3. Nonparametric Regression

The multiple linear regression model is

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$$

where $\mathbb{E}[\epsilon] = 0$, $\mathrm{Var}\,[\boldsymbol{\epsilon}] = \sigma^2$, and ϵ is independent of $x_1, \ldots, x_p$.

The model is useful because:

- it is interpretable—the effect of each explanatory variable is captured by a single coefficient
- theory supports inference and prediction is easy
- simple interactions and transformations are easy
- dummy variables allow use of categorical information
- computation is fast.

3.1 Additive Models

But additive linear fits are too flat. And the class of all possible smooths is too large—the COD makes it hard to smooth in high dimensions. The class of **additive models** is a useful compromise.

The additive model is

$$Y = \beta_0 + \sum_{k=1}^{p} f_k(x_k) + \epsilon$$

where the f_k are unknown smooth functions fit from the data.

The basic assumptions are as before, except we must add $\mathbb{E}[f_k(X_k)] = 0$ in order to prevent identifiability problems.

The parameters in the additive model are $\{f_k\}$, β_0, and σ^2. In the linear model, each parameter that is fit costs one degree of freedom, but fitting the functions costs more, depending upon what kind of univariate smoother is used.

Some notes:

- one can require that some of the f_k be linear or monotone;
- one can include some low-dimensional smooths, such as $f(X_1, X_2)$;
- one can include some kinds of interactions, such as $f(X_1 X_2)$;
- transformation of variables is done automatically;
- many regression diagnostics, such as Cook's distance, generalize to additive models;
- ideas from weighted regression generalize to handle heteroscedasticity;
- approximate deviance tests for comparing nested additive models are avaialable;
- one can use the bootstrap to set pointwise confidence bands on the f_k (if these include the zero function, omit the term);

However, model selection, overfitting, and multicollinearity (concurvity) are serious problems. And the final fit may still be poor.

3.1.1 The Backfitting Algorithm

The backfitting algorithm is used to fit additive models. It allows one to use an arbitrary smoother (e.g., spline, Loess, kernel) to estimate the $\{f_k\}$

As motivation, suppose that the additive model is exactly correct. The for all $k = 1, \ldots, p$,

$$\mathbb{E}[Y - \beta_0 - \sum_{k \neq j} f_k(X_k) \,|\, x_j] = f_j(x_j).$$

The backfitting algorithm solves these p estimating equations iteratively. At each stage it replaces the conditional expectation of the partial residuals, i.e., $Y - \beta_0 - \sum_{k \neq j} \hat{f}_k(X_k)$ with a univariate smooth.

Notation: Let $\boldsymbol{y}$ be the vector of responses, let $\boldsymbol{X}$ be the $n \times p$ matrix of explanatory values with columns $\boldsymbol{x}_{\cdot k}$. Let $\boldsymbol{f}_k$ be the vector whose ith entry is $f_k(x_{ik})$ for $i = 1, \ldots, n$.

For $\boldsymbol{z} \in \mathbb{R}^n$, let $S(\boldsymbol{z} \,|\, \boldsymbol{x}_{\cdot k})$ be a smooth of the scatterplot of $\boldsymbol{z}$ against the values of the kth explanatory variable.

The backfitting algorithm works as follows:

1. Initialize. Set $\hat{\beta}_0 = \bar{Y}$ and set the f_k functions to be something reasonable (e.g., a linear regression). Set the $\boldsymbol{f}_k$ vectors to match.

2. Cycle. For $j = 1, \ldots, p$ set

$$f_k = S(\boldsymbol{Y} - \hat{\beta}_0 - \sum_{k \neq j} \boldsymbol{f}_k \,|\, \boldsymbol{x}_{\cdot k})$$

and update the $\boldsymbol{f}_k$ to match.

3. Iterate. Repeat step (2) until the changes in the f_k between iterations is sufficiently small.

One may use different smoothers for different variables, or bivariate smoothers for predesignated pairs of explanatory variables.

The estimating equations that are the basis for the backfitting algorithm have the form:

$$\boldsymbol{P f} = \boldsymbol{Q Y}$$

for suitable matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$.

The iterative solution for this has the structure of a Gauss-Seidel algorithm for linear systems (cf. Hastie and Tibshirani; 1990, *Generalized Additive Models*, chap. 5.2).

This structure ensures that the backfitting algorithm converges for smoothers that correspond to a symmetric smoothing matrix with all eigenvalues in $(0, 1)$. This includes smoothing splines and most kernel smoothers, but not Loess.

