalski replaced the function \hat{m}_1 by a logarithm. This estimate performs better than the PPR. In the second step they did a PPR and computed the quantities E_0 , S and K . Table 1.1 shows the results compared with an earlier estimation. We can not see any significant difference. In fact, a further investigation shows that the choice of the logarithm is quite reasonable from the PPR model and the expected behaviour from the theory has been found.

The interpretability is easily achieved, since the goal of the application is to compare a model derived from theory with practical experiences.

Sekulic, Seasholtz, Wang, Kowalski, Lee and Holt (1993) did a large study

| Electrode | PPR model | | | Simplex model | | |
|-----------|-----------|-------|-------|---------------|-------|-------|
| | E_j^0 | S_j | K_j | E_j^0 | S_j | K_j |
| 1 | 44.50 | 51.63 | 0.43 | 44.51 | 51.64 | 0.44 |
| 2 | -5.36 | 49.70 | 6.51 | -5.36 | 49.69 | 6.51 |
| 3 | 45.47 | 44.81 | 2.27 | 45.37 | 44.78 | 2.39 |
| 4 | 63.25 | 55.18 | 3.07 | 63.07 | 55.07 | 3.18 |
| 5 | 41.81 | 52.52 | 0.70 | 41.81 | 52.72 | 0.70 |

Table 1.1 Estimated constants by a PPR model and another method.

with 6 datasets (1 simulated, 5 real) and 9 regression methods (principal component regression (PCR), partial least squares regression (PLS), nonlinear PCR, nonlinear principal component regression nonlinear (PLS), nonlinear partial least squares regression, locally weighted regression (LWR), projection pursuit regression, alternating conditional expectation (ACE), multivariate adaptive regression splines (MARS), neural networks (NN)) to find out which of the methods is most appropriate for multivariate calibration. As in Beebe and Kowalski (1988) the aim is to predict a new sample from the regression surface. For the model validation they chose 2 criteria: the root mean squared error of cross validation (RMSECV) if only calibration samples are available and the root mean squared error of prediction (RMSEP) if calibration and prediction samples are available.

| Data set | Calibration samples | Prediction samples | Number of variables | Model validation |
|-----------------------|---------------------|--------------------|---------------------|------------------|
| Emission (artificial) | 30 | 10000 | 40 | RMSEP |
| Taguchi (2x) | 50 | 50 | 8 | RMSEP |
| FIA | 46 | - | 100 | RMSECV |
| 3M | 163 | - | 32 | RMSECV |
| Cargill-1 | 208 | 195 | 350 | RMSEP |
| Cargill-2 (2x) | 60 | - | 700 | RMSECV |

Table 1.2 Characteristics of data sets.

Emission is constructed to simulate infrared emission of a thin layer of material on a hot metal surface. The emission is simulated for 40 wave lengths. Although PPR should be able to handle such a number of vari-

ables the authors have used a partial least squares regression to reduce the dimensionality of the data set (also for MARS and ACE).

Taguchi has been taken from an array of 8 Taguchi gas sensors to measure responses to varying mixtures of toluene and benzene. Here are two responses present, one for toluene and one for benzene. A dimension reduction via PCA has been done for LWR.

FIA samples consists of varying concentration of iron and nickel . Dimension reduction has been done for LWR, PPR, ACE and MARS.

3M is from a quality control with 32 optical measurements . Again dimension reduction has been done by PLS before using PPR, ACE and MARS.

Cargill-1 are samples from soybeans to predict the oil content of the beans. A dimension reduction has been done by PLS before using PPR, ACE and MARS.

Cargill-2 are samples of corn gluten meal from a wet milling process . Two outputs have been supplied: concentrations of moisture and protein. Once again dimension reduction has been done by PLS before using PPR, ACE and MARS and PCA before using LWR and NN.

As the result of the study Sekulic et al. (1993) found that the nonlinear methods perform best, but nearly equally well on all datasets. The only exception are the Taguchi data where the NN performs much better than all other methods. Unfortunately, the authors do not give much details about the differences in the models neither any information about the time to compute the models.

Recently van Leeuwen, Jonker and Gill (1994) has been compared PPR with PCR and NN for prediction of octane number in gasoline which is a key property. The octane quality is determined by the composition of the gasoline which can be measured using a gas chromatograph. The high quality chromatograph identifies 418 individual components and we classify them in 89 different PIANO groups (paraffines, iso-paraffines, aromates, naphthenes and olefins). For each sample and each PIANO group the amount of carbon is measured. For all 824 gasoline samples the research octane number (RON) is known, and obvious outliers are removed before the analysis. As a model selection criteria RMSEP is used again. It turns out that PPR (0.34) is slightly better than NN (0.35) and PCR (0.36). But the models are quite different, in PPR 1 smoothing function, in NN 4 hidden units and in PCR 50 principal components are used. The relationship turns out to be only slightly non-linear.

The PPR has been applied to the whole dataset and it reveals two steps. The two steps are induced by the occurrence of some outliers (different compositions but the same RON). This leads to an inspection of the RON values which are given with one digit after the decimal point. It turns out that the

distribution of the digits after the decimal point is not uniform. In the class with 0 (as digit after the decimal point) are twice as much observations as in any other class.

The authors remark that no guidelines are established, neither for PPR nor for NN (including the choice of parameters), what makes the use of these techniques time consuming for an unexperienced user.

1.2.8 Software

| Symbol | Definition |
|--------|---|
| LMV | logarithm of the median value of owner-occupied homes |
| CRIM | per capita rate by town |
| ZN | proportion of a town's residential land zoned for lots greater than 25000 square feet |
| IND | proportion of nonretail business acres per town |
| CHAS | Charles river dummy variable with value 1 if tract bounds on the Charles river |
| NOX | nitrogen oxide concentration (parts per hundred millions) squared |
| RM | average number of rooms squared |
| AGE | proportion of owner-occupied units built prior to 1940 |
| DIS | logarithm of the weighted distances to five employment centers in the Boston region |
| RAD | logarithm of index of accessibility to radial highways |
| TAX | full-value property-tax rate (per 10000 US-\$) |
| PTR | pupil-teacher ratio |
| B | $(Bk - 0.63)^2$ where Bk is the proportion of blacks in the population |
| LSTA | logarithm of the proportion of the population of lower status |

Table 1.3 Variables of the Boston housing dataset

1.2.8.1 Boston Housing Data. Harrison and Rubinfeld (1978) collected 14 variables (see Table 1.3) for 506 districts in the Boston standard metropolitan statistic area.

After they transformed the variables they did a linear regression to estimate the relationship between MEDV and the other variables ($R^2 = 0.806$).

