Lecture 16: polynomial regression, splines, and kernel regression

3 December 2007

In this lecture we'll learn about improving regression modeling for continuous predictor variables with three available techniques:

- using polynomial functions of the predictor;
- using splines;
- using nonparametric/kernel methods.

1 Motivation

In the example data analysis of the dative alternation of Lecture 14, we had several categorical predictors for the categorical response variable of whether the double-object (DO) or prepositional-object (PO) alternative was used in a dative construction. In addition, we had two continuous response variables: the lengths (in words) of the theme and recipient arguments. In that lecture, we compared the options of modeling the effect of length on the linear predictor on two scales: raw length and log-length. As you can imagine, however, there is an infinite space of possible transformations that could be applied to a continuous predictor such as length (or, for example, frequency or familiarity) before it is entered into a regression. This lecture covers some of the options.

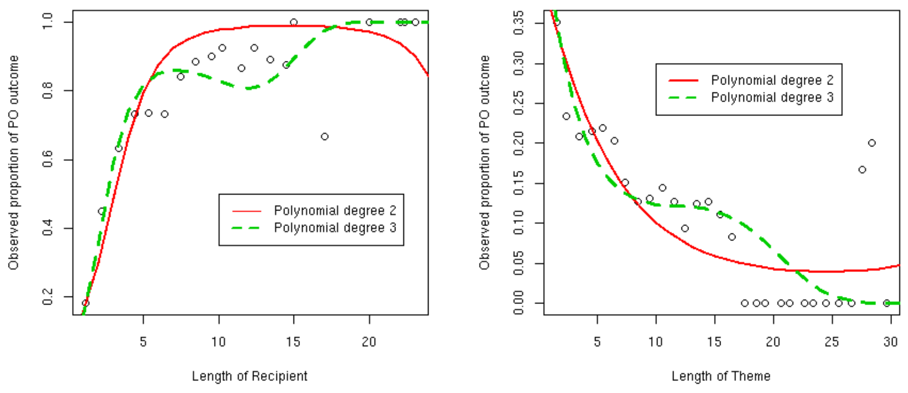

Figure 1: Quadratic polynomial regression for lengths of recipient and theme in `dative` dataset

2 Nonlinear terms

The simplest way to create a nonlinear relationship between the predictor variable X and the linear predictor is to create extra predictor variables that are transforms of X. For example, if we add a predictor that is the square of X we get the model

$$\eta = \alpha + \beta_1 X + \beta_2 X^2$$

As an example, let us construct a linear regression on length of recipient for the `dative` dataset and compare it with observed sample proportions estimated from a shingle (Figure 1):
We get the generalization that the effect of length flattens out with increasing length (consistent with the idea that log-space is the natural relation between length and linear predictor). However, extrapolation using polynomial regression is dangerous. This is particularly evident for the polynomial of degree 2, for which we get the wrong generalization—we would extrapolate that as the argument gets longer than the maximum observed length, its effect *reverses*.

```
my.intervals <- cbind(1:29-0.5,1:29+1.5)
response <- ifelse(dative$RealizationOfRecipient=="PP",1,0)
recipient.x <- with(dative,tapply.shingle(LengthOfRecipient,
  shingle(LengthOfRecipient,my.intervals),mean))
recipient.y <- with(dative,tapply.shingle(response,
  shingle(LengthOfRecipient,my.intervals),mean))
plot(recipient.x, recipient.y,xlab="Length of Recipient",
  ylab="Observed proportion of PO outcome")
dummy.data.frame <- data.frame(LengthOfRecipient=1:45)
dative.rec.glm.p2 <- glm(response ~ pol(LengthOfRecipient,2),dative,
  family="binomial")
lines(dummy.data.frame$LengthOfRecipient,invlogit(predict(dative.rec.glm.p2,
  dummy.data.frame)),col=2,lwd=2)
dative.rec.glm.p3 <- glm(response ~ pol(LengthOfRecipient,3),dative,
  family="binomial")
lines(dummy.data.frame$LengthOfRecipient,invlogit(predict(dative.rec.glm.p3,
  dummy.data.frame)),col=3,lwd=3,lty=2)
legend(10,0.5,c("Polynomial degree 2", "Polynomial degree 3"), lty=c(1,2),lwd=c(1,
# now repeat with LengthOfTheme rather than LengthOfRecipient
```

3 Restricted Cubic Splines

As mentioned in the previous section, polynomial regression is useful but potentially dangerous. For example, even a cubic regression creates strange predictions about highly frequent words (Figure 2, left):

```
> plot(exp(RTlexdec) ~ WrittenFrequency,english,pch=".")
> english.p3 <- lm(exp(RTlexdec) ~ pol(WrittenFrequency,3),english)
> x <- seq(0,12,by=0.1)
> dummy <- data.frame(WrittenFrequency = x)
> lines(x,predict(english.p3,dummy),lwd=2,col=2) # odd upturn at the end!
```

