Chapter 10

Additive Models, GAM, and Neural Networks

In observational studies we usually have observed predictors or covariates $X_1, X_2, \ldots, X_p$ and a response variable Y, not just on X. A scientist is interested in the relation between the covariates and the response, a statistician summarizes the relationship with

$$E(Y|X_1, \ldots, X_p) = f(X_1, \ldots, X_n) \tag{10.1}$$

Knowing the above expectation helps us

- understand the process producing Y
- assess the relative contribution of each of the predictors

- predict the Y for some set of values $X_1, \ldots, X_n$.

One example is the air pollution and mortality data. The response variable Y is daily mortality counts. Covariates that are measured are daily measurements of particulate air pollution X_1, temperature X_2, humidity X_3, and other pollutants X_4,...,X_p.

Note: In this particular example we can consider the past as covariates. The methods we talk about here are more appropriate for data for which this doesn't happen.

In this section we will be looking at a diabetes data set which comes from a study of the factors affecting patterns in insulin-dependent diabetes mellitus in children. The objective was to investigate the dependence of the level of serum C-peptide on various other factors in order to understand the patterns of residual insulin secretion. The response measurements Y is the logarithm of C-peptide concentration (mol/ml) at diagnosis, and the predictor measurements are age and base deficit, a measurement of acidity.

A model that has the form of (10.1) and is often used is

$$Y = f(X_1, \ldots, X_n) + \varepsilon \tag{10.2}$$

with ε a random error with mean 0 and variance σ^2 independent from all the $X_1, \ldots, X_p$.

Usually we make a further assumption, that ε is normally distributed. Now we are not only saying something about the relationship between the response and covariates but also about the distribution of Y.

Given some data, "estimating" $f(x_1, \ldots, x_n)$ can be "hard". Statisticians like to make it easy assuming a linear regression model

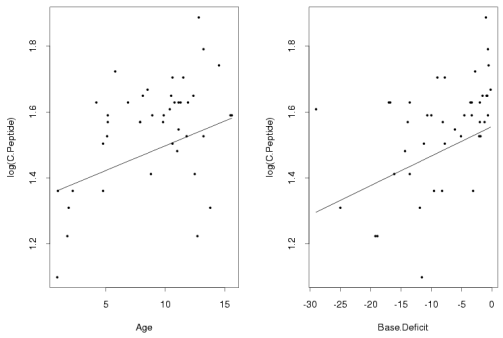
$$f(X_1, \ldots, X_n) = \alpha + \beta_1 X_1 + \ldots + \beta_p X_p$$

This is useful because it

- is very simple
- summarizes the contribution of each predictor with one coefficient
- provides an easy way to predict Y for a set of covariates $X_1, \ldots, X_n$.

It is not common to have an observational study with continuous predictors where there is "science" justifying this model. In many situations it is more useful to let the data "say" what the regression function is like. We may want to stay away from linear regression because it forces linearity and we may never see what $f(x_1, \ldots, x_n)$ is really like.

In the diabetes example we get the following result:



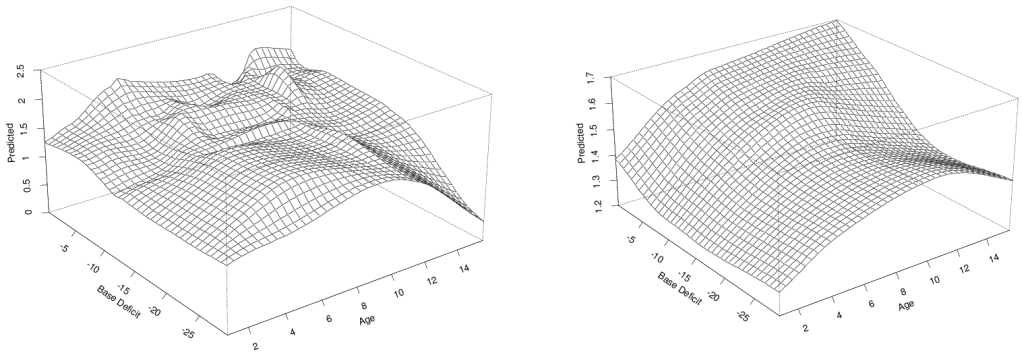
So does the data agree with the fits? Let's see if a "smoothed" version of the data agrees with this result.

But how do we smooth? Some of the smoothing procedures we have discussed may be generalized to cases where we have multiple covariates.

There are ways to define splines so that $g : I \subset \mathbb{R}^p \to \mathbb{R}$. We need to define knots in $I \subset \mathbb{R}^p$ and restrictions on the multiple partial derivative which is difficult but can be done.

It is much easier to generalize loess. The only difference is that there are many more polynomials to choose from: $\beta_0, \beta_0 + \beta_1 x, \beta_0 + \beta_1 x + \beta_2 y, \beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy, \beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy + \beta_4 x^2$, etc...

This is what we get when we fit local planes and use 15% and 66% of the data.



However, when the number of covariates is larger than 2 looking at small "balls" around the target points becomes difficult.

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want to fit loess using $\lambda \times 100\%$ of the data in the local fitting. If we have p covariates and we are forming $p - dimensional$ cubes, then each side of the cube must have size l determined by $l^p = \lambda$. If $\lambda = .10$ (so its supposed to be very local) and $p = 10$ then $l = .1^{1/10} = .8$. So it really isn't local! This is known as *the curse of dimensionality*.

10.1 Projection Pursuit

One solution is projection-pursuit (originally proposed in th 70s). It assumes a model of the form

$$f(X_1, \ldots, X_n) = \sum_{j=1}^{p} f_j\{\boldsymbol{\alpha}_j' \mathbf{X}\}$$

where $\boldsymbol{\alpha}_j' \mathbf{X}$ denotes a one dimensional projection of the vector $(X_1, \ldots, X_p)'$ and f_j is an arbitrary function of this projection.

The model builds up the regression surface by estimating these univariate regressions along carefully chosen projections defined by the $\boldsymbol{\alpha}_k$. Thus for $K = 1$ and $p = 2$ the regression surface looks like a corrugated sheet and is constant in the directions orthogonal to $\alpha_{\mathbf{k}}$.

Iterative algorithms are used. Given the αs we try to find the best fs. Then, given the fs we search for the best directions α.

We will describe two special cases of projection pursuit: GAM and Neural Net-

works.

10.2 Additive Models

Additive models are specific application of projection pursuit. They are more useful in scientific applications.

In additive models we assume that the response is linear in the predictors effects and that there is an additive error. This allows us to study the effect of each predictor separately. The model is like (10.2) with

$$f(X_1, \ldots, X_p) = \sum_{j=1}^{p} f_j(X_j).$$

Notice that this is projection pursuit with the projection

$$\boldsymbol{\alpha}_j' \mathbf{X} = X_j.$$

The assumption made here is not as strong as in linear regression, but its still quite strong. It's saying that the effect of each covariate is additive. In practice this may not make sense.

