

Regression III

Lecture 6: Nonlinearity

Dave Armstrong

University of Wisconsin – Milwaukee
Department of Political Science

e: armstrod@uwm.edu
w: www.quantoid.net/ICPSR.php

1 / 30

Transformations/Polynomials or Non-Parametric Models

We have talked about two techniques that provide more or less flexible ways to model non-linearity.

- Transformations and Polynomials are easier to present and interpret, but still make functional form assumptions about the nature of the relationship.
- Nonparametric models make no global functional form assumptions, but are more difficult to interpret and present.

We might want to know when it makes sense to trade this flexibility for ease of presentation. As we saw yesterday, this makes sense when a model that imposes a functional form is not statistically worse than one that imposes a more flexible functional form

2 / 30

Confidence Bands for LPR Models

Predicted values \hat{y}_i can be obtained by

$$\hat{y}_i = \sum_{k=1}^n s_{ij}(x_i)y_i$$

The $s_{ij}(x_i)$ are the coefficients from the local regressions that transform y_i into its fitted value space. In matrix form, it is $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$, where \mathbf{S} is an $n \times n$ smoothing matrix whose (i, j) elements are $s_{ij}^2(x_i)$ - the local coefficients at each focal point.

$$\mathbf{S} = \begin{bmatrix} s_{11}^2(x_1) & s_{12}^2(x_1) & \cdots & s_{1k}^2(x_1) \\ s_{21}^2(x_2) & s_{22}^2(x_2) & \cdots & s_{2k}^2(x_2) \\ \vdots & \cdots & \ddots & \vdots \\ s_{k1}^2(x_k) & s_{k2}^2(x_k) & \cdots & s_{kk}^2(x_k) \end{bmatrix}$$

3 / 30

Variance of the Fitted Values

We can obtain the variance of the non-parametric fitted values as follows:

$$\begin{aligned} V(\hat{\mathbf{y}}) &= \mathbf{S}V(\mathbf{y})\mathbf{S}' \\ &= \sigma^2\mathbf{S}\mathbf{S}' \end{aligned}$$

We still need an estimate of σ^2 which we can obtain from:

$$\hat{\sigma}^2 = \frac{\sum e_i^2}{n - tr(\mathbf{S})}$$

notice here that the residual degrees is $n - tr(\mathbf{S})$. This needn't be an integer since $tr(\mathbf{S})$ is almost certainly non-integer valued.

We can form point-wise confidence bounds for \hat{y}_i as:

$$\hat{y}_i \in \hat{y}_i \pm 2\sqrt{v_{ii}}$$

where v_{ii} is the diagonal element of $V(\hat{\mathbf{y}}) = \hat{\sigma}^2\mathbf{S}\mathbf{S}'$.

4 / 30

F-Tests and Nonparametric Models

We can perform an F -test on nonparametric models to test the equivalent of $H_0 : \beta = 0$ in the linear model.

$$F = \frac{TSS - RSS/df_{mod} - 1}{RSS/df_{res}}$$

where $TSS = \sum(y_i - \bar{y})^2$, $RSS = \sum(y_i - \hat{y}_i)^2$, $df_{mod} = tr(\mathbf{S})$, and $df_{res} = n - tr(\mathbf{S})$.

We can also test the difference between non-parametric models and simpler parametric alternatives with an approximate F -test.

$$F = \frac{(RSS_0 - RSS_1)/J}{RSS_1/df_{res}}$$

where RSS_0 is the residual sum of squares from the parametric model, RSS_1 is the residual sum of squares from the nonparametric model, $J = tr(\mathbf{S}) - (k + 1)$ is the difference between the number of parameters estimated across the two models (where k is the number of regressors in the parametric model) and df_{res} is defined as above.