The standardized coefficients of the linear regression and the coefficients of the projection vectors of PPR in XploRe and S-Plus for the Boston housing data can be found in Table 1.4 (Klinke, 1997). The first projection is equivalent to the linear regression.

| Proj | CRIM | ZN | IND | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTR | B | LSTA |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Lin.Reg. | -0.02 | +0.00 | +0.00 | +0.16 | -0.01 | +0.01 | +0.00 | -0.33 | +0.17 | -0.00 | -0.05 | +0.64 | -0.65 |
| XploRe 1 | +0.01 | +0.00 | +0.00 | -0.25 | +0.01 | -0.01 | -0.00 | +0.31 | -0.14 | +0.00 | +0.10 | -0.31 | +0.83 |
| XploRe 2 | -0.00 | +0.00 | +0.02 | -0.63 | -0.00 | -0.00 | +0.00 | -0.64 | +0.20 | +0.00 | -0.01 | -0.35 | -0.07 |
| XploRe 3 | +0.00 | -0.00 | +0.00 | +0.13 | -0.00 | -0.03 | +0.00 | +0.28 | +0.10 | +0.00 | -0.01 | -0.66 | -0.67 |
| XploRe 4 | -0.00 | -0.00 | -0.00 | -0.76 | +0.00 | -0.00 | -0.00 | -0.13 | +0.16 | +0.00 | -0.01 | -0.59 | -0.11 |
| XploRe 5 | +0.02 | -0.00 | +0.00 | -0.06 | -0.01 | -0.08 | +0.00 | -0.05 | -0.28 | -0.00 | -0.08 | +0.02 | -0.95 |
| XploRe 6 | +0.00 | +0.00 | +0.02 | +0.11 | -0.00 | +0.00 | +0.00 | +0.07 | +0.02 | +0.00 | +0.02 | +0.99 | +0.02 |
| XploRe 7 | +0.00 | -0.00 | +0.01 | +0.37 | +0.00 | -0.01 | +0.00 | +0.05 | +0.02 | -0.00 | -0.04 | -0.92 | -0.08 |
| XploRe 8 | -0.00 | -0.01 | +0.01 | +0.06 | -0.01 | +0.00 | +0.01 | +0.41 | +0.20 | -0.00 | +0.04 | -0.89 | -0.05 |
| S-Plus 1 | -0.02 | -0.00 | +0.00 | +0.11 | -0.01 | +0.01 | -0.00 | -0.21 | +0.22 | -0.00 | -0.08 | +0.44 | -0.82 |
| S-Plus 2 | +0.00 | +0.00 | -0.01 | -0.01 | -0.02 | -0.00 | +0.00 | -0.92 | +0.04 | -0.00 | +0.01 | -0.36 | -0.01 |
| S-Plus 3 | +0.01 | +0.00 | -0.00 | -0.03 | +0.00 | +0.02 | -0.00 | -0.42 | -0.33 | -0.00 | +0.02 | +0.51 | +0.65 |
| S-Plus 4 | -0.02 | +0.00 | -0.01 | -0.24 | -0.00 | +0.04 | -0.00 | -0.37 | +0.30 | +0.00 | +0.03 | -0.34 | +0.76 |
| S-Plus 1 | -0.02 | -0.00 | -0.00 | +0.11 | -0.01 | +0.02 | +0.00 | -0.27 | +0.14 | -0.00 | -0.08 | +0.67 | -0.66 |
| S-Plus 2 | +0.00 | +0.00 | -0.02 | +0.01 | -0.02 | -0.00 | +0.00 | -0.97 | +0.07 | -0.00 | +0.02 | -0.25 | +0.02 |
| S-Plus 3 | +0.00 | +0.00 | -0.01 | +0.09 | -0.00 | +0.02 | -0.01 | -0.33 | -0.20 | -0.00 | +0.01 | +0.70 | +0.59 |
| S-Plus 4 | -0.02 | +0.00 | -0.02 | -0.28 | -0.01 | +0.05 | +0.00 | -0.41 | +0.31 | +0.00 | +0.03 | -0.40 | +0.70 |
| S-Plus 5 | +0.00 | -0.02 | +0.11 | +0.08 | +0.02 | +0.04 | -0.02 | +0.82 | +0.24 | -0.00 | +0.02 | -0.43 | +0.23 |
| S-Plus 6 | -0.01 | +0.01 | -0.01 | +0.68 | -0.02 | -0.02 | +0.00 | -0.21 | -0.60 | +0.00 | +0.11 | +0.29 | -0.21 |
| S-Plus 7 | +0.03 | -0.00 | -0.03 | -0.38 | -0.00 | -0.03 | +0.00 | -0.35 | +0.40 | -0.00 | -0.00 | +0.25 | -0.71 |
| S-Plus 8 | -0.01 | -0.00 | +0.15 | +0.42 | -0.08 | -0.01 | +0.01 | -0.61 | +0.13 | +0.00 | -0.17 | -0.12 | -0.60 |

Table 1.4 Projection vectors from PPR and the linear regression for the Boston Housing Data. The coefficients of the linear regression (first line) are rescaled so that the euclidean norm is 1. Then, we see the projections of a 8-term-PPR-fit in XploRe and the projection of a 4-term- and 8-term-fit in S-Plus.

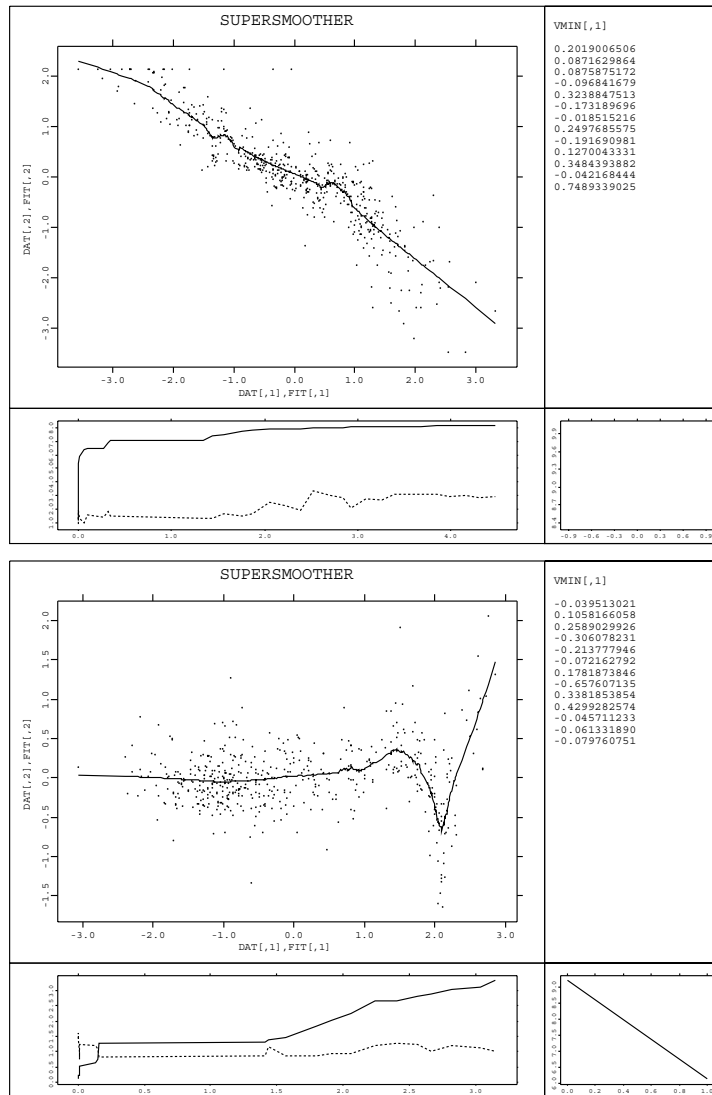


Fig. 1.2 First and second term of eight term PPR fit of the Boston housing data .

Figure 1.2 shows the first and second term of a PPR fit in XploRe. Since the linear regression fits very well it is sufficient to use just one term. The graphic shows that the first term gives a R^2 about 0.9. The second term gives a R^2 about 0.6 on the residuals of the one term fit.