Example: In the diabetes example consider an additive model that models log(C-peptide) in terms of age X_1 and base deficit X_2. The additive model assumes that for two different ages x_1 and x_1' the conditional expectation of Y (seen as a random variable depending on base deficit):

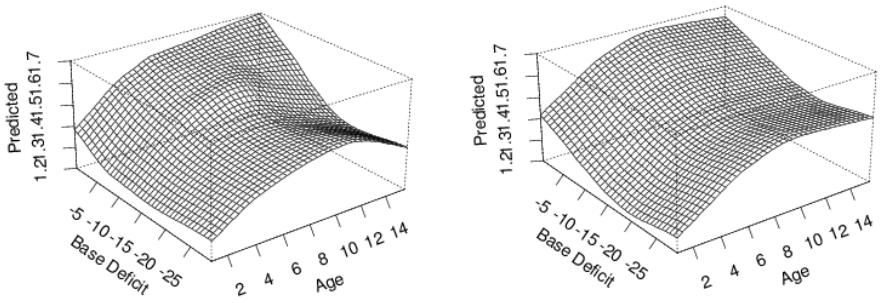
$$\mathrm{E}(Y|X_1 = x_1, X_2) = f_1(x_1) + f_2(X_2)$$

and

$$\mathrm{E}(Y|X_1 = x_1', X_2) = f_1(x_1') + f_2(X_2).$$

This say that the way C-peptide depends on base deficit only varies by a constant for different ages. It is not easy to disregard the possibility that this dependence changes. For example, at older ages the effect of high base deficit can be dramatically bigger. However, in practice we have too make assumptions like these in order to get some kind of useful description of the data.

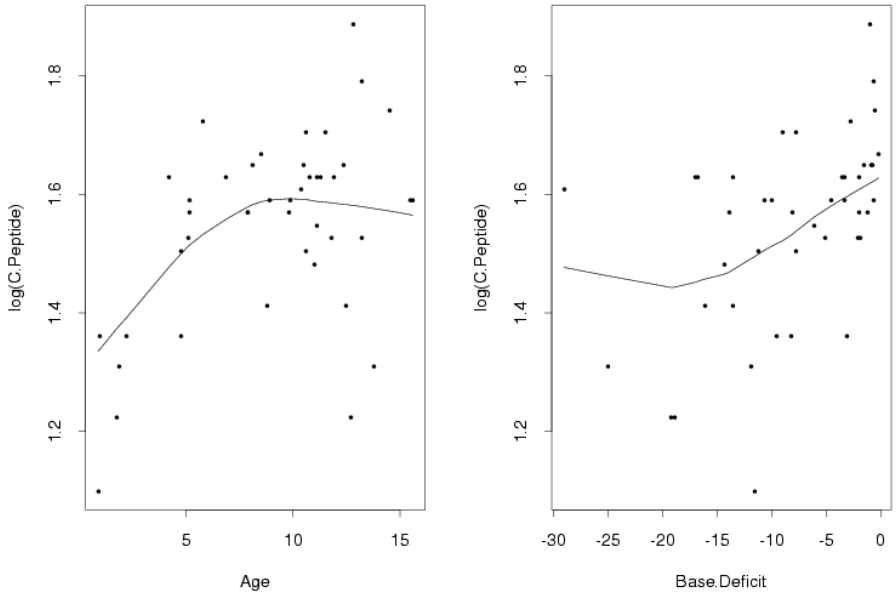
Comparing the non-additive smooth (seen above) and the additive model smooth shows that it is not completely crazy to assume additivity.



Notice that in the first plots the curves defined for the different ages are different. In the second plot they are all the same.

How did we create this last plot? How did we fit the additive surface. We need to estimate f_1 and f_2. We will see this in the next section.

Notice that one of the advantages of additive model is that no matter the dimension of the covariates we know what the surface $f(X_1, \ldots, X_p)$ is like by drawing each $f_j(X_j)$ separately.



10.2.1 Fitting Additive Models: The Back-fitting Algorithm

Conditional expectations provide a simple intuitive motivation for the back-fitting algorithm.

If the additive model is correct then for any k

$$\mathrm{E}\left(Y - \alpha - \sum_{j \neq k} f_j(X_j) \,\middle|\, X_k \right) = f_k(X_k)$$

This suggest an iterative algorithm for computing all the f_j.

Why? Let's say we have estimates $\hat{f}_1, \ldots, \hat{f}_{p-1}$ and we think they are "good" estimates in the sense that $\mathrm{E}\{f_j(X_j) - f_j(X_j)\}$ is "close to 0". Then we have that

$$\mathrm{E}\left(Y - \hat{\alpha} - \sum_{j=1}^{p-1} \hat{f}_j(X_j) \,\middle|\, X_p \right) \approx f_p(X_p).$$

This means that the partial residuals $\hat{\epsilon} = Y - \hat{\alpha} - \sum_{j=1}^{p-1} \hat{f}_j(X_j)$

$$\hat{\epsilon}_i \approx f_p(X_{ip}) + \delta_i$$

with the δ_i approximately IID mean 0 independent of the X_p's. We have already discussed various "smoothing" techniques for estimating f_p in a model as the above.

Once we choose what type of smoothing technique we are using for each covariate, say its defined by $S_j(\cdot)$, we obtain an estimate for our additive model following these steps

1. Define $\mathbf{f}_j = \{f_j(x_{1j}), \ldots, f_j(x_{nj})\}'$ for all j.
2. Initialize: $\alpha^{(0)} = \text{ave}(y_i)$, $\mathbf{f}_j^{(0)} =$ linear estimate.
3. Cycle over $j = 1, \ldots, p$

$$\mathbf{f}_j^{(1)} = S_j \left(\mathbf{y} - \alpha^{(0)} - \sum_{k \neq j} \mathbf{f}_k^{(0)} \middle| \mathbf{x}_j \right)$$

4. Continue previous step until functions "don't change", for example until

$$\max_j \left|\left| \mathbf{f}_j^{(n)} - \mathbf{f}_j^{(n-1)} \right|\right| < \delta$$

   with δ is the smallest number recognized by your computer. In my computer using S-Plus its:

```
.Machine$double.eps = 2.220446e-16
```

Things to think about:

Why is this algorithm valid? Is it the solution to some minimization criterion? Its not MLE or LS.

10.2.2 Justifying the back-fitting algorithm

The back-fitting algorithm seems to make sense. We can say that we have given an intuitive justification.

However statisticians usually like to have more than this. In most cases we can find a "rigorous" justification. In many cases the assumptions made for the "rigorous" justifications too work are carefully chosen so that we get the answer we want, in this case that the back-fitting algorithm "converges" to the "correct" answer.

In the GAM book, H&T find three ways to justify it: Finding projections in L^2 function spaces, minimizing certain criterion with solutions from reproducing-kernel Hilbert spaces, and as the solution to penalized least squares. We will look at this last one.