5 / 30

An Example

Remember yesterday we saw that a log-transformation of income was appropriate when trying to predict prestige. We might wonder whether that is *as good* as a model that doesn't impose that functional form.

```
> library(car)
> data(Prestige)
> mod <- lm(prestige ~ income, data=Prestige)
> trans.mod <- lm(prestige ~ log(income), data=Prestige)
> loess.mod <- loess(prestige ~ income, data=Prestige, span=.5,
+   family="symmetric")
> rss0 <- sum(trans.mod$residuals^2)
> rss1 <- sum(loess.mod$residuals^2)
> F0 <- ((rss0-rss1)/(loess.mod$trace.hat-trans.mod$rank))/
+   (rss1 / (loess.mod$n-loess.mod$trace.hat))
> cat("F = ", round(F0, 2), "\n", sep="")

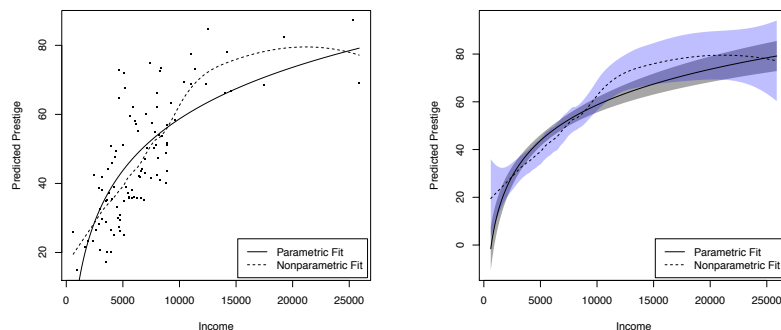
F = 1.73

> pval <- with(loess.mod, pf(F0, (trace.hat-length(coef(trans.mod))),
+   (n-trace.hat), lower.tail=F))
> cat("Pr( > F) = ", round(pval, 2), "\n", sep="")

Pr( > F) = 0.12
```

6 / 30

Plots of Predictions



7 / 30

Code for Previous Graph (Left Panel)

```
> inc.seq <- with(Prestige, seq(from=min(income), to=max(income), length=1000))
> para.pred <- predict(trans.mod,
+   newdata=data.frame(income=inc.seq), se.fit=T)
> nonpara.pred <- predict(loess.mod,
+   newdata=data.frame(income=inc.seq), se=T)
> plot(para.pred$fit ~ inc.seq, type="l", xlab="Income",
+   ylab="Predicted Prestige",
+   ylim=range(Prestige$prestige))
> lines(nonpara.pred$fit ~ inc.seq, lty=2)
> points(prestige ~ income, data=Prestige, pch=".", cex=3)
> legend("bottomright", c("Parametric Fit", "Nonparametric Fit"),
+   lty=c(1,2), inset=.01)
> NA
```

8 / 30

Code for Previous Graph (Right Panel)

```
> poly.x <- c(inc.seq, rev(inc.seq))
> para.lower <- with(para.pred, fit - 2*se.fit)
> para.upper <- with(para.pred, fit + 2*se.fit)
> nonpara.lower <- with(nonpara.pred, fit - 2*se.fit)
> nonpara.upper <- with(nonpara.pred, fit + 2*se.fit)
> para.poly.y <- c(para.lower, rev(para.upper))
> nonpara.poly.y <- c(nonpara.lower, rev(nonpara.upper))
> with(para.pred, plot(fit ~ inc.seq, type="n", xlab="Income",
+ ylab="Predicted Prestige",
+ ylim=c(min(c(para.lower, nonpara.lower)),
+ max(c(para.upper, nonpara.upper))))))
> polygon(poly.x, para.poly.y, col="gray65", border=NA)
> polygon(poly.x, nonpara.poly.y, col=rgb(0,0,1,.25), border=NA)
> lines(para.pred$fit ~ inc.seq, lty=1)
> lines(nonpara.pred$fit ~ inc.seq, lty=2)
> legend("bottomright", c("Parametric Fit", "Nonparametric Fit"),
+ lty=c(1,2), inset=.01)
> NA
```

9 / 30

Advantages of Splines over LPR Models

- Splines provide the best MSE fit
- Smoothing splines (a particular kind of spline) are designed to prevent overfitting
- Methods for estimating spline models continue to evolve.
- Splines are more easily incorporated in *semi*-parametric models.