1.2.8.2 S-Plus. S-Plus offers an intrinsic function for doing PPR:

```
ppr <- ppreg (x,y,termno)
```

with x a multivariate variable and y the 1-dimensional response. `termno` gives the minimal number of terms which will be used. `ppr` itself is a list of the following components

| | |
|-----------------|---|
| ypred | the residuals of the fitted model |
| fl2 | the squared residuals divided by all corrected sums of squares |
| alpha | a matrix of projection directions used |
| beta | a matrix of the weights for every observation y_i |
| z | a matrix which contains $\alpha_i^T X$ |
| zhat | a matrix which contains the smoothed values $\sum_{i=1}^{termno} g_i(\alpha_i^T X)$ |
| allalpha | three-dimensional array which contains the fitted alphas for every step |
| allbeta | three-dimensional array which contains the fitted betas for every step |
| esq | contains the fraction of unexplained variance |
| esqrsp | contains the fraction of unexplained variance for every observation. |

The algorithm is based on the work of Friedman (1985) and uses backfitting. Since `ppreg` is an intrinsic function we do not have any insight in the algorithm.

1.2.8.3 XploRe. The PPR is part of the ADDMOD-library of XploRe 3.2 (additive modeling). We have two macros available, `PPR` does the PPR, `PPRINTER` helps to analyze the found projections and fits. The calls are

```
(xs ys vs fs) = PPR (x cmd) and PPRINTER(x vs fs).
```

The PPR-macro allows an interactive fitting of a PPR model with different smoothers. The macro call is:

```
LIBRARY("addmod")      ; Load the addmod library
x = READ("djppr")      ; Read a dataset generated from
                        ; Donoho and Johnstone (1989)
(xs ys vs fs) = PPR(x) ; Do the PPR
```

From the input variable x it is expected that the last column contains the response. The outputs `xs` and `ys` contain the standardized data. `vs` contains the projection vectors found by a m -term fit and `fs` the fitted function values.

When you enter the macro you can choose among different smoothing methods and smoothing parameters.

As a result you will get a sequence of pictures as in Figure 1.2. In the large window you see the current fit and the projected data. The window beyond the large window shows how the error function will increase by changing the projection vector. The dotted line shows the value of the interpretability index of Morton. The right lower window shows how the error in each term decreases and the upper window shows the actual projection.

The algorithm in **S-Plus** was designed to achieve a result quickly. As a consequence, **S-Plus** uses an intrinsic function for doing PPR. **XploRe** allows the user to have a look at the algorithm since everything is written in macro language. Of course the disadvantage is that **XploRe** needs 15 minutes to fit one term for the Boston housing data on a 486DX4-100 PC.

1.3 NEURAL NETWORKS

1.3.1 Relationship between Projection Pursuit Regression and Neural Networks

In recent years artificial neural networks (ANN) became very popular for pattern recognition, classification and forecasting problems in medicine, economics, industry, molecular biology and several other disciplines. In practical contents more publications about applying feedforward neural networks (FNN) than for applying PPR models can be found, although both FNN's and PPR models have the property of being universal approximators. One reason will surely be that the computational algorithm of FNN's is much easier to understand and therefore easier to program. But, also many applicants seem to be attracted by the magic within the concept of artificial neural networks coming from the idea to model the function of the brain. In the following subsections we will shortly give an impression about the basic concept of single-hidden layer feedforward neural networks and their relation to PPR. For this we will go into the model architecture, learning algorithms, approximation and generalization properties and regularization of neural networks.

1.3.2 The Model Architecture

For the comparison of neural networks with PPR we will consider the single-hidden layer feedforward network. To estimate the regression function

$$E[Y|X = x] = m(x_1, \dots, x_p, \theta), \quad \theta = (\theta_1, \dots, \theta_l) \quad (1.8)$$

we can build a neural network consisting of p input units, one layer of a certain

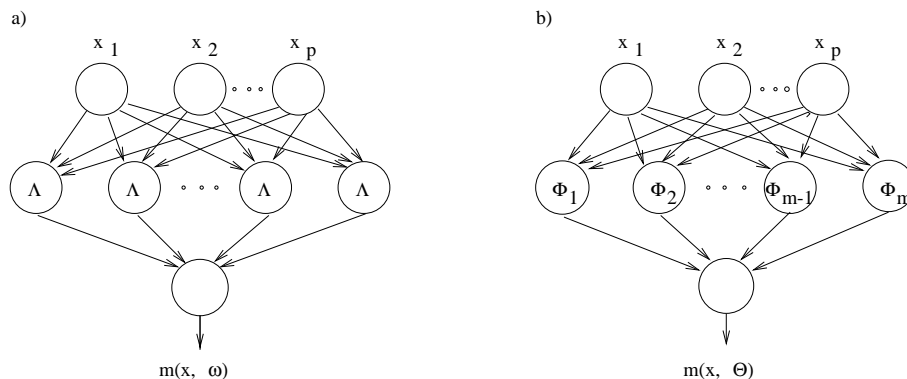


Fig. 1.3 Architecture of a) one hidden layer FNN with a logistic function Λ as activation function and b) a PPR model with smoothers $\Theta_1, \dots, \Theta_m$.

number of hidden units and one output unit forming the output layer. There are weighted connections between the layers in the direction from the input to the output. The input units represent the input variables, or predictor variables in statistical notation. The input of the hidden units is then usually the weighted sum of the values of the input variables whereas the output units have usually the weighted sum of the outputs of the hidden units as input. The output of each unit is determined by a prespecified activation function which is often taken to be the identity (for the input units), linear, logistic, or a threshold function (for hidden and output units). Using the “tanh” function with range $(-1,1)$ for the hidden units instead of the logistic function with range $(0,1)$ is sometimes said to be practically more convenient by resulting in faster convergence. But, theoretically there is no difference, because they differ only by a linear transformation. Taking m logistic hidden units and a linear output unit we get the following model equation for our regression function $m(x, \theta)$

$$m(x, w) = w_0^{(2)} + \sum_{j=1}^m w_j^{(2)} \Lambda \left(w_{j0}^{(1)} + \sum_{i=1}^p w_{ji}^{(1)} x_i \right)$$

with $\Lambda(z) = \frac{\exp(z)}{1+\exp(z)}$ the logistic function and $\omega = (w_{00}^{(1)}, \dots, w_{mp}^{(1)}, w_0^{(2)}, \dots, w_m^{(2)})$. The absolute parameters w_0 and the w_{j0} are called “bias” or “threshold parameters”. The similarity to a PPR model is that both project the input vector by the first layer of weights onto a one-dimensional hyperplane followed by a nonlinear transformation by the activation functions of the hidden units and a final linear combination giving the output value. Thus, both reduce the dimension of the feature space by taking the sum of univariate nonlinear functions of linear combinations of the input variables to overcome the well-known ‘curse of dimensionality’. One main difference is that the activation

functions in a PPR model can be different for each hidden unit and are estimated nonparametrically, whereas they are prespecified parametric functions (e.g., logistic) in FNN's.

1.3.3 Parameter Estimation

Once, we have defined a network architecture the network has to be trained. To train a neural network means to estimate the weights of the connections within the network. Regularization parameters like the weight decay parameter are optimized separately. Typically, the parameters in a neural network are estimated simultaneously, whereas those in a PPR model are fitted successively for each hidden unit. After the projection direction is represented by the weights between the input and the hidden layer of a PPR model is found, the activation function of the current hidden unit is adapted by some smoothing technique. Then, the weights from the hidden to the output unit can be estimated by a least-squares procedure. Finally, using backfitting all weights in the model can be updated together. There are hidden units added to the model until a certain fitting criterion is reached.