We extend the idea of penalized least squares by considering the following criterion

$$\sum_{i=1}^{n} \left\{ y_i - \sum_{j=1}^{p} f_j(x_{ij}) \right\}^2 + \sum_{j=1}^{p} \lambda_j \int \{f_j''(t)\}^2 \, dt$$

over all p-tuples of functions $(f_1, \ldots, f_p)$ that are twice differentiable.

As before we can show that the solution to this problem is a p-tuple of cubic splines with knots "at the data", thus we may rewrite the criterion as

$$\left(\mathbf{y} - \sum_{j=1}^{p} \mathbf{f}_j \right)' \left(\mathbf{y} - \sum_{j=1}^{p} \mathbf{f}_j \right) + \sum_{j=1}^{p} \lambda_j \mathbf{f}_j \mathbf{K}_j \mathbf{f}_j$$

where the $\mathbf{K}_j$s are penalty matrices for each predictor defined analogously to the $\mathbf{K}$ of section 3.3.

If we differentiate the above equation with respect to the function $\mathbf{f}_j$ we obtain $-2(\mathbf{y} - \sum_k \mathbf{f}_k) + 2\lambda_j \mathbf{K}_j \mathbf{f}_j = 0$. The $\hat{\mathbf{f}}_j$'s that solve the above equation must

satisfy:

$$\hat{\mathbf{f}}_j = (\mathbf{I} + \lambda_j \mathbf{K}_j)^{-1} \left(\mathbf{y} - \sum_{k \neq j} \hat{\mathbf{f}}_k \right), j = 1, \ldots, p$$

If we define the smoother operator $\mathbf{S}_j = (\mathbf{I} + \lambda_j \mathbf{K}_j)^{-1}$ we can write out this equation in matrix notation as

$$\begin{pmatrix} \mathbf{I} & \mathbf{S}_1 & \ldots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \ldots & \mathbf{S}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \ldots & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_p \end{pmatrix} = \begin{pmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \vdots \\ \mathbf{S}_p\mathbf{y} \end{pmatrix}$$

One way to solve this equation is to use the Gauss-Seidel algorithm which in turn is equivalent to solving the back-fitting algorithm. See Buja, Hastie & Tibshirani (1989) Ann. Stat. 17, 435–555 for details.

Remember that that for any set of linear smoother

$$\hat{\mathbf{f}}_j = \mathbf{S}_j \mathbf{y}$$

we can argue in reverse that it minimizes some penalized least squares criteria of the form

$$(\mathbf{y} - \sum_j \mathbf{f}_j)'(\mathbf{y} - \sum_j \mathbf{f}_j) + \sum_j \mathbf{f}_j'(\mathbf{S}_j^- - I)\mathbf{f}_j$$

and conclude that it is the solution to some penalized least squared problem.

10.2.3 Standard Error

When using GAM in R we get point-wise standard errors. How are these obtained?

Notice that our estimates $\hat{\mathbf{f}}_j$ are no longer of the form $\mathbf{S}_j\mathbf{y}$ since we have used a complicated back-fitting algorithm. However, at convergence we can express $\hat{\mathbf{f}}_j$ as $\mathbf{R}_j\mathbf{y}$ for some $n \times n$ matrix $\mathbf{R}_j$. In practice this $\mathbf{R}_j$ is obtained from the last calculation of the $\hat{\mathbf{f}}_j$'s but finding a closed form is rarely possible.

Ways of constructing confidence sets is not straight forward, and (to the best of my knowledge) is an open area of research.

10.3 Generalized Additive Models

What happens if the response is not continuous? This the same problem that motivated the extension of linear models to generalized linear models (GLM).

In this Chapter we will discuss two method based on a likelihood approach similar to GLMs.

10.3.1 Generalized Additive Models

We extend additive models to generalized additive models in a similar way to the extension of linear models to generalized linear models.

Say Y has conditional distribution from an exponential family and the conditional mean of the response $E(Y|X_1, \ldots, X_p) = \mu(X_1, \ldots, X_p)$ is related to an additive model through some link functions

$$g\{\mu_i\} = \eta_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij})$$

with μ_i the conditional expectation of Y_i given $x_{i1}, \ldots, x_{ip}$. This motivates the use of the IRLS procedure used for GLMs but incorporating the back-fitting algorithms used for estimation in Additive Models.

As seen for GLM the estimation technique is again motivated by the approximation:

$$g(y_i) \approx g(\mu_i) + (y_i - \mu_i)\frac{\partial \eta_i}{\partial \mu_i}$$

This motivates a weighted regression setting of the form

$$z_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i,\ i = 1, \ldots, n$$

with the εs, the working residuals, independent with $\mathrm{E}(\varepsilon_i) = 0$ and

$$\mathrm{var}(\varepsilon_i) = w_i^{-1} = \left(\frac{\partial \eta_i}{\partial \mu_i}\right)^2 V_i$$

where V_i is the variance of Y_i.

The procedure for estimating the function f_js is called the *local scoring procedure*:

1. Initialize: Find initial values for our estimate:
$$\alpha^{(0)} = g\left(\sum_{i=1}^{n} y_i/n\right); f_1^{(0)} = \ldots, f_p^{(0)} = 0$$
2. Update:
   - Construct an adjusted dependent variable
   $$z_i = \eta_i^{(0)} + (y_i - \mu_i^{(0)})\left(\frac{\partial \eta_i}{\partial \mu_i}\right)_0$$
   with $\eta_i^{(0)} = \alpha^{(0)} + \sum_{j=1}^{p} f_j^{(0)}(x_{ij})$ and $\mu_i^{(0)} = g^{-1}(\eta_i^{(0)})$
   - Construct weights:
   $$w_i = \left(\frac{\partial \mu_i}{\partial \eta_i}\right)_0^2 (V_i^{(0)})^{-1}$$
   - Fit a weighted additive model to z_i, to obtain estimated functions $f_j^{(1)}$, additive predictor $\eta^{(1)}$ and fitted values $\mu_i^{(1)}$.
   Keep in mind what a fit is.... $\hat{\mathbf{f}}$.
   - Compute the convergence criteria
   $$\Delta(\eta^{(1)}, \eta^{(0)}) = \frac{\sum_{j=1}^{p} ||f_j^{(1)} - f_j^{(0)}||}{\sum_{j=1}^{p} ||f_j^{(0)}||}$$

- A natural candidate for $||f||$ is $||\mathbf{f}||$, the length of the vector of evaluations of f at the n sample points.

3. Repeat previous step replacing $\eta^{(0)}$ by $\eta^{(1)}$ until $\Delta(\eta^{(1)}, \eta^{(0)})$ is below some small threshold.

10.3.2 Penalized Likelihood

How do we justify the local scoring algorithm? One way is to minimize a penalized likelihood criterion.

Given a generalized additive model let

$$\eta_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij})$$

and consider the likelihood $l(f_1, \ldots, f_p)$ as a function $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_p)'$.