If it converges, the solution is unique unless there is concurvity. In that case, the solution depends upon the initial conditions.

Concurvity occurs when the $\{\boldsymbol{x}_i\}$ values lie upon a smooth manifold in $\mathbb{R}^p$. In our context, a manifold is smooth if the smoother used in backfitting can interpolate all the $\{\boldsymbol{x}_i\}$ perfectly.

This is exactly analogous to the non-uniqueness of regression solutions when the $\boldsymbol{X}$ matrix is not full-rank.

Let $\boldsymbol{P}$ be an operator on p-tuples of functions $\boldsymbol{g} = (g_1, \ldots, g_p)$ and let $\boldsymbol{Q}$ be an operator on a function h. Then the **concurvity space** of

$$\boldsymbol{P}\boldsymbol{g} = \boldsymbol{Q}h$$

is the set of additive functions $g(\boldsymbol{x}) = \sum g_j(x_j)$ such that $\boldsymbol{P}\boldsymbol{g} = \mathbf{0}$. That is,

$$g_j(x_j) + \mathbb{E}[\sum_{k \neq j} g_k(x_k) \,|\, x_j] = 0.$$

We shall now consider several extensions of the general idea in additive modeling.

3.2 Generalized Additive Model

The generalized additive model assumes that the response variable $\boldsymbol{Y}$ comes from an exponential family (e.g., binomial or Poisson). This is like analysis with the generalized linear model of McCullagh and Nelder (1989; *Generalized Additive Models*, 2nd ed., Chapman and Hall).

Recall that in generalized linear models the explanatory values are related to the response through a link function g. If

$$\mu = \mathbb{E}[Y \mid \boldsymbol{X}], \ \textbf{then} \ g(\mu) = \alpha + \boldsymbol{x}'\boldsymbol{\beta}.$$

For example, if Y is Bernoulli, then $\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = p(\boldsymbol{x}) = \mathbb{P}[Y = 1 \mid \boldsymbol{x}]$. Then

$$g(p(\boldsymbol{x}) = \textbf{logit}(p(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x})}{1 - p(\boldsymbol{x})}$$

which yields logistic regression.

The generalized additive model expresses the link function as an additive, rather than linear, function of $\boldsymbol{x}$:

$$g(\boldsymbol{\mu}) = \beta_0 + \sum_{j=1}^{p} f_j(x_j).$$

As before, the link function is chosen by the user based on domain knowledge. Only the relation to the explanatory variables is modeled.

Thus an additive version of logistic regression is

$$\mathbf{logit}(p(\boldsymbol{x})) = \beta_0 + \sum_{j=1}^{p} f_j(x_j).$$

Generalized linear models are fit by iterative scoring, a form of iteratively reweighted least squares. The generalized additive model modifies backfitting in a similar way (cf. Hastie and Tibshirani; 1990, *Generalized Additive Models*, chap. 6).

3.3 Projection Pursuit Regression

A different extension of the additive model is Projection Pursuit Regression (PPR). This treats models of the form:

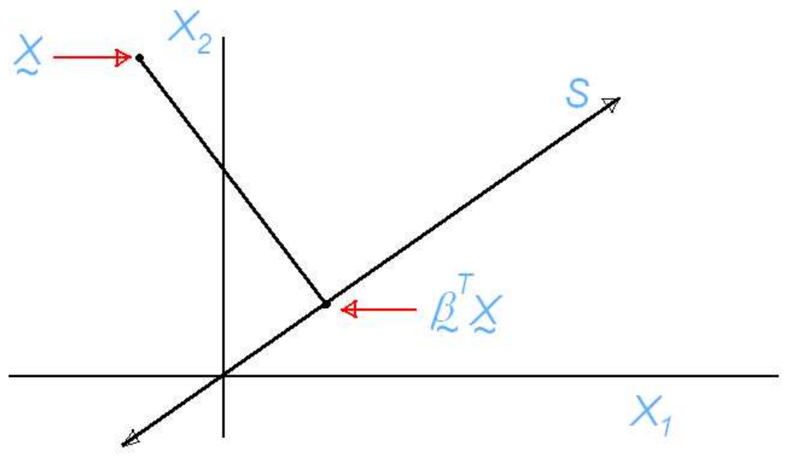
$$Y = \beta_0 + \sum_{j=1}^{r} f_j(\boldsymbol{\beta}'\boldsymbol{X}) + \epsilon$$

where r is found from the data by cross-validation, the f_j are backfitting smooths, and the $\boldsymbol{\beta}_j$ are predictive linear combinations of explanatory variables.