Common error functions for FNN's are the mean squared error

$$E_S = \sum_{s=1}^N \|y^s - m(x^s; \theta)\|^2,$$

or the Kullback-Leibler distance

$$E_{KL} = \sum_n \sum_k \left\{ y^s \log \left(\frac{y^s}{m(x^s; \theta)} \right) + (1 - y^s) \log \left(\frac{1 - y^s}{1 - m(x^s; \theta)} \right) \right\}.$$

Obviously, its minimization is equivalent to minimize the negative log-likelihood function

$$E_{ML} = -\log(L) = -\sum_n \sum_k \{y^s \log m(x^s; \theta) + (1 - y^s) \log(1 - m(x^s; \theta))\}.$$

For the minimization we have to choose an appropriate optimization method which depends on the smoothness of the activation functions.

The first and well-known algorithm to train networks with hidden layers was the classical backpropagation algorithm (generalized delta rule) that was popularized by Rumelhart, Hinton and Williams (1986). This algorithm calculates the derivatives of the error function with respect to the weights recursively from the output to the input layer using the chain rule. Of course, it has to be assumed that the activation functions are differentiable. Originally, the backpropagation algorithm was only mentioned with respect to the simple gradient descent method to minimize the sum-of-squares error function of a multilayer perceptron. But, actually the idea of propagating the error

backwards through the network to compute derivatives can also be applied to other kinds of networks, other error functions and to evaluate the second derivatives like the Hessian matrix, that is useful for some more higher level optimization algorithms. An exact evaluation of the Hessian matrix is given in Bishop (1995).

The original backpropagation algorithm works as follows. The weights are updated in a steepest descent direction

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

with E the error function and step size η , also called learning rate, hold fixed. Two common update types are possible, the *batch* type, where the weights are updated after the presentation of the complete training set, and the *sequential* type, where the weights are updated after each presentation of one element of the training set. This only makes sense, of course, for the data arranged in a random order. The advantage of the *sequential* over the *batch* method is, that if there is redundant information in the data set, say, the data set is repeated 10 times, then the *batch* method needs 10 times as long, while the *sequential* method will be unaffected by the replication of the data. Another advantage is that the *sequential* method is basically a stochastic method that has the chance to escape from local minima. In general it is not the goal to find the global minimum of the error function, but to find a “good” local minimum, where the generalization error is supposed to be small. It should also be mentioned that there is a symmetry of the weight space of a feedforward neural network. For a neural network with M hidden units there are $M!2^M$ points in the weight space that generate the same network output. But, for the matter of the training it is of little importance, because the minima are just replicated several times.

Because backpropagation can be very slow, efforts were made to speed up the convergence. One well-known modification is to add a ‘momentum’ term that changes the step width depending on the previous one. The updating formula becomes

$$\Delta w_{ij} = -\eta \left[(1 - \alpha) \frac{\partial E}{\partial w_{ij}} + \alpha (\Delta w_{ij})_{previous} \right].$$

But this introduces one more parameter that has to be tuned in some way. To speed up the convergence η and α can be chosen adaptively as the iteration goes on. Another often implemented algorithm is the Quickprop algorithm (Fahlman, 1989), that treats the dimensions of the objective function as independent. To approximate the error surface by a polynomial of each of the weights it uses two successive evaluations of the error function and an evaluation of its gradient. The minimum of the parabola is finally taken for the next weight update. But also general unconstrained optimization algorithms as described in 1.4 can be used.

1.3.4 Initializing the Weights

To fit a neural network model we have to choose starting values for the weights. Usually, they are chosen randomly from a range of small values. They should be random to avoid getting stuck in the first bad local minimum. To not to drive the sigmoidal activation functions into the saturation regions where the first derivative is very small they have to be small enough, but not too small in order to avoid the linear part of these activation functions. One general recipe is to generate the starting values for each unit from a Gaussian with mean zero and standard deviation $\sigma \propto d^{-\frac{1}{2}}$ with d the number of weights feeding the unit. This would result in an input to the unit with variance 1, because

$$\sum_1^d \sigma^2 = d\sigma^2 = 1.$$

Another suggestion is made by Le Cun (1996), who recommends the following activation function

$$a(x) = 1.7159 \tanh\left(\frac{2}{3}x\right) + bx.$$

The main properties of this activation function are that $a(-1) = -1$ and $a(1) = 1$ which avoids the saturation of the output units for binary target values, and the maximum of the second derivative is at $x = 1$. The linear term can also be useful to avoid saturation areas.

1.3.5 Model Complexity and Regularization

Both PPR models and FNN's can approximate any continuous function to any degree, provided the model is complex enough. The complexity of a model is given by the effective number of its parameters, that increases with an increasing number of units in the hidden layer of a FFN. Analogously, the complexity of a PPR model increases as the number of its nonlinear terms increases. But the "universal approximator" property does not say anything about the statistical properties of the approximation based on noisy data of limited sample size. White (1990) shows how the complexity of a FFN must grow in relation to the size of the data set in order to be consistent. But in practice we want to find the optimal model complexity given limited and noisy data. In the statistical literature it is often focused on the 'trade-off between bias and variance'. That is, the mean-squared error can be divided in a bias and a variance component of the estimated regression function. Both can contribute to poor performance. Asymptotically, both should approach zero for a consistent estimate for increasing sample size. But in practice, with finite sample sizes it is a complicated issue to balance them (Geman, Bienenstock and Doursat, 1992). In nonparametric regression analysis (e.g.,

kernel regression) this is done by adjusting a bandwidth parameter which is responsible for the smoothness of the fitted function. For a bandwidth too large, the fitted function will be very smooth and will not be able to detect details of the real regression function. Thus, a bias component is introduced. For a very small bandwidth the estimator will learn the noise of the data which increases the variance component of the error function and thus will not be able to generalize well. For FNN's there is a number of heuristic methods to find the optimal complexity. Such methods range from regularization, early stopping, growing and pruning algorithms to using committees of networks.

Regularization. One possibility to handle the bias-variance-tradeoff is regularization that is to add a penalty term to the error function. In the neural network literature weight decay is one of the most often used approach to regularization. Weight decay means to add a fraction d of the sum of squared weights of the current model to the error function and we get

$$E_d = E_S + d \sum_{ij} w_{ij}^2$$

with w_{ij} the weights in the network in order to penalize large weights during the training process. Before applying this regularization method the input variables should be normalized. One way to choose the weight decay parameter d is by cross-validation (CV) (Stone, 1974; Efron and Tibshirani, 1993), that is, by minimizing an estimate of the generalization ability with respect to the following algorithm. We divide the training set D in a specified number of subsets D_j of sizes N_j

$$\bullet D = \cup D_j, \quad \cap D_j = \emptyset, \quad \sum N_j = N,$$

train the network for all but one of these subsets $D^{(-D_j)}$ and estimate the generalization error CV_{D_j} using the subset left out as a test set

$$\bullet CV_{D_j}(d) = \frac{1}{N_j} \sum_{(x_l, Y_l) \in D_j} (Y_l - f_d(x_l, D^{(-D_j)}))^2.$$