10 / 30

Inference for Nonparametric Models

Splines

Simple Splines

More Complex Models

11 / 30

Definition of Splines

Splines are:

... piecewise regression functions we constrain to join at points called knots (Keele 2007, 70)

- In their simplest form, they are dummy regressors that we use to force the regression line to change direction at some value(s) of X .
- These are similar in spirit to LPR models where we use a subset of data to fit local regressions (but the window doesn't move here).
- These are also allowed to take any particular functional form, but they are a bit more constrained than the LPR model.

12 / 30

Choices in Spline Models

Degree: the analyst has to choose the degree of the polynomial fit to the subsets of the data.

Number of knots: the analyst has to choose the number of knots

Location of knots: most importantly - the analyst must choose the location of the chosen number of knots. This is probably the most important decision, though unlike span in the LPR, this decision can often be guided by theory.

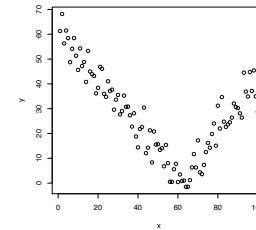
13 / 30

Regression Splines

We start with the following familiar model:

$$y = f(x) + \varepsilon$$

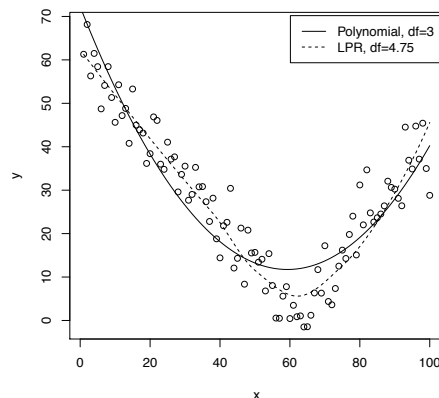
Here, we would like to estimate this with one model rather than a series of local models.



14 / 30

Inadequate Fixes

Given what we already learned, we could fit a quadratic polynomial or an lpr:



15 / 30

Simple Example

It is easy to figure out what sort of model we want:

- We know that we only need one knot - there is only one place where the direction of the relationship changes abruptly
- We know that the knot is around 60.
- We know that the relationship is linear between 0 and 60 and linear (though with a different slope) between 60 and 100.

These are all the things we need to know right now to model the relationship.

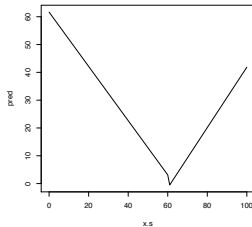
16 / 30

Splines vs. Dummy Interactions

You might ask, what does a spline buy us that we couldn't get with an interaction between x and a dummy variable coded 1 if $x > 60$ and zero otherwise.

$$y = b_0 + b_1x_1 + b_2d + b_3x \times d + e$$

This seems like a perfectly reasonable thing to do. What can it give you though:



17 / 30

Splines and Discontinuity

Splines are defined so the linear predictions meet smoothly at the knots (unlike in the previous figure). To fit the data in the previous slide, we need to estimate the following spline model:

$$y = \alpha + \beta_1x + \beta_2(x)_+ + \varepsilon$$

where

$$(x)_+ = \begin{cases} x & \text{if } x > c_1 \\ 0 & \text{if } x \leq c_1 \end{cases}$$

This will generate two different regression models:

$$\begin{aligned} \hat{y} &= \alpha + \beta_1x \text{ if } x \leq c_1 \\ &= \alpha + \beta_1x + \beta_2(x - c_1) \text{ if } x > c_1 \end{aligned}$$

In our example, $c_1 = 60$. Let's see what this gets us.

