Splines, Knots and Penalties. The Craft of Smoothing

Computer Lab 2

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 scattered data, normally distributed around a trend, are simulated.

2. We smooth the data with the functions *ps.normal()*, which can be found in the file *ps_normal.r*, using the default parameters. To get a quick impression of the function, type *ps.normal*: it will produce a list of the function on the console, which you can scroll through.

3. We can easily set all parameters to our preferred values.

4. Now we read in some real data and look at the first lines of the data.

5. Smooth the scatterplot of ozone concentration vs. wind speed, using default parameters.

6. We like to optimize the amount of smoothing, using cross-validation (CV). The object that *ps.normal()* returns has a field *cv,* giving the RMS value of the CV error. We try some values of *lambda*. You can try other values, if you like.

7. It is a lot of work to try many values, so we generate a sequence of values for *lambda*, do the smoothing and collect the corresponding CV results. Notice that we use a linear grid for the og of *lambda*. The minimum is found for *log(lambda)* = 1.4. If you like, you can try a more detailed grid.

8. Smooth the data with *lambda* = 10^1.4. The plot looks pleasing.

9. We switch to count data, using the 'coal mining disaster' data, which are read in from a text file.

10. The function for smoothing of Poisson data is *ps.poisson()*. First we try the default parameter values.

11. To optimize the smoothing, we use AIC (not CV). The procedure for finding the minimum of AIC is the same: construct a (log-linear) grid of values for *lambda* and smooth the data for each value on the grid.

12. Apparently, *lambda* = 1 is a good choice.

13. Histograms are a common source of Poisson data. We read in the Old Faithful data, compute a histogram of the eruption lengths and smooth it for two values of *lambda*.

14. Searching for the optimum (of AIC) should be familiar now.

15. The smoothed histogram, using the optimal *lambda*, looks good.

16. Finally we will smooth binomial (0/1) data, using *ps.binomial()*.

17. Use the experience you gained from the previous part of the script to optimize *lambda* for these data.