This is repeated for all subsets. Finally, the generalization error is averaged over all subsets which gives the cross-validation error CV_D

$$\bullet CV_D(d) = \frac{1}{k} \sum_{j=1}^k CV_{D_j}(d) \rightarrow_d \min!.$$

The optimal d (out of a given one-dimensional grid) is that with minimal CV_D within the training data set. One disadvantage of weight decay is that it is not consistent with linear transformations of the input or output units. Bishop (1995) gives a regularizer that is scale-invariant for the weights and that is also shift-invariant for the threshold parameters. By extending the backpropagation algorithm to the second derivatives (Bishop, 1993) it is also possible to use curvature information in the regularization term. Closely related to the technique of regularization is "training with noise" (Sietsma and Dow, 1991). Some noise that has zero mean and is uncorrelated between the inputs

is added to each input before it is used for training. This procedure is closely related to regularization (Bishop, 1995) and can improve the generalization performance. Another way to improve the generalization performance is by “soft weight sharing” (Nowlan and Hinton, 1992), which means that groups of weights are encouraged to have similar values. This is reached by assuming a mixture of Gaussians

$$p(w) = \sum_{j=1}^J \alpha_j \Phi_j(w)$$

as the probability distribution of the weights. By taking the negative log-likelihood

$$-\sum_i \log p(w_i) = -\sum_i \log \left(\sum_{j=1}^J \alpha_j \Phi_j(w_i) \right)$$

as a regularization term.

Early Stopping. An alternative approach to regularization is “early stopping”. While the training error is in general monotonically decreasing during the training process the test error often begins to increase as the network starts to overfit the data. The idea is to reduce the effective number of degrees of freedom of the network by stopping the training process at the point of the smallest test error. It is expected that the resulting network will give a low generalization error for so far unknown examples.

Committees of Networks. Instead of training different networks and choosing the one with the best generalization performance it is also possible to combine the networks together to a committee (Perrone and Cooper, 1992). The output of a committee network is the (weighted) average over the outputs of the individual networks. It can be shown that

$$E_{committee} \leq E_{ave}$$

with E_{ave} the average error of the individual networks.

Number of Hidden Units. There are many heuristic methods to find the optimal number of hidden units. Usually one starts with a very small network and add hidden units sequentially until a certain criteria is fulfilled. “Cascade-Correlation” (CC) (Fahlman and Lebiere, 1990) and “Sequential Network Construction” (SNC) (Moody, 1994) belong to this class of methods. In SNC the optimal number of hidden units is reached when the generalization error begins to increase. The idea of CC is to maximize the correlation between the output of the new weight and the current residual error and to stop when no significant error reduction has occurred. Both are in a certain sense comparable with adding terms in PPR, but not completely analogue. If the resulting network is supposed to be relatively small or if there is a enough computing power available than one can also start with a large network and prune out the unimportant connections or units. Examples for those methods are “Optimal Brain Damage” (LeCun, Denker and Solla, 1990) and “Node

Pruning” (Mozer and Smolensky, 1989).

1.4 OPTIMIZATION METHODS

The estimation of the parameters of a nonlinear regression function

$$f(x, \theta), \quad \theta = (\theta_1, \dots, \theta_l) \quad (1.9)$$

typically requires the minimization of some kind of an error function $h(\theta)$ like the weighted or unweighted squared error or the negative log-likelihood function. Often these minimization problems cannot be solved analytically, and an iterative method has to be applied. Usually, such an error function represents a landscape of hills and valleys, so that we hope to find at least one of the whole lot of local minima. Generally, we decide among three approaches to iterative minimization of a function $h(\theta)$:

1. Methods using second derivatives of h in terms of θ (Newton method, modified Newton methods),
2. methods using the first derivatives of h (simple gradient descent, discrete Newton methods, quasi-Newton methods, conjugate gradient methods) and
3. methods using just function values, but no derivatives of h (Direct search methods, for example, the simplex algorithm).

The choice of the optimization method plays a key role in implementing neural networks and projection pursuit methods. For neural networks this is stressed by LeCun (1996) and for projection pursuit methods by Posse (1995).

For neural networks it seems that Newton methods perform quite well, whereas in projection pursuit models simpler methods like the simplex algorithm, simulated annealing or even random searches are preferred.

1.4.1 Modified Newton Methods

The modified Newton methods make use of both the gradient vector and the matrix of second derivatives, the Hessian matrix, of the function h . For the second derivatives available it is said to be the most robust and reliable method for minimizing a general smooth function. The classical Newton method is modified either in the Hessian to find a descending search direction or in the step length of the Newton step. The idea of the classical Newton algorithm, also called Newton-Raphson algorithm, is as follows:

- Let

$$g(\theta) = \frac{\partial h(\theta)}{\partial \theta} = \left(\frac{\partial h(\theta)}{\partial \theta_1}, \frac{\partial h(\theta)}{\partial \theta_2}, \dots, \frac{\partial h(\theta)}{\partial \theta_l} \right)^T$$

be the gradient vector and

$$H(\theta) = \frac{\partial^2 h(\theta)}{\partial \theta \partial \theta^T} = \left[\left(\frac{\partial^2 h(\theta)}{\partial \theta_i \partial \theta_j} \right) \right], i, j = 1, \dots, l$$

the Hessian matrix.

- Take a second order truncated Taylor series approximation at the point θ^s

$$h(\theta^s + \delta) \approx h(\theta^s) + g(\theta^s)^T \delta + \frac{1}{2} \delta^T H(\theta^s) \delta \quad (1.10)$$

and choose δ to minimize the right side of (1.10) in the s th step,

$$\delta^s = -H^{-1}(\theta^s)g(\theta^s),$$

which is called the Newton step.

For the starting point θ^1 close enough to a local minimum θ^* , the Newton method will converge at a quadratic rate, that is

$$0 \leq \lim_{s \rightarrow \infty} \frac{\|\theta^{s+1} - \theta^*\|}{\|\theta^s - \theta^*\|^2} < \infty.$$

However, there are two main reasons for the necessity of the modification of the classical approach.

1. The Newton step may be too long with the danger of increasing $h(\theta)$. Thus, there is the need of an adjustable step length λ^s .
2. The Hessian may not be positive definite in each iteration, so that it does not ensure a descent direction of the next step. In this case the Hessian H^s is usually replaced by a positive definite matrix \tilde{H}^s .

Thus, a modified s th Newton step would take on the form

$$\delta^s = -\lambda^s \tilde{H}^{-1}(\theta^s)g(\theta^s).$$

For further information about how to get λ^s and \tilde{H} see Seber and Wild (1989). Modified Newton methods tend to have fast local convergence and are able to detect saddle points, because they use curvature information. But the calculation of the second derivatives can be very time-consuming, and much storage place is needed. There are also attractive minimization algorithms using only first derivative information. Examples of such methods are discrete Newton methods using finite differences of the gradient to approximate the Hessian matrix, conjugate-gradient methods based on the concept of conjugate directions, and quasi-Newton methods, also known as variable-metric methods, approximating the Hessian gradually as the iteration proceeds. In terms of abbreviation only the last mentioned method is described here. But first, the principle of the simple gradient descent algorithm will be explained.