Consider the following optimization problem: Over p-tuples of functions $f_1, \ldots, f_p$ with continuous first and second derivatives and integrable second derivatives find one that minimizes

$$pl(f_1, \ldots, f_p) = l(\boldsymbol{\eta}; \mathbf{y}) - \frac{1}{2} \sum_{j=1}^{p} \lambda_j \int \{f_j''(x)\}^2 \, dx$$

where $\lambda_j \geq 0, j = 1, \ldots, p$ are smoothing parameters.

Again we can show that the solution is an additive cubic spline with knots at the unique values of the covariates.

In order to find the $\mathbf{f}$s that maximize this penalized likelihood we need some optimization algorithm. We will show that the Newton-Raphson algorithm is equivalent to the local-scoring procedure.

As before we can write the criterion as:

$$pl(\mathbf{f}_1, \ldots, \mathbf{f}_p) = l(\boldsymbol{\eta}, \mathbf{y}) - \frac{1}{2} \sum_{j=1}^{p} \lambda_j \mathbf{f}_j' \mathbf{K}_j \mathbf{f}_j.$$

In order to use Newton-Raphson we let $\mathbf{u} = \partial l / \partial \boldsymbol{\eta}$ and $\mathbf{A} = -\partial^2 l / \partial \boldsymbol{\eta}^2$. The first step is then taking derivatives and solving the score equations:

$$\begin{pmatrix} \mathbf{A} + \lambda_1 \mathbf{K}_1 & \mathbf{A} & \ldots & \mathbf{A} \\ \mathbf{A} & \mathbf{A} + \lambda_2 \mathbf{K}_2 & \ldots & \mathbf{A} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A} & \mathbf{A} & \ldots & \mathbf{A} + \lambda_p \mathbf{K}_p \end{pmatrix} \begin{pmatrix} \mathbf{f}_1^1 - \mathbf{f}_1^0 \\ \mathbf{f}_2^1 - \mathbf{f}_2^0 \\ \vdots \\ \mathbf{f}_p^1 - \mathbf{f}_p^0 \end{pmatrix} = \begin{pmatrix} \mathbf{u} - \lambda_1 \mathbf{K}_1 \mathbf{f}_1^0 \\ \mathbf{u} - \lambda_1 \mathbf{K}_1 \mathbf{f}_2^0 \\ \vdots \\ \mathbf{u} - \lambda_1 \mathbf{K}_1 \mathbf{f}_p^0 \end{pmatrix}$$

where both $\mathbf{A}$ and $\mathbf{u}$ are evaluated at $\boldsymbol{\eta}^0$. In the exponential family with canonical family, the entries in the above matrices are of simple form, for example the matrix $\mathbf{A}$ is diagonal with diagonal elements $a_{ii} = (\partial \mu_i / \partial \eta_i)^2 V_i^{-1}$.

To simplify this further, we let $\mathbf{z} = \boldsymbol{\eta}^0 + \mathbf{A}^{-1}\mathbf{u}$, and $\mathbf{S}_j = (\mathbf{A} + \lambda_j \mathbf{K}_j)^{-1}\mathbf{A}$, a

weighted cubic smoothing-spline operator. Then we can write

$$\begin{pmatrix} \mathbf{I} & \mathbf{S}_1 & \dots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \dots & \mathbf{S}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \dots & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{f}_1^1 \\ \mathbf{f}_2^1 \\ \vdots \\ \mathbf{f}_p^1 \end{pmatrix} = \begin{pmatrix} \mathbf{S}_1\mathbf{z} \\ \mathbf{S}_2\mathbf{z} \\ \vdots \\ \mathbf{S}_p\mathbf{z} \end{pmatrix}$$

Finally we may write this as

$$\begin{pmatrix} \mathbf{f}_1^1 \\ \mathbf{f}_2^1 \\ \vdots \\ \mathbf{f}_p^1 \end{pmatrix} = \begin{pmatrix} \mathbf{S}_1(\mathbf{z} - \sum_{j\neq 1} \mathbf{f}_j^1) \\ \mathbf{S}_2(\mathbf{z} - \sum_{j\neq 2} \mathbf{f}_j^1) \\ \vdots \\ \mathbf{S}_p(\mathbf{z} - \sum_{j\neq p} \mathbf{f}_j^1) \end{pmatrix}$$

Thus the Newton-Raphson updates are an additive model fit; in fact they solve a weighted and penalized quadratic criterion which is the local approximation to the penalized log-likelihood.

Note: any linear smoother can be viewed as the solution to some penalized likelihood. So we can set-up to penalized likelihood criterion so that the solution is what we want it to be.

This algorithm converges with any linear smoother.

10.3.3 Inference

Deviance

The deviance or likelihood-ratio statistic, for a fitted model $\hat{\boldsymbol{\mu}}$ is defined by

$$D(\mathbf{y}; \hat{\boldsymbol{\mu}}) = 2\{l(\boldsymbol{\mu}_{max}; \mathbf{y}) - l(\hat{\boldsymbol{\mu}})\}$$

where $\boldsymbol{\mu}_{max}$ is the parameter value that maximizes $l(\hat{\boldsymbol{\mu}})$ over all $\boldsymbol{\mu}$ (the saturated model). We sometimes unambiguously use $\hat{\boldsymbol{\eta}}$ as the argument of the deviance rather than $\hat{\boldsymbol{\mu}}$.

Remember for GLM if we have two linear models defined by η_1 nested within η_2, then under appropriate regularity conditions, and assuming η_1 is correct, $D(\hat{\eta}_2; \hat{\eta}_1) = D(y; \hat{\eta}_1) - D(y; \hat{\eta}_2)$ has asymptotic χ^2 distribution with degrees of freedom equal to the difference in degrees of freedom of the two models. This result is used extensively in the analysis of deviance tables etc...

For non-parametric we can still compute deviance and it still makes sense to compare the deviance obtained for different models. However, the asymptotic approximations are undeveloped.

H&T present heuristic arguments for the non-parametric case.

Standard errors

Each step of the local scoring algorithm consists of a back-fitting loop applied to the adjusted dependent variables $\mathbf{z}$ with weights $\mathbf{A}$ given by the estimated information matrix. If $\mathbf{R}$ is the weighted additive fit operator, then at convergence

$$\begin{aligned}\hat{\boldsymbol{\eta}} &= \mathbf{R}(\hat{\boldsymbol{\eta}} + \mathbf{A}^{-1}\hat{\boldsymbol{\mu}}) \\ &= \mathbf{R}\mathbf{z},\end{aligned}$$

where $\hat{\mathbf{u}} = \partial l/\partial\hat{\boldsymbol{\eta}}$. The idea is to approximate $\mathbf{z}$ by an asymptotically equivalent quantity $\mathbf{z}_0$. We will not be precise and write $\approx$ meaning asymptotically equivalent.

Expanding $\hat{\mathbf{u}}$ to first order about the true $\boldsymbol{\eta}_0$, we get $\mathbf{z} \approx \mathbf{z}_0 + \mathbf{A}_0^{-1}\mathbf{u}_0$, which has mean $\boldsymbol{\eta}_0$ and variance $\mathbf{A}_0^{-1}\phi \approx \mathbf{A}\phi$.