Friedman and Stuetzle (1981; *Journal of the American Statistical Association*, **76**, 817-823) based PPR on exploratory data analysis strategies used to rotate point clouds in order to visualize interesting structure.

PPR tends to work when the explanatory variables are commensurate; e.g., in predicting lifespan, similar biometric measurements might be bundled into one linear combination, and education measurements might form another.

Picking out a linear combination is equivalent to choosing a one-dimensional projection of $\boldsymbol{X}$. For example, take $\boldsymbol{\beta}' = (1,1)$ and $\boldsymbol{x} \in \mathbb{R}^2$. The $\boldsymbol{\beta}'\boldsymbol{x}$ is the projection of $\boldsymbol{x}$ onto the subspace $\mathcal{S} = \{\boldsymbol{x} : x_1 = x_2\}$.



If $r = 1$, then the fitted PPR surface is constant along lines orthogonal to $\mathcal{S}$. If f_1 were the sine function, then the surface would look like corrugated aluminium, but oriented so that the ridges were perpendicular to $\mathcal{S}$.

When $r > 1$ the surface is hard to visualize, especially since the $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_r$ need not be mutually orthogonal. As $r \to \infty$, the PPR fit is a consistent estimator of smooth surfaces (Chen, 1991; *Annals of Statistics*, **19**, 142-157).

The PPR algorithm alternately applies backfitting (to estimate the f_j) and Gauss-Newton search (to estimate the β_j). It seeks $\{\hat{f}_j\}$ and $\{\hat{\boldsymbol{\beta}}_j\}$ that minimize:

$$\sum_{i=1}^{n} [Y_i - \sum_{j=1}^{r} \hat{f}_j(\hat{\boldsymbol{\beta}}_j' \boldsymbol{x}_j)]^2.$$

The algorithm assumes a fixed r, but this can be relaxed by doing univariate search on r.

The Gauss-Newton step starts with initial guesses for $\{\hat{f}_j\}$ and $\{\hat{\boldsymbol{\beta}}_j\}$ and uses the multivariate first-order Taylor expansion around the initial $\{\hat{\boldsymbol{\beta}}_j\}$ to improve the estimated projection directions.

The PPR algorithm works as follows:

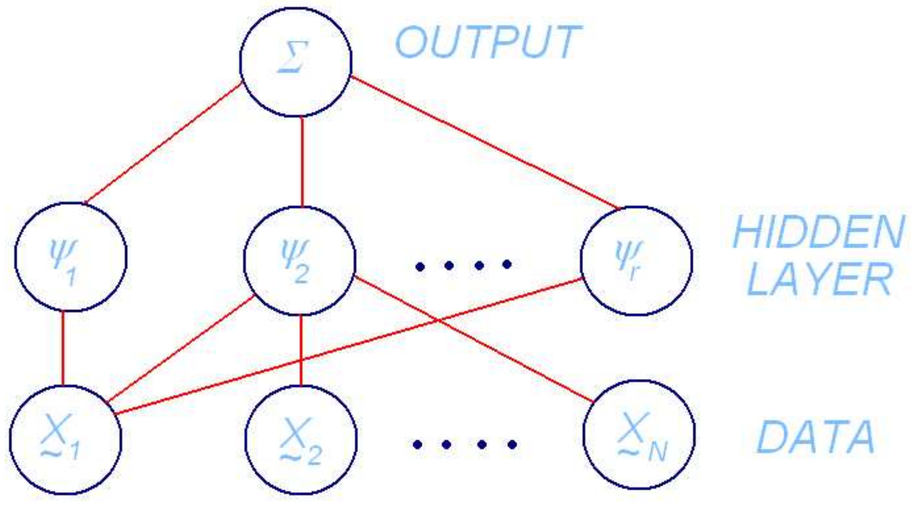
```
1. Fix $r$.
1. Initialize. Get initial estimates for $\{\hat{f}_j\}$ and $\{\hat{\boldsymbol{\beta}}_j\}$.
2. Loop.
      For $j = 1, \ldots, r$ do:
      $\hat{f}_j(\hat{\boldsymbol{\beta}}_j'\boldsymbol{x}) = S(\boldsymbol{Y} - \sum_{j \neq k} \hat{f}_k(\hat{\boldsymbol{\beta}}_k'\boldsymbol{x}) \,|\, \boldsymbol{\beta}_j)$
      End For.
      Find new $\hat{\boldsymbol{\beta}}_j$ by Gauss-Newton
      If the maximum change in $\{\hat{\boldsymbol{\beta}}_j\}$ is less than some threshold, exit.
      End Loop.
```

This converges uniquely under essentially the same conditions as for the AM.

