# 36-490 Spring 2010: Additive models and cross-validation

**Brian Junker**

**March 22, 2010**

- Cross-Reference to R
- Interpolating Splines and Smoothing Splines
- MSE and the Bias-Variance Tradeoff
- *K*-fold Cross-Validation
- Additive and Generalized Additive Models
- Generalized Additive Models
- Projection Pursuit Regression



## Cross-Reference to R

- Interpolating and smoothing splines: `spline()`, `smooth.spline()`, and kernel regression `ksmooth()`
- Cross-validation using `crossval()` from `library(bootstrap)`
- Generalized additive models
  - `gam()` from `library(gam)` is the classic GAM function. It allow for plotting of residuals etc just like ordinary `lm()`.
  - `gam()` from `library(mgcv)` has a bit more flexibility, and chooses its smoothing parameters by a more recent generalized cross-validation method. But it does not seem to let you plot residuals, etc. for easy model checks (?).
- Projection Pursuit Regression: `ppr()` from `library(MASS)`.

# Interpolating Splines and Smoothing Splines

Splines are smooth functions (possessing some number of continuous derivatives) which are piecewise polynomials of a fixed order.

*Interpolation splines*

**Definition.** Let $x_0 < \cdots < x_K$ and $y_k = g(x_k)\ \forall\ i$. Then $s(x)$ is a *natural interpolating spline* of order $M$ if:

(a) $s(x_k) = g(x_k)\ \forall\ k$;
(b) $s^{(m+1)}(x) \equiv 0$ on each interval $(x_k, x_{k+1})$;
(c) $s^{(m)}(x)$ exists and is continuous on $(x_1, x_n)$, for all $m = 1, \ldots, M-1$;
(d) $s^{(M-1)}(x_1) = s^{(M-1)}(x_n) = 0$.

For example

- The natural interpolating spline of order 1 yields piecewise linear interpolation between the points $(x_i, y_i)$.
- A common choice for interpolating splines are the natural cubic splines.



**Theorem.** There is a unique cubic ($k = 3$) spline function $s(x)$ satisfying (a) through (d) above. Moreover,

- $s(x)$ is a piecewise cubic function:

  $$s(x) = c_{k0} + c_{k1}(x - x_k) + c_{k2}(x - x_k)^2 + c_{k3}(x - x_k)^3 \text{ for } x \in (x_k, x_{k+1});$$

  with the coefficients satisfying a linear system $c = Ay$ implied by (a)–(d) above.

- Equivalently, $s(x)$ can be written as a linear combination of basis functions, e.g. for $K$ knots a basis satisfying (b)–(d) would be

  $$\begin{aligned} N_1(x) &= 1,\ N_2(x) = x, \\ N_{k+2}(x) &= \frac{(x - x_k)_+^3 - (x - x_K)_+^3}{x_K - x_k} - \frac{(x - x_{K-1})_+^3 - (x - x_K)_+^3}{x_K - x_{K-1}} \\ &\quad \forall\ k = 1, \ldots, K-2 \end{aligned}$$

  That is, for any $x$,

  $$s(x) = \sum_{j=1}^{K} \theta_j N_j(x) = \mathcal{N}(x)\theta$$

  where $\mathcal{N}(x) = (N_1(x), \ldots, N_K(x))$ and $\theta = (\theta_1, \ldots, \theta_K)^T$.

We can find the coefficients $\theta = (\theta_1, \ldots, \theta_K)^T$ by solving the linear system

$$\begin{aligned} y_1 &= g(x_1) = s(x_1) = \mathcal{N}(x_1)\theta \\ &\vdots \\ y_n &= g(x_n) = s(x_n) = \mathcal{N}(x_n)\theta \end{aligned}$$

or

$$\vec{y} = \mathcal{N}(\vec{x})\theta$$

where $\vec{y} = (y_1, \ldots, y_n)^T$ and $\mathcal{N}(\vec{x})$ is the $K \times K$ matrix with $(k, j)^{th}$ entry $N_j(x_k)$.

This is a linear regression problem (why??), so it has solution

$$\hat{\theta} = (\mathcal{N}(\vec{x})^T \mathcal{N}(\vec{x}))^{-1} \mathcal{N}(\vec{x})^T \vec{y}$$

Thus, the fitted values $\hat{y}$ may be expressed as

$$\hat{y} = \mathcal{N}(\vec{x})\hat{\theta} = \mathcal{N}(\vec{x})(\mathcal{N}(\vec{x})^T \mathcal{N}(\vec{x}))^{-1} \mathcal{N}(\vec{x})^T \vec{y} = S\vec{y}$$

$s(\vec{x})$ is an example of a *linear smoother*

and $S = \mathcal{N}(\vec{x})(\mathcal{N}(\vec{x})^T \mathcal{N}(\vec{x}))^{-1} \mathcal{N}(\vec{x})^T$ is analogous to the hat matrix $H = X(X^T X)^{-1} X^T$ in linear regression.