Remember for additive models we had the fitted predictor $\hat{\boldsymbol{\eta}} = \mathbf{R}\mathbf{y}$ where $\mathbf{y}$ has covariance $\sigma^2\mathbf{I}$. Here $\hat{\boldsymbol{\eta}} = \mathbf{R}\mathbf{z}$, and $\mathbf{z}$ has asymptotic covariance $\mathbf{A}_0^{-1}$. $\mathbf{R}$ is not a linear operator due to its dependence on $\hat{\mu}$ and thus $\mathbf{y}$ through the weights, so we need to use its asymptotic version $\mathbf{R}_0$ as well. We therefore have

$$\mathbf{cov}(\hat{\boldsymbol{\eta}}) \approx \mathbf{R}_0\mathbf{A}_0^{-1}\mathbf{R}_0'\phi \approx \mathbf{R}\mathbf{A}^{-1}\mathbf{R}'\phi$$

Similarly

$$\mathbf{cov}(\hat{\mathbf{f}}_j) \approx \mathbf{R}_j\mathbf{A}^{-1}\mathbf{R}_j'\phi$$

where $\mathbf{R}_j$ is the matrix that produces $\hat{\mathbf{f}}_j$ from z.

Under some regularity conditions we can further show that $\hat{\boldsymbol{\nu}}$ is asymptotically normal, and this permits us to construct confidence intervals.

10.3.4 Degrees of freedom

Previously we described how we defined the degrees of freedom of the residuals as the expected value of the residual sum of squares. The analogous quantity in generalized models is the deviance. We therefore use the expected value of the deviance to define the *relative degrees of freedom*.

We don't know the exact or asymptotic distribution of the deviance so we need some approximation that will permit us to get an approximate expected value.

Using a second order Taylor approximation we have that

$$\mathrm{E}[D(\mathbf{y};\hat{\boldsymbol{\mu}})] \approx \mathrm{E}[(\mathbf{y}-\hat{\boldsymbol{\mu}})'\mathbf{A}^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}})]$$

with $\mathbf{A}$ the Hessian matrix defined above. We now write this in terms of the "linear terms".

$$\mathrm{E}[(\mathbf{y}-\hat{\boldsymbol{\mu}})'\mathbf{A}(\mathbf{y}-\hat{\boldsymbol{\mu}})] \approx (\mathbf{z}-\hat{\boldsymbol{\eta}})'\mathbf{A}(\mathbf{z}-\hat{\boldsymbol{\eta}})$$

and we can show that this implies that if the model is unbiased

$$\mathrm{E}(D) = df\,\phi$$

with

$$df = n - \mathrm{tr}(2\mathbf{R} - \mathbf{R}'\mathbf{A}\mathbf{R}\mathbf{A}^{-1})$$

This gives the degrees of freedom for the whole model not for each smoother. We can obtain the dfs for each smoother by adding them one at a time and obtaining

$$\mathrm{E}[D(\hat{\boldsymbol{\eta}}_2;\hat{\boldsymbol{\eta}}_1)] \approx \mathrm{tr}(2\mathbf{R}_1 - \mathbf{R}_1'\mathbf{A}_1\mathbf{R}_1\mathbf{A}_1^{-1}) - \mathrm{tr}(2\mathbf{R}_2 - \mathbf{R}_2'\mathbf{A}_2\mathbf{R}_2\mathbf{A}_2^{-1})$$

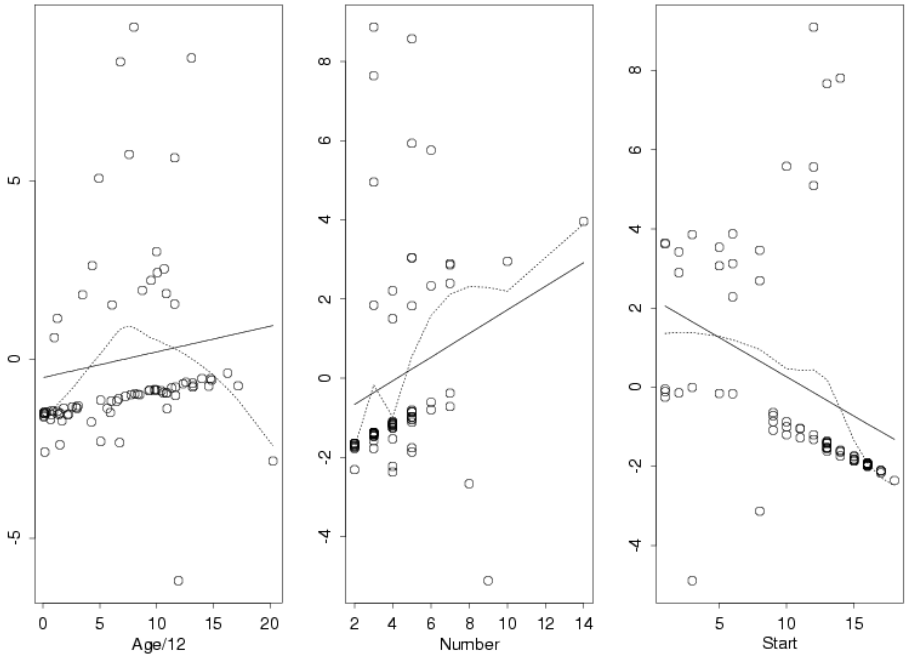
In general, the crude approximation $df_j = \mathrm{tr}(\mathbf{S}_j)$ is used.

10.3.5 An Example

The kyphosis data frame has 81 rows representing data on 81 children who have had corrective spinal surgery. The binary outcome Kyphosis indicates the presence or absence of a postoperative deformity (called Kyphosis). The other three variables are `Age` in months, `Number` of vertebra involved in the operation, and the beginning of the range of vertebrae involved (`Start`).