3.4 Neural Networks

A third version of the additive model is **neural networks**. These methods are very close to PPR.

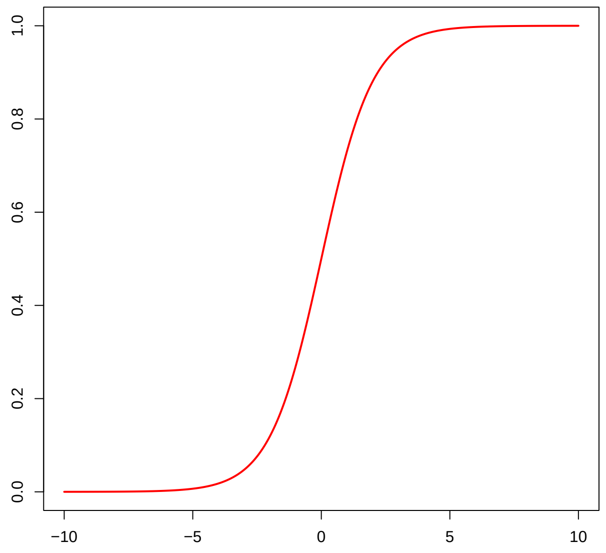
There are many different fiddles on the neural network strategy. We focus on the basic feed-forward network with one hidden layer.



Neural networks fit a model of the form

$$Y = \beta_0 + \sum_{j=1}^{r} \gamma_j \psi(\boldsymbol{\beta}_j' \boldsymbol{x} + \nu_j)$$

where ψ is a sigmoidal (or logistic) function and the other parameters (except r) are estimated from the data.



The only difference between PPR and the neural net is that neural nets assumes that the additive functions have a parametric (logistic) form:

$$\psi(\boldsymbol{x}) = \frac{1}{1 + \exp(\alpha_0 + \boldsymbol{\beta}'\boldsymbol{x})}.$$

The parametric assumption allows neural nets to be trained by backpropagation, an iterative fitting technique. This is very similar to backfitting, but somewhat faster because it does not require smoothing.

Barron (1993; *IEEE Transactions on Information Theory*, **39**, 930-945) showed that neural networks evade the Curse of Dimensionality in specific, rather technical, sense. We sketch his result.

A standard way of assessing the performance of a nonparametric regression procedure is in terms of **Mean Integrated Square Error** (MISE). Let $g(\boldsymbol{x})$ denote the true function and $\hat{g}(\boldsymbol{x})$ denote the estimated function. Then

$$\mathbf{MISE}[\hat{g}] = \mathbb{E}_F \left[\int [\hat{g}(\boldsymbol{x}) - g(\boldsymbol{x})]^2 \, d\boldsymbol{x} \right]$$

where the expectation is taken with respect to the randomness in the data $\{(Y_i, \boldsymbol{X}_i)\}$.

Before Barron's work, it had been thought that the COD implied that for any regression procedure, the MISE had to grow faster than linearly in p, the dimension of the data. Barron showed that neural networks could attain an MISE of order $\mathcal{O}(r^{-1}) + \mathcal{O}(rp/n)\ln n$ where r is the number of hidden nodes.

Recall that $a_n = \mathcal{O}(h(n))$ means there exists c such that for n sufficiently large, $a_n \leq ch(n)$.

Barron's theorem is technical. It applies to the class of functions $g \in \Gamma_c$ on $\mathbb{R}^p$ whose Fourier transforms $\tilde{g}(\omega)$ satisfy

$$\int |\omega| \tilde{g}(\omega)\, d\omega \leq c$$

where the integral is in the complex domain and $|\cdot|$ denotes the complex modulus.

The class Γ_c is **thick**, meaning that it cannot be parameterized by a finite-dimensional parameter. But it excludes important functions such as hyperflats.

The strategy in Barron's proof is:

- Show that for all $g \in \Gamma_c$, there exists a neural net approximation $\hat{g}^*$ such that $\|g - \hat{g}^*\|^2 \leq c^*/n$.
- Show that the MISE in estimating any of the $\hat{g}^*$ functions is bounded.
- Combine these results to obtain a bound on the MISE of a neural net estimate $\hat{g}$ for an arbitrary $g \in \Gamma_c$.