What is really happening in this case is that there is relatively little data at the right periphery; the flexibility granted by the cubic regression is being used to nudge the regression to an optimal shape in the large cloud. We can confirm this by comparing the results when we truncate the dataset at words of log written frequency above 8 for purposes of fitting the regression:

```
> english.p3.1 <- lm(exp(RTlexdec) ~ pol(WrittenFrequency,3),
    subset(english,WrittenFrequency < 8))
> lines(x,predict(english.p3.1,dummy),lwd=2,col=3,lty=2) # odd upturn even worse!
```

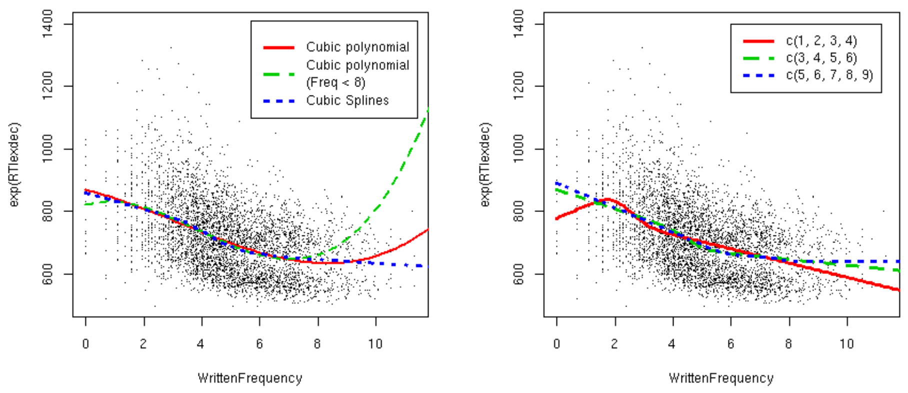

Figure 2: Modeling effects of written frequency on lexical-decision reaction time using polynomial regression and splines

The odd upturn got *much worse*, not better.

This is a difficult problem to deal with—extrapolation beyond what you have seen is never easy—but one approach that has become quite popular in the statistical literature is the use of RESTRICTED (also called NATURAL) CUBIC SPLINES. These are characterized as follows:

1. Place a set n of points—called KNOTS—on the real line: one at each of the maximum and minimum observed values of your predictor X, and the rest somewhere in between. Call these knots $\tau_1, \dots, \tau_n$.

2. The contribution of X to the linear predictor is determined as follows:

   (a) The i-th interval (the one lying between the knots τ_i and τ_{i+}) is characterized by a cubic curve, such that the contribution of X is determined by

   $$\beta_{i0} + \beta_{i1}(X - \tau_i) + \beta_{i2}(X - \tau_i)^2 + \beta_{i3}(X - \tau_i)^3$$

   for some choices of β_{ij};

   (b) Beyond the observed data (i.e., for $X < \tau_1$ and $X > \tau_n$), X contributes linearly to the linear predictor (i.e., as $\alpha + \beta(X - \tau_k)$);

(c) The spline parameters must be chosen such that the spline is SMOOTH everywhere (technically, such that the spline and its first and second derivatives are continuous).

In R we can use the rcs() function to introduce restricted cubic splines in our regression. [1] A simple example is given below: rcs()

```
> english.rcs <- ols(exp(RTlexdec) ~ rcs(WrittenFrequency),english)
> lines(x,predict(english.rcs,dummy),lwd=3,lty=3,col=4)
> legend(6,1300,c("Cubic polynomial","Cubic polynomial\n(Freq < 8)",
    "Cubic Splines"), lwd=3,lty=c(1,2,3),col=c(2,3,4))
```

Notice how the spline results have the desirable properties of the cubic polynomial regression (they give a more nuanced picture in the large cloud), but don't have the undesirable upturn at the right edge of the graph.

The rcs() function chooses knot positions for you. You can also choose your own (Figure 2, right):

```
> rcslines <- function(knots,i) {
  model <- ols(exp(RTlexdec) ~ rcs(WrittenFrequency,knots),english)
  lines(x,predict(model,dummy),lwd=3,lty=i,col=i+1)
}
> plot(exp(RTlexdec) ~ WrittenFrequency,english,pch=".",ylim=c(500,1400))
> knotsets <- list(c(1,2,3,4),c(3,4,5,6),c(5,6,7,8,9))
> rcslines(knotsets[[1]],1)
> rcslines(knotsets[[2]],2)
> rcslines(knotsets[[3]],3)
> legend(6,1400,as.character(knotsets),lwd=3,lty=1:3,col=2:4)
```

4 Further Reading

Treatments of spline regression and related techniques can be found in many places, including **?**, **?**, Section 8.7, **?**, Section 2.4, **?**, Chapter 5, and **?**, Sections 7.5 and 7.6.

[1] **Warning:** the lm() function does not interact properly with rcs()—use the ols() function (ordinary least squares) from the Design package instead. You can also use the ns() function from the splines library for restricted cubic splines.

References

Green, P. J. and Silverman, B. W. (1994). *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach.* London: Chapman & Hall.

Harrell, Jr, F. E. (2001). *Regression Modeling Strategies.* Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning.* Springer.

Maindonald, J. and Braun, J. (2007). *Data Analysis and Graphics using R.* Cambridge, second edition.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S.* Springer, fourth edition.