Using GLM these are the results we obtain

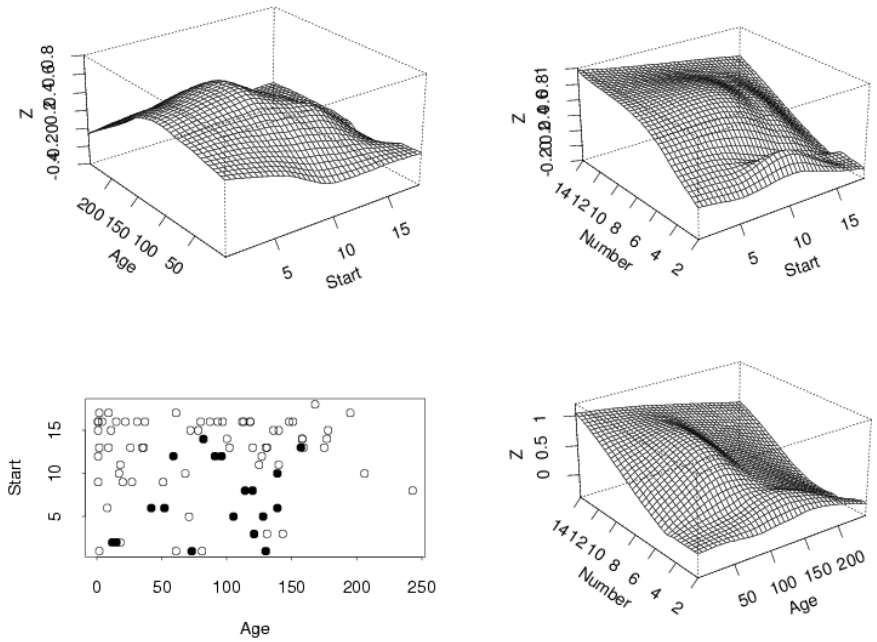
```
                  Value  Std. Error   t value
(Intercept) -1.213433077 1.230078549 -0.986468
        Age  0.005978783 0.005491152  1.088803
     Number  0.298127803 0.176948601  1.684827
      Start -0.198160722 0.065463582 -3.027038

Null Deviance: 86.80381 on 82 degrees of freedom

Residual Deviance: 65.01627 on 79 degrees of freedom
```



The dotted lines are smooths of the residuals. This does not appear to be a very good fit.

We may be able to modify it a bit, by choosing a better model than a sum of lines. We'll use smoothing and GAM to see what "the data says".

Here are some smooth versions of the data:

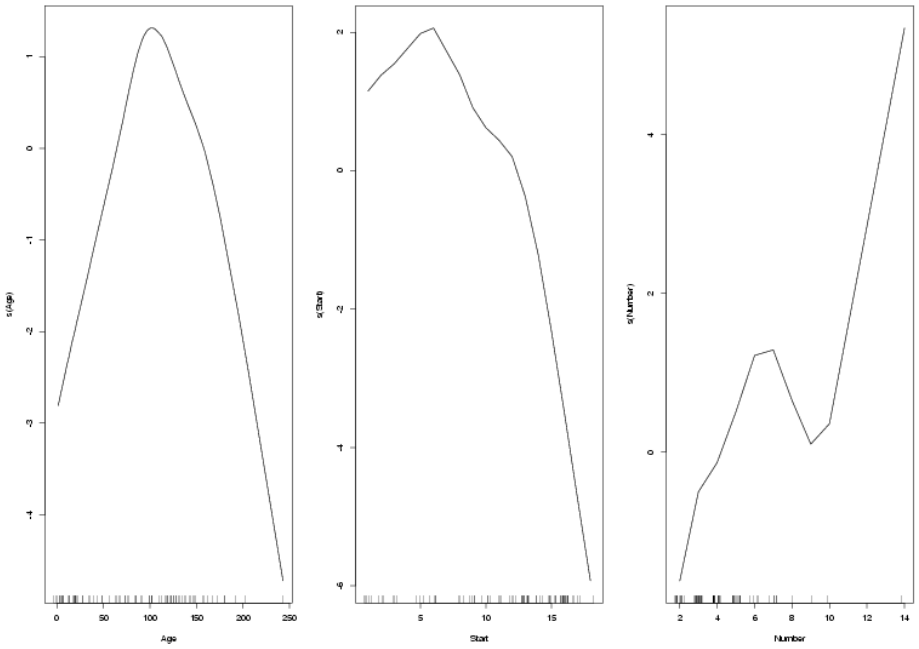


And here are the gam results:

```
Null Deviance: 86.80381 on 82 degrees of freedom

Residual Deviance: 42.74212 on 70.20851 degrees of freedom

Number of Local Scoring Iterations: 7

DF for Terms and Chi-squares for Nonparametric Effects
```

```
            Df Npar Df Npar Chisq    P(Chi)
(Intercept)  1
     s(Age)  1     2.9   6.382833 0.0874180
   s(Start)  1     2.9   5.758407 0.1168511
  s(Number)  1     3.0   4.398065 0.2200849
```

Notice that it is a much better fit and not many more degrees of freedom. Also notice that the tests for linearity are close to "rejection at the 0.05 level".



We can either be happy considering these plots as descriptions of the data, or we can use it to inspire a parametric model:

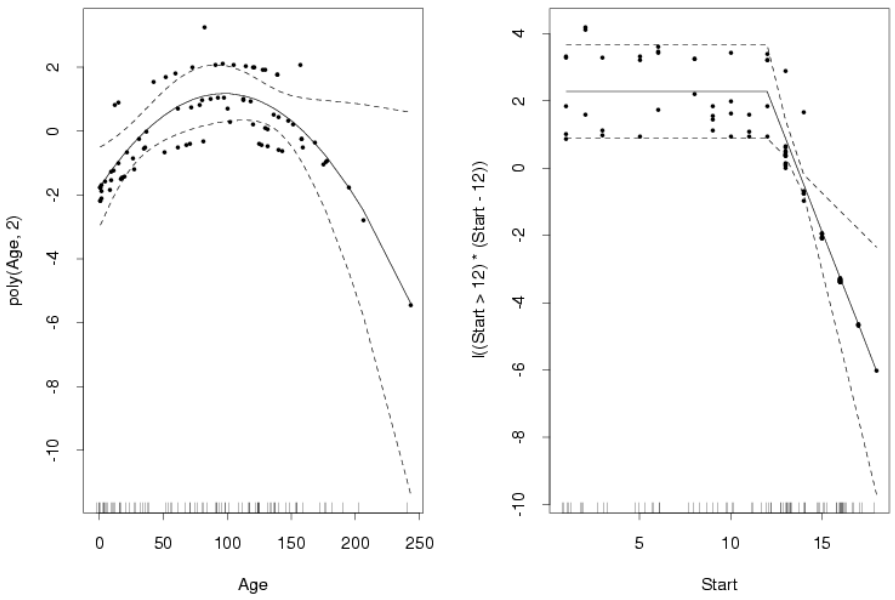
Before doing so, we decide not to include Number because it seems to be associated with "Start" and not adding much to the fit. This and other considerations suggest we not include `Number` . The gam plots suggest the following "parametric" model.

```
glm2 <- glm(Kyphosis~poly(Age,2) + I((Start > 12) * (Start - 12)),
            family=binomial)
```

Here are the results of this fit... much better than the original GLM fit.

```
Coefficients:
                                    Value Std. Error    t value
                  (Intercept)  -0.5421608  0.4172229 -1.2994512
                poly(Age, 2)1   2.3659699  4.1164283  0.5747628
                poly(Age, 2)2 -10.5250479  5.2840926 -1.9918364
I((Start > 12) * (Start - 12))  -1.3840765  0.5145248 -2.6900094

(Dispersion Parameter for Binomial family taken to be 1 )

    Null Deviance: 86.80381 on 82 degrees of freedom

Residual Deviance: 56.07235 on 79 degrees of freedom

Number of Fisher Scoring Iterations: 6
```

Here are the residual plots:



10.3.6 Prediction using GAM

Often we wish to evaluate the fitted model at some new values.