18 / 30

What do we get?

Notice that both pieces of the model generate exactly the same prediction at c_1 .

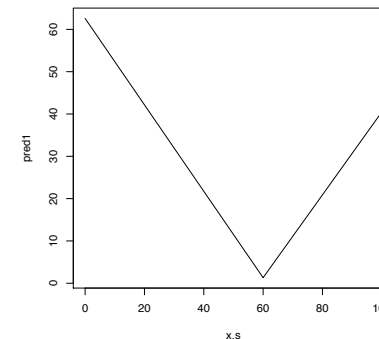
$$\begin{aligned} \hat{y} &= \alpha + \beta_1c_1 \text{ if } x \leq c_1 \\ &= \alpha + \beta_1c_1 + \beta_2c_1 - \beta_2c_1 \text{ if } x > c_1 \end{aligned}$$

By defining the $(\cdot)_+$ function subtracting the knot value from x , we force the two models to generate the same prediction at c_1 (the knot value).

19 / 30

Fixing the Discontinuity

We estimate the model suggested above including x and $(x)_+$ as regressors, which generates the following predictions:



20 / 30

Basis Functions

We can also think about this model in terms of basis functions. Basis functions are transformations to the model predictors over which we want to put a smooth function.

We need one basis for each piecewise function we want to put in the model. For the example above, we would need to include two basis functions:

$$B_L(x) = \begin{cases} c - x & \text{if } x < c \\ 0 & \text{otherwise} \end{cases}$$
$$B_R(x) = \begin{cases} x - c & \text{if } x > c \\ 0 & \text{otherwise} \end{cases}$$

21 / 30

Getting the Prediction

Now, instead of our X matrix (the model matrix) containing a column of 1's and a column of x , it has a column of 1's a column of $B_L(x)$ and $B_R(x)$.

With this new X matrix, we can generate:

$$\mathbf{H} = \mathbf{X}'\mathbf{X}\mathbf{X}'$$
$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$$

Notice what we get from this. When $x = 59$ and $c = 60$, we get:

$$\begin{aligned} \hat{y} &= \alpha + \beta_1 B_L(59) + \beta_2 B_R(59) \\ &= \alpha + \beta_1(60 - 59) + \beta_2(0) \\ &= \alpha + \beta_1 \end{aligned}$$

22 / 30

Getting the Prediction (2)

When $x = 60$:

$$\begin{aligned} \hat{y} &= \alpha + \beta_1 B_L(60) + \beta_2 B_R(60) \\ &= \alpha + \beta_1(0) + \beta_2(0) \\ &= \alpha \end{aligned}$$

When $x = 61$:

$$\hat{y} = \alpha + \beta_2$$

23 / 30

Investigating the Equation

We can get a sense of what is going on. To remind, our equation is:

$$\hat{y} = \alpha + \beta_1 B_L(60) + \beta_2 B_R(60)$$

When $x < c$, then,

$$\begin{aligned} \hat{y} &= \alpha + \beta_1(c - x) + 0\beta_2 \\ &= \underbrace{\alpha + \beta_1 c}_{\text{intercept}} - \beta_1 x \end{aligned}$$

When $x > c$, then,

$$\begin{aligned} \hat{y} &= \alpha + 0\beta_1 + \beta_2(x - c) \\ &= \underbrace{\alpha - \beta_2 c}_{\text{intercept}} + \beta_2 x \end{aligned}$$

24 / 30

Inference for Nonparametric Models

Splines

Simple Splines

More Complex Models

25 / 30

Other Basis Functions: Cubic

Often we might want to allow the estimated regression function to be more “curved”. To do this, we can combine polynomials with the regression splines of the previous few slides - *piecewise polynomials*.