3.5 ACE and AVAS

Alternating Conditional Expectations (ACE) seeks transformations $f_1, \ldots, f_p$ and g of the p explanatory variables and the response variable Y that maximize the correlation between

$$g(Y) \textbf{ and } \sum_{j=1}^{p} f_j(X_j).$$

This is equivalent to minimizing

$$\mathbb{E}[(g(Y) - \sum_{j=1}^{p} f_j(X_j))^2]/\mathbb{E}[g^2(Y)]$$

where the expectations are taken with respect to $\{(Y_i, \boldsymbol{X}_i)\}$.

ACE modifies the additive modeling strategy by

- allowing arbitrary transformations of the response variable,
- using maximum correlation, squared error, for optimization.

ACE was developed by Breiman and Friedman (1985; *Journal of the American Statistical Association*, **80**, 580-619). It resembles canonical correlation.

The ACE algorithm works as follows:

1. Initialize. Set $g(y_i) = (y_i - \bar{y})/s_y$; set $f_j(x_j)$ as the regression of Y on X_j.

2. Backfit. Fit an additive model to $g(y)$ to obtain new functions $f_1(x_1), \ldots, f_p(x_p)$.

3. Compute. Use smoothing to estimate

$$\tilde{g}(y) = \mathbb{E}[\sum_{j=1}^{p} f_j(x_j) \,|\, Y_i = y_i]$$

and standardize a new $g(y)$ as

$$g(y) = \tilde{g}(y) / \sqrt{\text{Var}\,[\tilde{g}(y)]}.$$

(This standardization ensures that the trivial solution $g \equiv 0$ does not arise.)

4. Alternate. Do steps 2 and 3 until $\mathbb{E}[(g(Y) - \sum_{j=1}^{p} f_j(X_j))^2]$ converges.

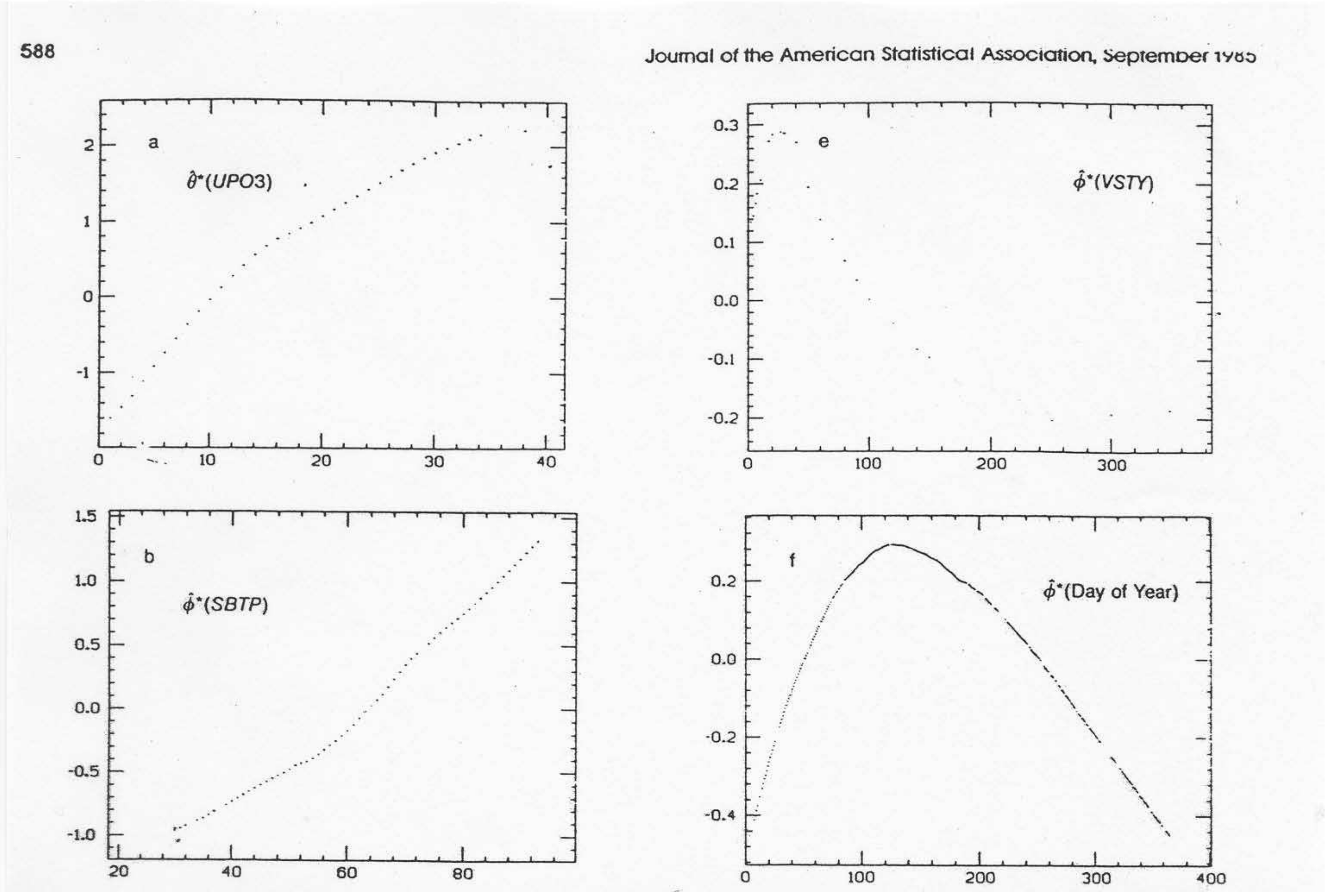
This finds the unique optimal solution, given by the eigenfunctions associated with the largest eigenvalue of a certain operator.