1.4.2 Gradient Descent

Since simple gradient descent methods play an important role in many applications of neural network training with respect to back-propagation we want to explain it briefly. Given a starting point θ^1 , we move at each step in the direction of the negative gradient of the function to be minimized and write

$$\delta^s = -\eta g(\theta^s)$$

with $\delta^s = \theta^{s+1} - \theta^s$. But, as simple this method is as difficult it is to choose the learning rate η conveniently. If η is too large the algorithm can lead to an increase of the objective function and finally to a divergent oscillation. Otherwise, if η is too small, it can become inefficiently slow. Naturally, η should be decreased during the minimization process. But, it is very difficult to find the right tuning of this parameter. Also, in the case that the surface of the objective function has a highly variable curvature (eigenvalues of the Hessian are very different) the local gradient does not directly point to the minimum. This can also lead to very slow convergence. There are several heuristic approaches to improve the convergence rate of the gradient descent algorithm in the neural network literature (see 1.3.3).

1.4.3 Quasi-Newton Methods

The most frequently called quasi-Newton methods are the Davidon-Fletcher-Powell (DFP) algorithm and the related Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. The basic idea of variable metric methods is to construct a sequence of matrices $B(\theta^s)$ converging to the Hessian matrix H

$$\lim_{s \rightarrow \infty} B(\theta^s) = H.$$

The current approximation $B(\theta^s)$ of $H(\theta^s)$ is maintained and used to find the direction d^s in the s th step by solving

$$B(\theta^s)d(\theta^s) = -g(\theta^s).$$

The actual step $\delta^s = \lambda^s d^s$ is obtained after a line search along d^s to find λ^s . Looking at the Taylor expansion

$$g(\theta^s + \delta^s) \approx g(\theta^s) + H(\theta^s)\delta^s$$

we can see, that

$$H(\theta^s)\delta^s \approx g(\theta^{s+1}) - g(\theta^s).$$

We would like to use The Hessian matrix here, but only $B(\theta^s)$ is available. Then $B(\theta^{s+1})$ is chosen to satisfy

$$B(\theta^{s+1})\delta^s = g(\theta^{s+1}) - g(\theta^s), \quad (1.11)$$

which is called the *quasi-Newton condition*. In order to preserve the good properties (e.g., positive definiteness) of B^s , an update

$$B^{s+1} = B^s + E^s$$

is made, with E^s of rank one or two, so that (1.11) remains valid. One often mentioned idea of possible updates comes from the Broyden single-parameter rank-2 family (Broyden, 1967), what the DFP and BFGS methods make use of. Theoretically, the positive definiteness of each B^s is preserved. In practice, however, because of roundoff errors, this may not be the case, and the resultant search direction may not be a descend one. Methods to prevent this problem by factorization of B^s are developed (see Seber and Wild (1989) for references).

1.4.4 The Simplex Method

Now we will go into the last type of the mentioned approaches to iterative minimization, that are methods without using derivatives. One fast and reliable method is the nonderivative quasi-Newton method which uses finite differences to estimate the derivatives. One direct search method relying only on the comparison of function values is the well-established simplex algorithm, also called polytope algorithm, of Nelder and Mead (1965). This method is completely unrelated and not to be confused with the famous simplex method for linear optimization.

The downhill simplex method for the minimization of a function h of p variables $\theta_1, \dots, \theta_p$ starts with $p + 1$ starting points which form the vertices of a nondegenerate simplex. The simplex adapts itself to the local landscape and contracts on to at least a local minimum by shifting the vertex with the highest function value. The advantages of this minimization method are the following.

- We do not need to compute derivatives.
- There is no one-dimensional sub-minimization necessary.
- The algorithm is easy to understand and not hard to program.

Although the algorithm is very slow, it can be a robust method for problems whose computational costs are small. The computational algorithm is as follows:

- Choose an initial simplex, that is $p + 1$ linear independent starting

points $\theta^1, \dots, \theta^{p+1}$ in the p -dimensional space. At each step $p+1$ points $\theta^1, \dots, \theta^{p+1}$ are ordered and renamed so that

$$h(\theta^1) \leq h(\theta^2) \leq \dots \leq h(\theta^{p+1}).$$

The point with the maximum function value, θ^{p+1} is reflected through the centroid $\bar{\theta}$ of the points θ^s with $s < p+1$. The resulting new point is

$$\theta^* = (1 + \alpha)\bar{\theta} - \alpha\theta^{p+1},$$

where the *reflection coefficient* α is a positive constant. The further step depends upon $h(\theta^*)$ as follows:

1. If $h(\theta^1) \leq h(\theta^*) \leq h(\theta^p)$, then replace θ^{p+1} by θ^* and reorder, else if
2. $h(\theta^*) < h(\theta^1)$, then do an *expansion*

$$\theta^{**} = \gamma\theta^* + (1 - \gamma)\bar{\theta},$$

where the *expansion coefficient* γ is greater than unity. If $h(\theta^{**}) < h(\theta^*)$, then replace θ^{p+1} by θ^{**} . If $h(\theta^{**}) \geq h(\theta^*)$, then we have a failed expansion and we replace θ^{p+1} by θ^* .

3. If $h(\theta^*) > h(\theta^p)$, so that the new point is still the worst one, then take a new θ^{p+1} which is either the old θ^{p+1} or θ^* , whichever has the lower function value, and do an *expansion*

$$\theta^{**} = \beta\theta^{p+1} + (1 - \beta)\bar{\theta},$$

where the *contraction coefficient* β lies between 0 and 1. Now replace θ^{p+1} by θ^{**} and restart. In the case of $\theta^{**} > \min(\theta^{p+1}, \theta^*)$ replace all the θ^s 's by $(\theta^s + \theta^1)/2$ and restart the process. A possible stopping rule could be to accept θ^1 , if the "standard error" of the function values at the corners of the simplex falls below a pre-specified value.

Acknowledgments

We would like to thank Jörg Polzehl and Jerome H. Friedman for helpful discussions and hints.

REFERENCES

- Ahn, Y. and Rhee, S. (1992). A simulation study on projection pursuit discriminant analysis (korean), *Korean J. Appl. Statist.* **5**: 103–111.
- Aldrin, M., Bølviken, E. and Schweder, T. (1993). Projection pursuit regression for moderate non-linearities, *Computational Statistics & Data Analysis* **16**: 379–403.
- Asimov, D. (1985). The grand tour: A tool for viewing multidimensional data, *SIAM Journal of Scientific and Statistical Computations* **6**(1): 128–143.
- Beebe, K. and Kowalski, B. (1988). Nonlinear calibration using projection pursuit regression: Application to an array of ion-selective electrodes, *Analytical Chemistry* pp. 2273–2278.
- Bishop, C. (1993). Curvature-driven smoothing: a learning algorithm for feedforward networks, *IEEE Transactions on Neural Networks* **4**(5): 882–884.
- Bishop, C. (1995). *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.
- Blough, D. (1989). Multivariate symmetry via projection pursuit, *Annals of the Institute of Statistical Mathematics* **41**: 461–475.
- Chen, H. (1991). Estimation of a projection-pursuit type regression model, *The Annals of Statistics* **19**(1): 142–157.
- Chen, J.-H. (1987). A result on the projection pursuit density estimation (chinese), *J. Systems Sci. Math. Sci.* **7**: 183–192.
- Cook, D., Buja, A. and Cabrera, J. (1993). Projection pursuit indexes based on orthonormal function expansions, *Journal of Computational and Graphical Statistics* **2**(3): 225–250.
- Cook, D. and Cabrera, J. (1992). Projection pursuit indices based on fractal dimension, *Proceedings of the 24th Symposium on the Interface between Computational Science and Statistics*.
- Donoho, D. and Johnstone, I. (1989). Projection-based approximation and a duality with kernel methods, *The Annals of Statistics* **17**(1): 58–106.
- Duan, N. (1990). The adjoint projection pursuit regression, *Journal of the American Statistical Association* **85**: 1029–1038.
- Efron, B. and Tibshirani, R. (1993). *An introduction to the bootstrap*, Monographs on Statistics and Applied Probability, Chapman & Hall.