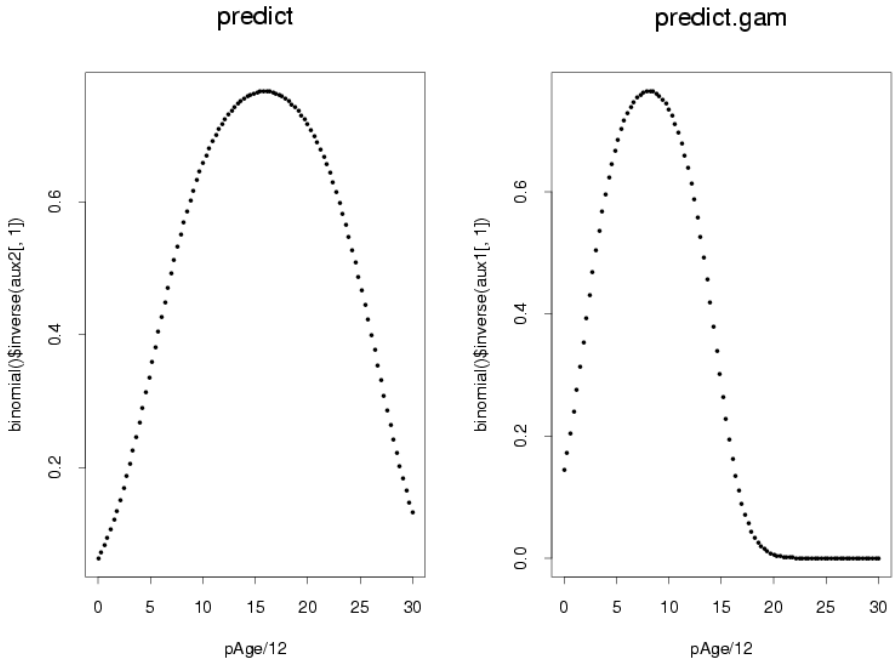
With parametric models this is simple because all we do is form a new design matrix and multiply by the estimated parameters.

Some of the functions used to create design matrices in lm, glm a and gam are

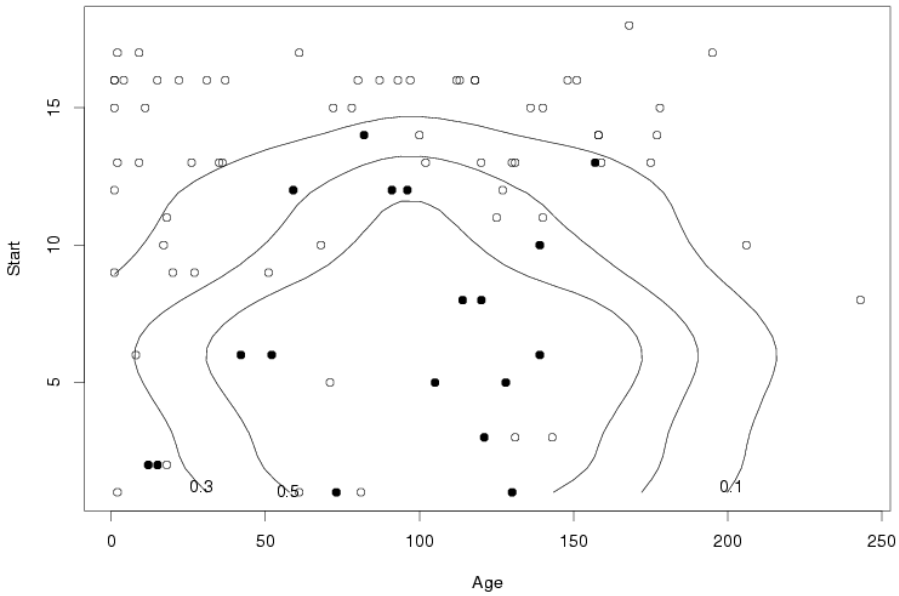
data dependent. For example `bs()`, `poly()`, make some standardization of the covariate before fitting and therefore new covariates would change the meaning of the parameters.

As an example look at what happens when we predict fitted values for new values of AGE in the Kyphosis example using `predict()`.

The solution is to use `predict.gam()` that takes this into account



`predict.gam` is especially useful when we want to make surface plots. For example:


Start
Age
0.3
0.5
0.1
0
50
100
150
200
250
5
10
15

10.3.7 Over-interpreting additive fits

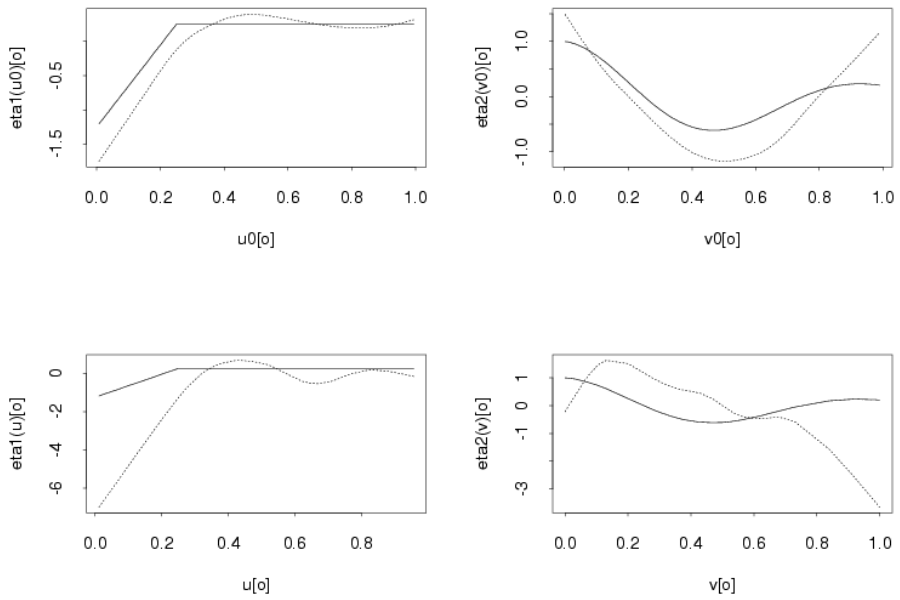
One of the advantages of GAM is their flexibility. However, because of this flexibility we have to be careful not to "over-fit" and interpret the results incorrectly.

Binary data is especially sensitive. We construct a simulated example to see this.