From the standpoint of nonparametric regression, ACE has several undersirable features.

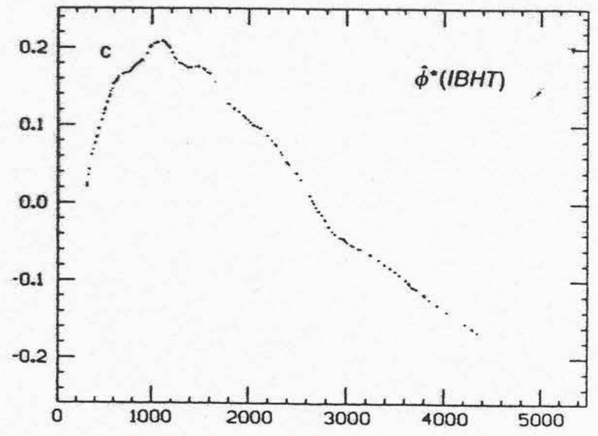
- If $g(Y) = f(X) + \epsilon$, then ACE generally will not find g and f but rather $h \circ g$ and $h \circ f$.
- The solution is sensitive to the marginal distributions of the explanatory variables.
- ACE treats the explanatory and response variables in the same way, but regression should be asymmetric.
- The eigenfunctions for the second-largest eigenvalue can provide better insight on the problem.

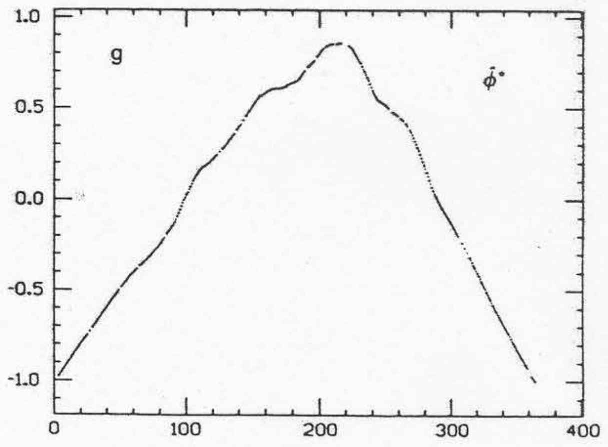
There are a few other pathologies. See the discussion of the Breiman and Friedman article for additional examples and details.

Breiman and Friedman illustrated ACE with an application to Los Angeles ozone data:



588 Journal of the American Statistical Association, September 1985





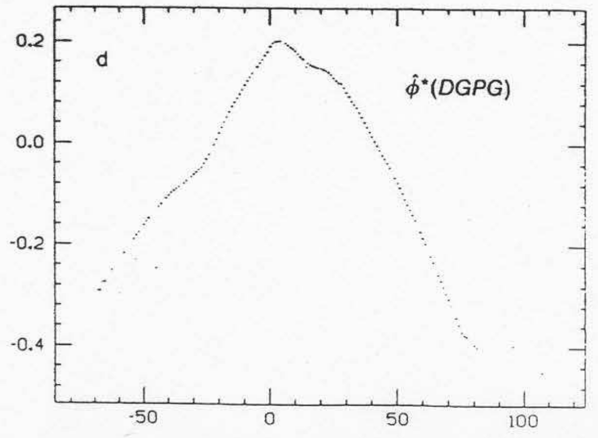


Figure 5. Air Pollution Data: (a) Transformed UPO3 (σ = 1.0); (b) Transformed SBTP (σ = .56); (c) Transformed IBHT (σ = .16); (d) Transformed DGPG (σ = .16); (e) Transformed VSTY (σ = .16); (f) Transformed Day of the Year (σ = .23); (g) Day of the Year as a Single Independent Variable.