- Fahlman, S. (1989). Faster learning variations on back-propagation: an empirical study, *in* D. Touretzky, G. Hinton and T. Sejnowski (eds), *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 38–51.
- Fahlman, S. and Lebiere, C. (1990). The cascade-correlation learning architecture, *Tech. report (cmu-cs-90-100)*, Carnegie Mellon University.
- Flick, T., Jones, L., Priest, R. and Herman, C. (1990). Pattern classification using projection pursuit, *Pattern Recognition* **23**: 1367–1376.
- Friedman, J. (1984a). Smart user’s guide, *Technical report no. 1*, Stanford University, Dept. of Statistics.
- Friedman, J. (1984b). A variable span smoother, *Technical report lcs 05*, Stanford University, Dept. of Statistics.
- Friedman, J. (1985). Classification and multiple regression through projection pursuit, *Technical report LCS012*, Department of Statistics, Stanford University.
- Friedman, J. (1987). Exploratory projection pursuit, *Journal of the American Statistical Association* **82**: 249–266.
- Friedman, J., Grosse, E. and Stuetzle, W. (1983). Multidimensional additive spline approximation, *SIAM Journal of Scientific and Statistical Computing* pp. 291–301.
- Friedman, J. and Stuetzle, W. (1981a). Projection pursuit classification, (unpublished manuscript).
- Friedman, J. and Stuetzle, W. (1981b). Projection pursuit regression, *Journal of the American Statistical Association* **76**: 817–823.
- Friedman, J., Stuetzle, W. and Schroeder, A. (1984). Projection pursuit density estimation, *Journal of the American Statistical Association* **79**: 599–608.
- Friedman, J. and Tukey, J. (1974). A projection pursuit algorithm for exploratory data analysis, *IEEE Transactions on Computers C* **23**: 881–889.
- Geman, S., Bienenstock, E. and Doursat, R. (1992). Neural networks and the bias/ variance dilemma, *Neural Computation* **4**(1): 1–58.
- Hall, P. (1988). Estimating the direction in which a data set is most interesting, *Probab. Theory Related Fields* **80**(2): 51–77.
- Hall, P. (1989a). On polynomial-based projection indices for exploratory projection pursuit, *The Annals of Statistics* **17**(2): 589–605.

- Hall, P. (1989b). On projection pursuit regression, *The Annals of Statistics* **17**(2): 573–588.
- Halmos, P. (1958). *Finite-Dimensional Vector Spaces*, Princeton: Van Nostrand.
- Harrison, D. and Rubinfeld, D. (1978). Hedonic prices and the demand for clean air, *Journal for Environmental Economics & Management* **5**: 81–102.
- Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*, Chapman and Hall.
- Hida, H., Tasaki, T. and Goto, M. (1992). Evaluating the adjoint projection pursuit regression through its applications (japanese), *Jap. J. Appl. Statist.* **21**: 101–111.
- Huber, P. (1985). Projection pursuit, *The Annals of Statistics* **13**: 435–475.
- Hwang, J., Lay, S., Maechler, M., Martin, D. and Schimert, J. (1994). Regression modelling in backpropagation and projection pursuit learning networks, *IEEE Transactions on Neural Networks* **5**(3): 342–353.
- Intrator, N. (1990). Combining exploratory projection pursuit and projection pursuit regression with applications to neural networks, *Neural Computation* **5**: 443–455.
- Jee, J. (1985). Exploratory projection pursuit using nonparametric density estimation, *ASA. Proc. of Stat'l. Computing Sect.*, pp. 335–339.
- Johns, M. (1986). Fully nonparametric empirical bayes estimation via projection pursuit, *Adapt. Statist. Proced. Related Topics, John Van Ryzin, Inst. Math. Statist. (Hayward)*, pp. 1–10.
- Jones, L. (1985). On a conjecture of huber concerning the convergence of projection pursuit regression, *The Annals of Statistics* **15**: 880–882.
- Jones, L. (1992). A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training, *The Annals of Statistics* **20**: 608–613.
- Jones, M. and Sibson, R. (1987). What is projection pursuit ?, *Journal of the Royal Statistical Society A* **150**: 1–18.
- Klinke, S. (1997). *Data Structures for Computational Statistics*, Physica Verlag.
- Kruskal, J. (1969). Toward a practical method which helps uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new “index of condensation”, *Statistical Computation*, Academic Press, pp. 427–440.

- Kruskal, J. (1972). Linear transformation of multivariate data to reveal clustering, *Multidimensional Scaling : Theory and Applications in the Behavioural Sciences*, Seminar Press.
- Léger, L. (1991). Additive nonparametric regression methods and the case of projection pursuit regression (italian), *Statistica (Bologna)* **51**: 35–52.
- Léger, L. (1989). Prediction error and model selection in projection pursuit regression (italian), *Statistica (Bologna)* **49**: 537–546.
- Léger, L. and Daudin, J. (1993). Investigation of a nonparametric regression model: Projection pursuit regression (french), *Rev. Statist. Appl.* **41**: 21–48.
- LeCun, Y. (1996). Efficient backprop, <ftp://www.first.gmd.de/persons/Mueller.Klaus-Robert/cookbook.ps.gz>.
- LeCun, Y., Denker, J. and Solla, A. (1990). Optimal brain damage, in D. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann Publisher, pp. 598–605.
- Li, K. (1991). Sliced inverse regression for dimension reduction (with discussion), *Journal of the American Statistical Association* **86**(414): 316–342.
- Maller, R. (1989). Some consistency results on projection pursuit estimators of location and scale, *Canad. J. Statist.* **17**: 81–90.
- McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*, Chapman and Hall.
- Moody, J. (1994). Prediction risk and architecture selection for neural networks, in Charkasky, Friedman and Wechsler (eds), *From statistics to neural networks. Theory and Applications*, Springer, pp. 147–165.
- Morton, S. (1989). *Interpretable Projection Pursuit*, PhD thesis, Department of Statistics, Stanford University.
- Morton, S. (1990). Interpretable projection pursuit, *ASA Proc. Statist. Comput. Sect., Amer. Statist. Assoc. (Alexandria, VA)*, pp. 63–68.
- Mozer, M. and Smolensky, P. (1989). Skeletonization: a technique for trimming the fat from a network via relevance assessment, in D. Touretzky (ed.), *Advances in Neural Information Processing Systems 1*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 107–115.
- Nason, G. (1995). Three-dimensional projection pursuit, *Applied Statistics* **4**(44): 411–430.
- Nelder, J. and Mead, R. (1965). A simplex method for function minimization, *The Computer Journal* **7**: 308–313.