The following figure shows the functional components f_1 and f_2 of a GAM

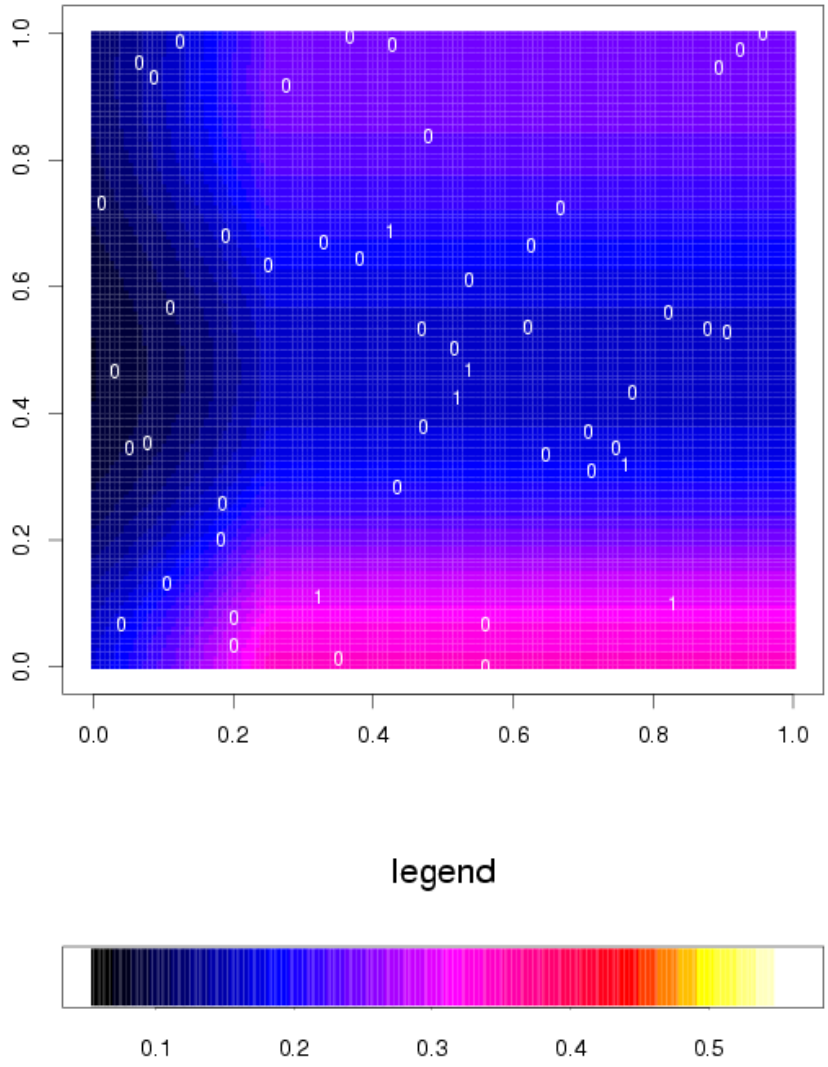
$$\text{logit}\{\Pr(Y = 1|U, V)\} = -1 + f_1(U) + f_2(V)$$

with U and V independent uniform(0,1).



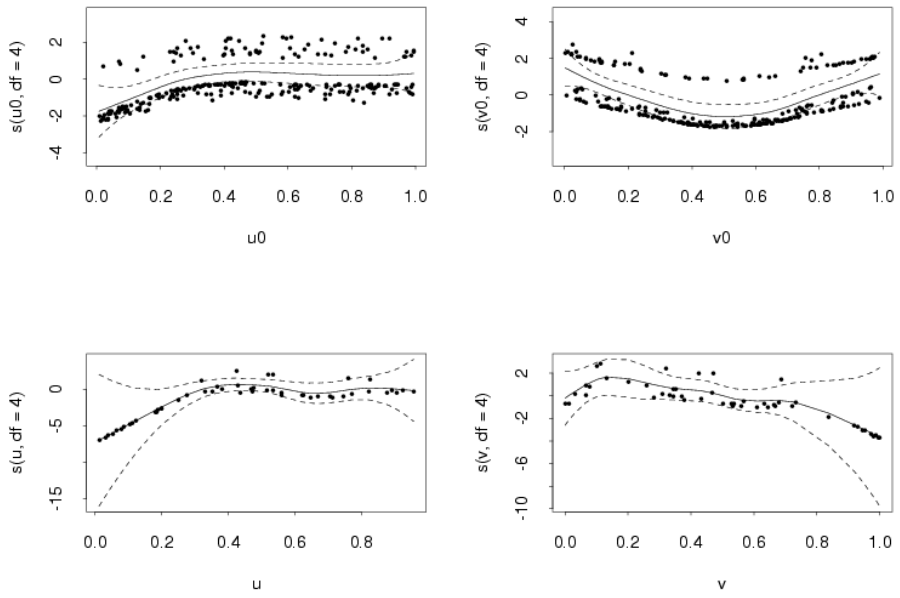
We also show the "smooths" obtained for a data set of 250 observations and a data set of 50 observations. Notice how "bad" the second fit is.

If we make a plot of the mean $\mu(u, v)$ and of it's estimate we see why this happens.


0.0
0.2
0.4
0.6
0.8
1.0
legend
0.1
0.2
0.3
0.4
0.5

We have relatively large neighborhoods of $[0,1] \times [0,1]$ that contain only 1s or only 0s. The estimates in these regions will have linear part close to infinity and minus infinity!

One way to detect this when we don't know "the truth" is to look at the estimates with standard errors and partial residuals. If the partial residuals follow the fit to closely and the standard errors "explode" we know something is wrong.



10.4 Neural Networks

Neural Networks has become a large field of research. In this section we describe the "vanilla" version.

Networks are typically described with diagrams such as the one Rafa drew on the board. The diagram has the outcome Y at the top, the covariates X at the bottom and a set of hidden states Z in the middle.

Although the diagrams make the method seem somewhat magical, once we write down the math, we see that Neural Networks are also a special case of Projection Pursuit.

For a K-class classification problem, there are K units at the top with the k-th unit modeling the probability of seeing a k. There are K target measurement each being coded with $Y_k, k = 1, \ldots, K$ each being 0-1.

Derived features, the Zs, are created from linear combinations of the *inputs*, i.e. the covariates or predictors. More specifically, we write:

$$\begin{aligned} Z_m &= \sigma(\alpha_{0,m} + \alpha' X), m = 1, \ldots, M, \\ T_k &= \beta_{0,k} + \beta_k' Z, k = 1, \ldots, K, \delta_k(X) = g_k(T_1, \ldots, T_K) \end{aligned}$$

where $Z = (Z_1, \ldots, Z_M)$.

σ is referred to as the activation function and is usually chosen to be the sigmoid $\sigma(v) = 1/\{1 + \exp(-v)\}$. It is relatively easy to see that this can be rewritten as

a projection Pursuit Model:

$$\begin{aligned} f_k(\omega' X) &= \beta_m \sigma(\alpha_0 + \alpha'_m X) \\ &= \beta_m \sigma(\alpha_o + ||\alpha_m||(\omega'_m X)) \end{aligned}$$

where ω_m=$\alpha_m/||\alpha_m||$ is the mth unit-vector. Notice this f_k is not as complex as the general function (splines, loess smooths) typically used in projection pursuit. This means, that in general, Neural networks need much larger Ms.

Bibliography

[1] Hastie, T. and Tibshirani, R. (1987), "Generalized additive models: Some applications," *Journal of the American Statistical Association*, 82, 371–386.