Additivity and Variance Stabilization (AVAS) is a modification of ACE that removes most of the undesirable features. It was developed by Tibshirani (1988; *Journal of the American Statistical Association*, **83**, 394-405).

Heteroscedasticity is a common problem in regression and lies at the root of ACE's difficulties.

It is known that if a family of distributions for Z has mean μ and variance $V(\mu)$, then the asymptotic variance stabilizing transformation for Z is

$$h(t) = \int_0^t V(s)^{-1/2}\, ds.$$

The AVAS algorithm is like the ACE algorithm except that in step 3 it applies the estimated variance stabilizing transformation to $\tilde{g}(Y)$ before standardization.

3.6 Recursive Partitioning Regression

Partition methods are designed to handle surfaces with significant interaction structure. The most famous of these methods is CART, for Classification and Regression Trees (Breiman, Friedman, Olshen, and Stone; 1984, Wadsworth).

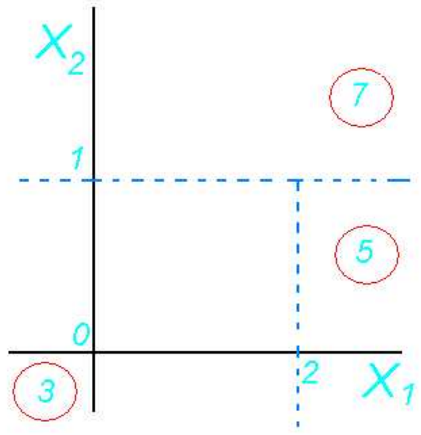
In regression, CART acts as a smart bin-smoother that performs automatic variable selection. Formally, it fits the model

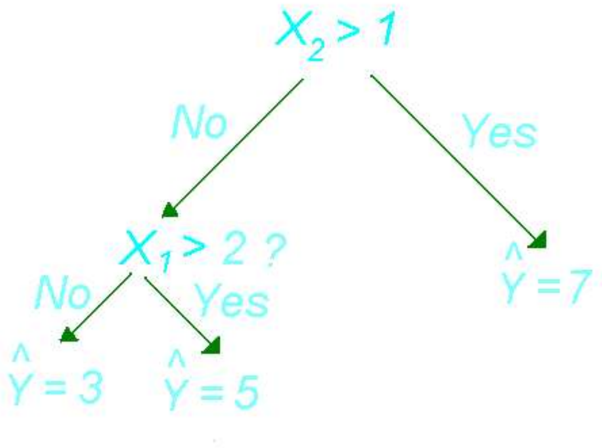
$$Y = \sum_{j=1}^{r} \beta_j I(\boldsymbol{x} \in R_j) + \epsilon$$

where the regions R_j and the coefficients β_j are estimated from the data. Usually the R_j are disjoint and the β_j is the average of the Y values in R_j.

The CART model produces a decision tree that is helpful in interpreting the results, and this is one of the keys to its enduring popularity.

The following partition and tree are equivalent:





Note that CART focuses on a different kind of interaction than the usual product term popular in multiple regression; it looks for thresholds. This leads to results that are not smooth, which some feel is a drawback in regression.

The CART algorithm has three parts:

1. A way to select a split at each intermediate node.
2. A rule for declaring a node to be terminal.
3. A rule for estimating Y at a terminal node.

The third part is easy—just use the sample average of the cases at that terminal node.

The first part is also easy—split on the value x_j^* which most reduces

$$SS_{\textbf{error}} = \sum_{i=1}^{n} (y_i - \hat{f}_c(\boldsymbol{x}_i))^2$$

Where $\hat{f}_c$ is the predicted value from the current tree.

The second part is the hard one. One must grow an overly complicated tree, and then use a pruning rule and cross-validation to find a tree with good predictive accuracy. This entails a complexity penalty.