One can show that the interpolation spline makes

$$RSS(f) = \sum_{k=1}^{K} (y_k - f(x_k))^2 \equiv 0$$

However, so do a lot of other functions (e.g. a single polynomial of degree $K - 1$).

Moreover, although it provides a perfect fit at the original data points, it is usually too highly variable away from the original data points to be of much use.

If we want a useful interpolating function, we will have to give up exact fit at the original data points in favor of some global structure.

*Smoothing splines*

Smoothing splines impose global structure by *regularization* or *penalization*. Instead of minimizing $RSS(f)$, we now try to minimize

$$PRSS(f) = \sum_{k=1}^{K}(y_k - f(x_k))^2 + \lambda \int [f''(t)]^2 dt = RSS(f) + \lambda J(f)$$

A function $f(x)$ minimizing PRSS provides a compromise between a regression spline fit and a linear fit.

- If $\lambda \approx 0$, the $RSS(f)$ term is emphasized, and the fit will be "close to" the data, jittery (little control over $f''(x)$), and have low bias and high variance.
- If $\lambda \gg 0$, th $J(f)$ term is emphasized and the fit will be smoothed away from the data and towards a linear fit ($f''(x)$ forced to be close to zero).



It can be shown that the $f(x)$ minimizing PRSS is a *natural cubic spline*, as before:

$$s(x) = \sum_{j=1}^{K} \theta_j N_j(x).$$

Plugging this into PRSS, we immediately see

$$PRSS(s) = (\vec{y} - \mathcal{N}(\vec{x})\theta)^T(\vec{y} - \mathcal{N}(\vec{x}))\theta) + \lambda\theta^T\Omega_{\mathcal{N}}\theta$$

where $\mathcal{N} = [N_j(x_k)]_{jk}$ as before, and $\Omega_{\mathcal{N}}$ has $(j,k)^{th}$ entry $\int N_j(t)N_k(t)dt$.

This has solution

$$\hat{\theta} = (\mathcal{N}(\vec{x})^T\mathcal{N}(\vec{x}) + \lambda\Omega_{\mathcal{N}})^{-1}\mathcal{N}(\vec{x})^T\vec{y}$$

Once again, the fitted values $\hat{y}_0$ are

$$\hat{y} = \mathcal{N}(\vec{x})\hat{\theta} = \mathcal{N}(\vec{x})(\mathcal{N}(\vec{x})^T\mathcal{N}(\vec{x}) + \lambda\Omega_{\mathcal{N}})^{-1}\mathcal{N}(\vec{x})^T y = S_\lambda y,$$

so again $s(x)$ is a *linear smoother*.

The matrix $S_\lambda$ is again analogous to the hat matrix $H = X(X^TX)^{-1}X^T$ in linear regression.

Another common linear smoother is the kernel regression estimator

$$f_\lambda(x) = \frac{\sum_{i=0}^n y_i K\left(\frac{x-x_i}{\lambda}\right)}{\sum_{i=0}^n K\left(\frac{x-x_i}{\lambda}\right)}$$

as are other (locally) weighted regression schemes such as `lowess`.

- The big question for smoothing splines, kernel regression, etc., is how to choose the smoothing parameter $\lambda$, to minimize the $RSS$, or equivalently the *mean squared error of prediction (MSE)*.
- A common and effective way to do this is with *cross validation*.

