Assignments: Smoothing Topic

Niels Richard Hansen

September 11, 2017

The first assignment topic is smoothing. If you draw the topic "smoothing" at the oral exam, you will have to present a solution of one of the two assignments below.

For all assignments in the course you should have the following five points in mind:

- How can you test that your implementation is correct?
- Can you implement alternative solutions?
- Can the code be restructured e.g. by modularization, abstraction or object oriented programming to improve readability?
- How does the implementation perform (benchmarking)?
- Where are the bottlenecks (profiling), and what can you do about them?

Some of these points are not covered yet (or you just don't master them right now), and you may need to return to your solution later in the course to improve and refine it. Moreover, the five points are not independent. Alternative solutions may provide ways of testing the implementation. Restructuring of code may improve or hurt performance. Optimization of performance may improve or hurt readability. And so on.

Assignment 1: Density smoothing

Implement a kernel smoother using the Epanechnikov kernel and implement bandwidth selection using either UCV or Sheather-Jones. Apply the implementation to the log(F12) data (as described in Exercise 10.1 in CS, and as used in Practical Exercises 1 to 3) and compare the result with the result of using `density` in R.

In addition to testing your implementation on the log(F12) data it is a good idea to consider simulated data as well. In particular for benchmarking and profiling.

For the Epanechnikov kernel, and other kernels with compact support, think about if you really need to evaluate all the terms in the sum (10.6)?

Assignment 2: Bivariate smoothing

Consider the same data as in Exercise 11.4a in CS. Implement a smoothing spline smoother using LOOCV for selecting the parameter λ. Compare the result with the result of using the R function `smooth.spline`. Compare also with a loess smooth.

A natural implementation relies of B-spline basis functions and their derivatives. These can be computed using the `splineDesign` function from the R package `splines` and numerical integration. Riemann-sum approximations can be used, but using Simpson's rule with breakpoints in the knots, the numerical integral becomes exact. Alternatively, the function `bsplinepen` from the fda package can be used.

In addition to testing your implementation on the temperature data it is a good idea to consider simulated data as well. In particular for benchmarking and profiling.

If you use linear algebra for computing the spline smooth think about the possibility of exploiting sparsity.