Splines, Knots and Penalties. The Craft of Smoothing

Computer Lab 1

Introduction. The computer labs make you familiar with the practical use of P-splines. There are three labs. The first one shows basic use of P-splines, staying very close to the formulas in the course. The second lab makes uses of a number of complete functions for smoothing of normal and non-normal (counts, binomial) data. The third lab makes use of the powerful *mgcv* library, written by Simon Wood. This is a well-documented and powerful R package; it does not provide P-splines by default, but the documentation shows how to add them.

Tools. The basic tool is the R system. Almost any version will work, but it makes sense to use a recent version. The computer labs were tested with R version 2.5. For computer labs 1 and 2 no additional packages (libraries) are needed. For computer lab 3 the *mgcv* package is needed. At the time of writing these instructions, no Windows binary for R version 2.5 was available, but the one for version 2.4 installs without complaints (first copy the ZIP file to your computer and install the package from this archive). If you work on one of the computers at the University, the software should already be installed.

Software files. The files to be used in the labs will be put in a place on the network. There will also be a USB memory stick that can be passed along. Some files contain functions that you will use. The other files are labeled ‘computer_lab1.r’ to ‘computer_lab3.r’. They contain step-by-step instructions for you to give to R. Make a copy of the files, so that you can change instructions freely, without loosing the original instructions.

How to work. If you are lazy, you copy individual instructions or blocks of instructions to R and see what happens. However, in the initial phase it is advisable to type the instructions yourself. It is better way to learn and to remember. Start R and change to directory in which you stored the files.

Comments. The instruction files contain many comments to document what is going on. A large number of comments contain section numbers, like S1 and S9. These are meant as references for the explanations that follow below.

1. First some data are simulated. They are somewhat special, because *x* is a linear sequence. Later we will work with scattered *x*.

2. A B-spline basis is computed, and plotted as symbols. The numbers corresponds to the columns of *B*. Smaller symbols and connecting lines make the graph more attractive.

3. It is easy to make the basis smaller (or larger) by changing the parameter *nseg*. Play with this number yourself.

4. By default, the B-splines are cubic: they consist of segments that are polynomials of degree three. Linear (or other degree) B-splines are obtained by changing the parameter *deg*.

5. The R function *lsfit()* performs linear regression. By default it uses an intercept. This should not be used here. See what happens if you specify *intercept = T*.

6. The *lsfit()* functions returns and R object with a number of fields. One of them, *coefficients*, contains the regression coefficients. For ease of use we put it in the variable *a*.

7. The default B-spline bases has *nseg* + *deg* basis functions. It is too much for our data, as you can see from the rather wiggly curve. With a smaller number for *nseg* we get nicer results.

8. We repeat the smoothing with linear B-splines. The result is a piece-wise linear fit. The breakpoints are specified explicitly by our choice of *nseg*; you can experiment with this parameter.

9. The function *lsfit()* uses a very stable algorithm for regression. Here we use the less stable explicit equations that you saw in the lectures. They have the advantage that it is more clear what happens (especially when we will add a penalty later). For many applications the stability is OK.

10. Our first penalty, one of order 1 (first differences of the coefficients). The penalty weight *lambda* is moderate. As a result we get a smoother result. Play with values of *lambda* between 0.01 and 100 (or a large range, if you like).

11. If we make *lambda* large, we get essentially a constant result. This is characteristic for the first order penalty.

12. The level of the constant line is equal to the average of *y.* We can check this with a horizontal line at the level *mean(y)*.

13. A heavy second order penalty makes the P-spline fit approach the linear regression line through the data.

14. We compute and plot the linear regression line as a check.

15. Now we will try scattered data, which will cause some slight complications. First we simulate the data.

16. The fit is OK, but we cannot simply plot *z* against *x*, because *x* is not ordered and the gaps between adjacent points are variable.

17. We compute a new basis, *Bg*, using a pleasant grid *xg*.

18. We now try interpolation and extrapolation. Play with *lambda* to see what happens