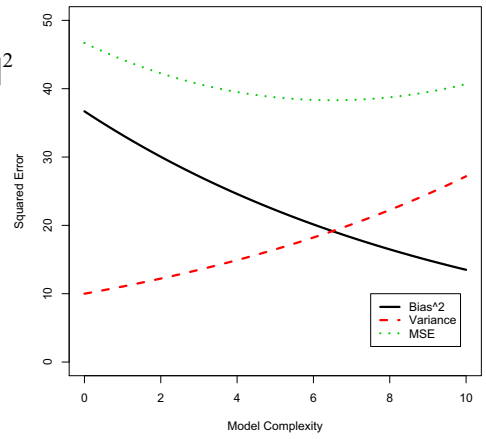


## MSE and the Bias-Variance Tradeoff

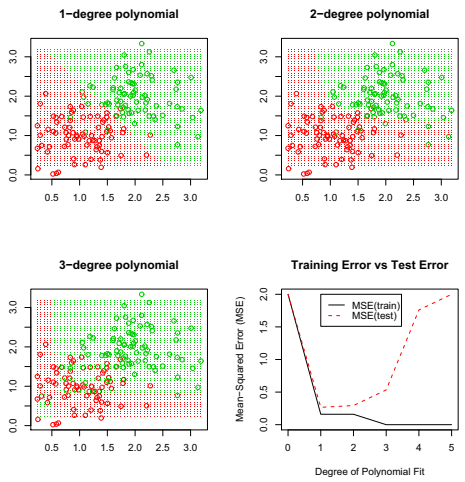
Let $\mathcal{T} = ((x_1, y_1), \ldots, (x_n, y_n))$ be a *training set* on which we train our predictor $\hat{y}_i = \hat{f}(x_i)$. Let $y_0 = f(x_0)$ represent a *new pair*, or *test point*. We can write the mean-squared error in predicting $y_0 \approx \hat{y}_0 = \hat{f}(x_0)$ as

$$\begin{aligned} &MSE(x_0) \\ &= E_{\mathcal{T}}[(f(x_0) - \hat{y}_0)^2] \\ &= E_{\mathcal{T}}[(\hat{y}_0 - E_{\mathcal{T}}(\hat{y}_0))^2] + [E_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\ &= \mathrm{Var}_{\mathcal{T}}(\hat{y}_0) + \mathrm{Bias}^2 \end{aligned}$$

(We try to choose the model—or model complexity—that minimizes MSE.)



## *Why Do We Need Training Set and Test Set?*




- Here we built linear regression functions to classify two kinds of observations (red and green cirlces).
- The figure compares linear, quadratic and cubic linear models.
- You can see that the prediction MSE on the test data is more like the ideal MSE curve, than the MSE on the training data alone.

We can always drive the bias down by making the model more complex.

Estimating MSE on the same data that the model was fitted on tends to under-estimate the variance in fitting new data, so estimates of MSE on the training data are usually too optimistic.



## *K*-fold Cross-Validation

- We can see from the MSE plot on the previous page that the error on the training data is likely to be too optimistic.
- That calculation was based on dividing data into large "training" and "testing" samples. Train on one sample, estimate the error rate on the other.
- A better estimate of MSE would average over several training and test sets. But data is scarce! An efficient way to re-use a single data set for this purpose is *K-fold cross-validation*:
  - Divide up the data into $K$ roughly-equal-sized parts.
  - Let $\hat{f}(x)^{-k}$ be the fitted value (classification, prediction, etc.) for $x$ with the $k^{th}$ part of the data removed, and let $k(i)$ be the part of the data containing $x_i$.
  - Then the $K$-fold cross-validation criterion is

$$CV = \frac{1}{N}\sum_{i=1}^{N} L(y, \hat{f}^{-k(i)}(x_i))$$

where $L(y, \hat{y})$ is some appropriate loss function [e.g. $L(y, \hat{y}) = (y - \hat{y})^2$, if we are interested in MSE].

## *Choosing the number K of cross validation groups*

*CV* can also be interpreted as trying to estimate the true error rate of the optimal $f(x)_{opt}$:

$$Error(f_{opt}(\cdot)) \approx \frac{1}{K}\sum_{k=1}^{K} Error(\hat{f}^{-k}(\cdot)) \approx \frac{1}{N}\sum_{k=1}^{N} L(y, \hat{f}^{-k(i)}(x_i)) = CV$$

- There is also a bias/variance tradeoff in CV estimates of prediction error:
  - *K large*: Less bias for the true prediction error; but potentially high variance because the *N* training sets are so similar to each other (hence different $\hat{f}$'s are highly dependent).
  - *K small*: tends to reverse these effects (smaller variance, larger bias).

  Often the point of diminishing returns in *this* bias/variance tradeoff is around $K = 5$ or 10.
- $K = N$ is called *"leave-one-out" cross-validation*. $\hat{f}^{-k}(x)$ least biased for $f_{opt}(x)$ (largest training sets), but most variance over replicated data sets (the training data sets are essentially all identical).



## *Using Cross-Validation*

- In principle, let $\alpha$ be any parameters of interest in $f(x;\alpha)$, and define

$$CV_K(\alpha) = \frac{1}{N}\sum_{k=1}^{N} L(y, \hat{f}^{-k(i)}(x_i;\alpha))$$

- Can use cross-validation to estimate $\hat{\alpha}$ (e.g. gradient-descent, etc.). This is sometimes done, but it is computationally expensive!
- Cross-validation is often useful for "tuning parameters" that are not part of the main model fitting but have to be set in some way
  - The list of variables/features to include in the model
  - The roughness penalty in a penalized function estimation problem
  - The degree of the polynomial in polynomial regression
  - The number of neighbors in *k*-nn
  - Etc.