The main problems with CART are:

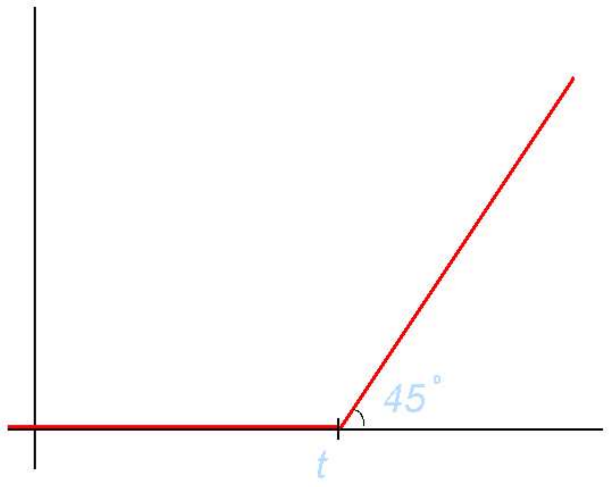
- discontinuous boundaries;
- it is difficult to approximate functions that are linear or additive in a small number of variables;
- it is usually not competitive in low dimensions;
- one cannot tell when a complex CART model is close to a simple model.

The Boston housing data gives an example of the latter situation.

3.7 MARS

Multivariate Adaptive Regression Splines (MARS) improves on CART by marrying it to PPR. It uses multivariate splines to let the data find flexible partitions or $\mathbb{R}^p$. And it incorporates PPR by letting the orientation of the region be non-parallel to the natural axes.

The basic building block is a "hockeystick" (first-order truncated basis) function $(x-t)^+$, which looks like:



This is a special kind of tensor spline. It has a knot at t.

The fitted model has the form

$$Y = \sum_{j=1}^{r} \beta_j B_m(\boldsymbol{x}) + \epsilon$$

where

$$B_m(\boldsymbol{x}) = \prod_{k \in \mathcal{K}} [s_{km}(x_{km} - t_{km})]^+$$

for $s_k m = \pm 1$ and $\mathcal{K}$ is a subset of the explanatory variables. Thus B_m is a product of hockeysticks, so $\hat{f}$ is continuous. The regions are determined by the knots $\{t_{km}\}$.

The MARS algorithms starts with $B_1(\boldsymbol{x}) = 1$ and constructs new terms until there are too many, as measured by generalized cross-validation.

Empirically, Friedman found that each term fit in a MARS model costs between 2 and 4 degrees of freedom. This reflects the variable selection involved in the fitting, but not smoothing.

For each basis function B_m:

1. For all variables not in B_m,
   - **A.** try putting $\pm$ hockeysticks at every observation (candidate knot) in the non-zero region of B_m;
   - **B.** select the best pair of hockeysticks according to an estimate of lack-of-fit.
2. Make two new basis functions from the product of B_m and the chosen pair of hockeystick functions.

After building too many basis functions, prune back as in CART using cross-validation.

Hockeysticks enter in pairs so that MARS is equivariant to sign changes.

Each B_m contains at most one term from each explanatory variable.

MARS is sort of interpretable, via an ANOVAesque decomposition:

$$\hat{f}(\boldsymbol{x}) = \beta_0 + \sum_{j \in \mathcal{J}} f_j(x_j) + \sum_{(j,k) \in \mathcal{K}} f_{jk}(x_j, x_k) + \ldots$$

where β_0 is the coefficient of the B_1 basis function, the first sum is over those basis functions that involve a single explanatory variable, the second sum is over those basis functions that involve exactly two explanatory variables, and so forth.

These terms can be thought of as the grand mean, the main effects, the two-way interactions, etc. One can plot these functions to gain insight.

MARS uses a tensor product basis (hence no explanatory variable appears twice in any B_m). Additive effects are captured by splitting B_1 on several variables. Nonlinear effects are captured by splitting B_1 with the same variable more than once.

MARS and CART are available commercially from Salford Systems, Inc., and versions of them are available in S^+ and R.

To get an integrated suite of code that that performs ACE, AVAS, MARS, PPR, neural nets (the CASCOR version), recursive partitioning, and Loess, one can download the DRAT package from

www.cs.cmu.edu/∼bobski/software/software.html

There are many kinds of neural network code. Users should be careful in using it—there are many possible twiddles and performance may vary. CASCOR is due to Fahlman and Lebiere (1990; *Advances in Neural Information Processing Systems 2*, 524-532, Morgan Kaufmann), and has the convenience of adaptively choosing the number of hidden nodes.