- Nowlan, S. and Hinton, G. (1992). Simplifying neural networks by soft weight sharing, *Neural Computation* **4**(4): 473–493.
- Perrone, M. and Cooper, L. (1992). When networks disagree: Ensemble methods for hybrid neural networks, <ftp://archive.cis.ohio-state.edu>.
- Polzehl, J. (1995). Projection pursuit discriminant analysis, *Computational Statistics & Data Analysis*.
- Portier, K., Dippon, D. and Hetrick, V. R. (1987). An implementation of projection pursuit for classification, *ASA Proc. Statist. Comput. Sect., Amer. Statist. Assoc. (Alexandria, VA)*, pp. 340–344.
- Posse, C. (1992). Projection pursuit discriminant analysis for two groups, *Communications in Statistics A—Theory Methods* **21**: 1–19.
- Posse, C. (1995). Projection pursuit exploratory data analysis, *Computational Statistics & Data Analysis* **20**: 669–887.
- Rejtő, L. and Walter, G. (1990). Generalized projection pursuit regression and density approximation, *Comput. Sci. Statist.: Proc. Symp. Interface, Connie Page, Raoul LePage, Springer-Verlag (Berlin; New York)*, pp. 501–503.
- Rejtő, L. and Walter, G. (1992). Remarks on projection pursuit regression and density estimation, *Stochastic Anal. Appl.* **10**: 213–222.
- Ripley, B. (1994). Neural networks and flexible regression and discrimination, *J. Appl. Statist.* **21**: 39–57.
- Ripley, B. (1996). *Pattern Recognition and Neural Networks*, Cambridge University Press.
- Roosen, C. and Hastie, T. (1993). Logistic response projection pursuit, *Document no. bl011214-930806-09tm*, AT&T Bell Laboratories.
- Roosen, C. and Hastie, T. (1994). Automatic smoothing spline projection pursuit, *Journal of Computational and Graphical Statistics* **3**: 235–248.
- Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning internal representations by error propagation, in D. Rumelhart, J. McClelland and the PDP Research Group (eds), *Parallel distributed Processing: Explorations in the microstructure of cognition, Volume 1: Foundations*, MIT Press, Cambridge, pp. 318–362.
- Seber, G. and Wild, C. (1989). *Nonlinear regression*, Wiley Series in Probability and Mathematical Statistics.
- Sekulic, S., Seasholtz, M., Wang, Z., Kowalski, B., Lee, S. and Holt, B. (1993). Nonlinear multivariate calibration methods in analytical chemistry, *Analyt. Chem.* **65**: 835–845.

- Sietsma, J. and Dow, R. (1991). Creating artificial neural networks that generalize, *Neural Networks* **4**(1): 67–79.
- Silverman, B. (1986). *Density estimation*, Chapman and Hall.
- Stone, M. (1974). Cross-validation choice and assessment of statistical predictions, *Journal of the Royal Statistical Society, Series B* **36**: 111–147.
- Stulajter, F. (1988). Projection pursuit quadratic regression: The normal case, *Appl. Mat.* **33**: 204–212.
- Tian, Z. and Rong, H. (1993). Projection pursuit regression and the treatment of data for all-around missiles, *Chinese J. Appl. Probab. Statist.* **9**: 319–325.
- van Leeuwen, J., Jonker, R. and Gill, R. (1994). Octane number prediction based on gas chromatographic analysis with non-linear regression techniques, *Chemometrics Intellig. Lab. Sys.* **25**: 325–340.
- White, H. (1990). Connectionist nonparametric regression: Multilayer feed-forward networks can learn arbitrary mappings, *Neural Networks* **3**: 535–549.
- Zhu, L. and Li, G. (1991). A consistent estimator for the supremum of the projection index based on sliced inverse regression, *Communications in Statistics A—Theory Methods* **20**: 5–16.

Appendix

A.1 TEST FUNCTIONS FOR PROJECTION PURSUIT REGRESSION

- Simple interaction function

$$g_1(x, y) = 10.391((x - 0.4)(y - 0.6) + 0.36)$$

- Radial function

$$g_2(x, y) = 24.234(r^2(0.75 - r^2)), r^2 = (x - 0.5)^2 + (y - 0.5)^2$$

- Harmonic function

$$g_3(x, y) = 42.659((2 + x)/20 + \Re(z^5)), z = x + iy - 0.5(1 + i)$$

- Additive function

$$g_4(x, y) = 1.3356(1.5(1 - x) + \exp(2x - 1) \sin(3\pi(x - 0.6)^2) + \exp(3(y - 0.5)) \sin(4\pi(y - 0.9)^2))$$

- Complicated interaction function

$$g_5(x, y) = 1.9(1.35 + \exp(x) \sin(13(x - 0.6)^2) \exp(y) \sin(7y))$$

Index

- k*-nn estimator, 8
- R^2 , 15
- Activation function, 28
- Alternating conditional expectation, 21
- Backfitting, 9, 19, 26
- Backpropagation
 - algorithm, 29
 - momentum term, 30
- Boston housing data, 16, 25, 27
- Curse of dimensionality, 7–8, 17, 28
- Data
 - 3M, 22
 - Boston housing, 16, 25, 27
 - Cargill-1, 22
 - Cargill-2, 22
 - Emission, 22
 - FIA, 22
 - Taguchi, 22
- Estimator
 - k*-nn, 8
 - Nadaraya-Watson, 12
- Exploratory projection pursuit, 15
- Final prediction error, 17
- Generalized additive model, 8, 10
- Generalized cross validation, 11, 14, 17
- Generalized partial linear models, 10
- Generalized projection pursuit regression,
 - 19
- Iteratively reweighted least squares, 19
- Kullback-Leibler distance, 29
- Locally weighted regression, 21
- Logistic function, 28
- Median absolute deviation, 15
- Model complexity, 31
- Multivariate adaptive regression splines, 21
- Nadaraya-Watson estimator, 12
- Neural network, 21
- Neural networks
 - weight decay, 32
 - activation function, 28
 - committees, 33
 - cross-validation, 32
 - early stopping, 33
 - error functions, 29
 - model architecture, 27
 - model complexity, 31
 - number of hidden units, 33
 - parameter estimation, 29
 - regularization, 32
 - similarity to a PPR model, 28
 - starting values, 31
- Nonlinear partial least squares regression,
 - 21
- Nonlinear principal component regression,
 - 21
- Partial least squares regression, 21
- Principal component regression, 21
- Projection pursuit regression
 - backfitting, 9
 - basic algorithm, 7
 - convergence rate, 12
 - generalized, 19
 - starting projection, 17
- Projection pursuit
 - classification, 7
 - density estimation, 7, 12
 - discriminant analysis, 7
 - exploratory, 7, 15
 - index, 15
 - logistic, 20
- Quickprop algorithm, 30
- Regression
 - alternating conditional expectation, 21
 - generalized projection pursuit, 19
 - locally weighted, 21
 - multivariate adaptive regression splines,
 - 21
 - nonlinear partial least squares, 21
 - nonlinear principal component, 21
 - partial least squares, 21
 - principal component, 21
 - sliced inverse, 16, 18
- Residual sum of squares, 17
- Root mean squared error of cross
 - validation, 21
- Root mean squared error of prediction,
 - 21–22
- Sliced inverse regression, 16, 18
- Smoothing spline, 13–14

- Spline
 - multivariate adaptive regression, 21
 - smoothing, 13–14
- Sum of the absolute deviation, 15
- Supersmoother, 9, 13
- Training algorithm
 - batch, 30
 - sequential, 30
- Weight decay, 32