For the spline models in the previous slides, we could extend them as follows:

$$\hat{y} = \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - c_1)_+^3$$

Below is convenient for estimation purposes, but less intuitive:

$$B(x, c) = \left[(c - .5)^2 - \frac{1}{12} \right] \left[(x - .5)^2 - \frac{1}{12} \right] \frac{1}{4} - \left[(|x - c| - .5) - .5(|x - c| - .5)^2 + \frac{7}{240} \right] \frac{1}{24}$$

26 / 30

Other Cubic Splines

Natural Splines:

- These are still cubic splines
- Two knots are added at $\min(x)$ and $\max(x)$ to ensure no erratic behavior happens at the extremes of the data

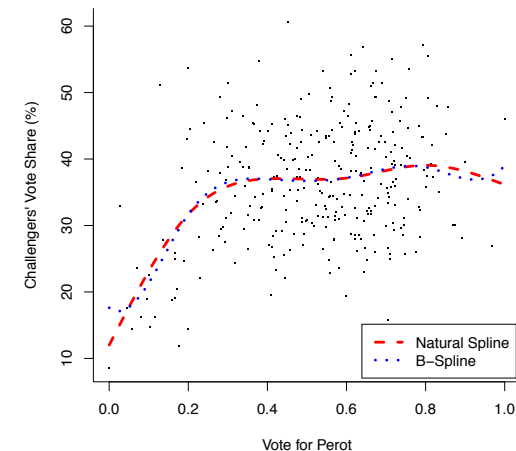
B-Splines:

- These are natural cubic splines
- The basis function is modified to reduce collinearity among the constructed variables.

The `splines` package in **R** has commands that making both natural cubic spline `ns()` and B-spline `bs()` basis functions

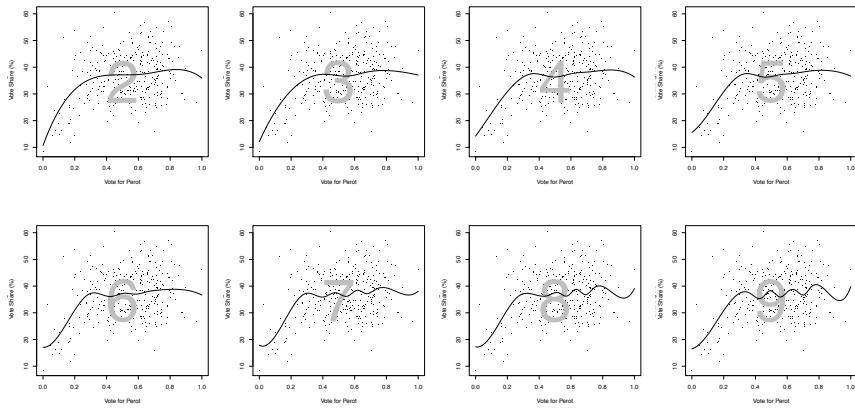
27 / 30

Comparison of Different Basis Functions



28 / 30

Comparing Number of Knots



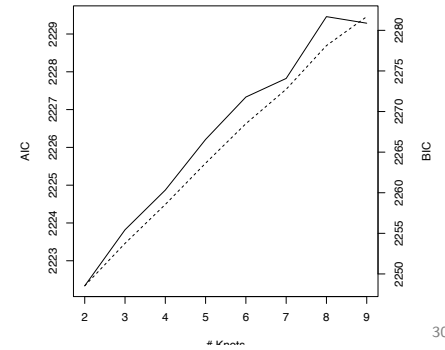
29 / 30

How Many Knots?

From the previous slide, it looks like two or three knots (the two in the upper-left) are probably the best fits. The others appear to be overfitting idiosyncrasies of the data.

We can use AIC or BIC to give us a sense of which model fits best (taking into account the number of parameters).

	AIC	BIC
2	2222.34	2248.54
3	2223.82	2253.76
4	2224.87	2258.56
5	2226.21	2263.64
6	2227.33	2268.51
7	2227.82	2272.74
8	2229.46	2278.12
9	2229.29	2281.69



30 / 30