*Aside...*

- For many *linear smoothers* ($\hat{y} = Sy$)

$$CV_N = \frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(S)/N}\right]^2 \equiv GCV$$

This is *generalized cross-validation*; it can be faster than $CV$, and it tends to smooth a bit more (choose models with lower df=trace($S$)) than CV.



## Additive and Generalized Additive Models

The basic idea of the *additive model* is to fit a model of the form

$$E[Y|X_1, \ldots, X_p] = \alpha + f_1(X_1) + \cdots + f_p(X_p)$$

by (penalized) least-squares or some similar method; thus the model is

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon$$

where $\epsilon$ has mean zero and finite variance. The $X_j$ may represent interactions or other functions of the original input variables, so there is scope for rather general model-fitting.

The functions $f_j(\cdot)$ are estimated non-parametrically, usually as smoothing splines or some other nonparametric regression approach.

For specificity we consider cubic smoothing splines. If we have replications $(y_i, x_{i1}, \ldots, x_{ip})$, $i = 1, \ldots, N$, we write the penalized residual sum of squares as

$$PRSS(\alpha, f_1, \ldots, f_p) = \sum_{i=1}^{N} \left\{ y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right\}^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j$$

where the $\lambda_i$ are tuning parameters (controlling bias/variance tradeoff for amount of smoothing).

The *backfitting* algorithm is a standard way to fit such models. Essentially it repeatedly fits the cubic spline model (or whatever smoothing model is in use) to the $j^{th}$ "deleted" residuals

$$r_i^{(j)} = y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})$$

(i.e. we fit $r_i^{(j)} = \hat{f}_j(x_{ij}) + \epsilon_i$).

It can be shown that PRSS is minimized by cubic smoothing splines $\hat{f}_j(x)$ with knots at each of the unique values of $x_{ij}$. The model can be identified by assuming that $\sum_i \hat{f}_j(x_{ij}) \equiv 0$, $\forall j$; which implies $\hat{\alpha} = \overline{y}$.

Additive models are again linear smoothers: $\hat{y} = Sy$ for some $S = S_{\lambda_1, \ldots, \lambda_p}$ that plays the role of the "hat matrix".



## Generalized Additive Models

The general form of generalized additive models is

$$g(E[Y|X_1, \ldots, X_p]) = \alpha + f_1(X_1) + \cdots + f_p(X_p)$$

together with stochastic model for observations. For example, the additive logistic regression has the form

$$\log \frac{P[Y = 1|X_1, \ldots, X_p]}{1 - P[Y = 1|X_1, \ldots, X_p]} = \alpha + f_1(X_1) + \cdots + f_p(X_p)$$

with the model of Bernoulli (or equivalently Binomial) responses for each fixed set of covariates $X_1, \ldots, X_p$.

The appropriate fitting criterion here is a penalized likelihood (rather than PRSS). The usual IRLS procedure for fitting generalized linear models reduces the nonlinear maximization to an iterative sequence of linear regression problems in the $f_j(X_j)$'s; the backfitting algorithm can be used in these linear regression problems to find the $\hat{f}_j(x)$'s in this case.

As with GLM's, fitted GAM's have a natural linear structure and may be treated as linear smoothers: there is a matrix $S_\lambda$ such that $g(\hat{y}) = S_\lambda y$, $df = trace(S_\lambda)$, etc.

## Projection Pursuit Regression

This is a variation of additive models,

$$E[Y|X_1, \ldots, X_p] = f_1(\omega_1^T X) + \cdots + f_M(\omega_M^T X)$$

which we would again like to fit by least squares. The functions $f_m$ are estimated by some smooth nonparmetric method (such as cubic smoothing splines). The vectors $\omega_m$ are of unit length and are estimated along with the $f_m$'s. Note that $X$ is the entire vector of independent variables in the problem.

Essentially we want to fit by penalized least-squares again. A variation of the backfitting algorithm is used:

- A single-term model $E[Y|X_1, \ldots, X_p] = f_1(\omega_1^T X)$ is estimated first, iterating between choice of $\omega_1$ and fitting the smoothing spline $f_i()$ (using penalized least-squares) until convergence.
- A second term $f_2(\omega_2^T X)$ is fitted to the residuals of the first model.
- The procedure continues until adding more terms does not appreciably improve the fit of the model (e.g. via cross-validation).
- After a "full" set of terms are fitted, the term set may be pruned back in backwards-selection style.