Assignments: Smoothing Topic

Niels Richard Hansen

September 3, 2020

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 generality, extendability and 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 estimation

Implement a kernel density estimator using the Epanechnikov kernel,

$$K(x) = \frac{3}{4}(1 - x^2)1_{[-1,1]}(x),$$

and implement one or more bandwidth selection algorithms using either AMISE plug-in methods or cross-validation methods. Test the implementation and compare the results with the results of using `density` in R.

It may be a good idea to use real data to investigate how the bandwidth selection works, but for benchmarking and profiling it is best to use simulated data.

Think about how to make your implementation as general and generic as possible.

Assignment 2: Bivariate smoothing

Implement a smoothing spline smoother using LOOCV for selecting the tuning parameter λ. Test the implementation and compare the results with the results of using the R function `smooth.spline`.

A natural implementation relies on 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 real data it is a good idea to consider simulated data as well. In particular for benchmarking and profiling.

Note that the teaching material covers some matrix decomposition techniques that can be particularly relevant in combination with LOOCV.