Resistant Nonparametric Smoothing with S-PLUS

Eva Cantoni
Department of Econometrics
University of Geneva
CH - 1211 Geneva 4, Switzerland

July 2002
revised January 2003

1 Goal

In this paper we introduce and illustrate the use of an S-PLUS set of functions to fit M-type smoothing splines with the smoothing parameter chosen by a robust criterion (either a robust version of cross-validation or a robust version of Mallows's C_p). The main reference is: Cantoni, E. and Ronchetti, E. (2001). Resistant selection of the smoothing parameter for smoothing splines. *Statistics and Computing*, **11**, 141–146.

2 The technique

Smoothing splines are flexible techniques for data modeling in a nonparametric framework. Despite their local nature, they can still suffer from potential robustness problems due to a few outlying points. M-type smoothing splines have therefore been proposed by Huber (1979). The approach consists of modifying the penalized criterion defining smoothing splines, by replacing the sum of squared residuals with a better suited function from the robustness point of view. But M-type smoothing splines are not enough to ensure resistance. In fact, the procedure to select the smoothing parameter is heavily affected by the presence of unusual data points. Cantoni and Ronchetti (2001) have proposed two robust techniques to overcome this undesirable behavior: the first one is a robust version of cross-validation that takes advantage of the one-step representation of M-splines, and the second one is a robustified version of Mallows's C_p constructed upon a weighted version of prediction error (see the original paper for details).

3 Code description

The code discussed in this paper has been developed under S-PLUS Version 3.4 Release 1 for Sun SPARC, SunOS 5.3 : 1996, and tested under S-PLUS Version 6.0 Release 1 for Sun SPARC, SunOS 5.6 : 2000. We were able to have the code running properly also in S-PLUS 6.1 Release 1 under Windows2000.

In addition to some auxiliary functions, the available code contains the main functions:

frob function to fit an M-type smoothing spline with a fixed (predefined) smoothing parameter.

opt.cv function to fit an M-type smoothing spline with a smoothing parameter chosen by robust cross-validation.

opt.RCp function to fit an M-type smoothing spline with a smoothing parameter chosen by robust C_p.

3.1 Instructions for use

The code is available as an archived file (`Mspline.tar`) which contains the files listed in Table 3.1.

The files `Ins3.4` and `Ins6.0` are scripts to install the code in S-PLUS version 3.4 and 6.0 respectively under a Unix system. The files `Msplinesfun3.4.s` and `Msplinesfun6.0.s` contain the whole set of S-PLUS functions for the two versions. `simulated.example.dump` is a dump file with the data of the example in Section 4.1 and `fraud.dat` is the dataset for the example in Section 4.2. The subdirectory `help3.4` (`help6.0` and `helpHTML`) contains the help files for the different versions of the software.

Unix
To restore the files from the archive use the command `tar xvf Mspline.tar`: a directory called `Mspline` is created. In order to use these functions, go to the directory `Mspline` and run the proper installation file depending on the software version you are using. You are then ready to open an S-PLUS session in this directory.

Windows
Decompress the file `Mspline.tar` with, for example, WinZip. A folder `Mspline` is created. Move the HTML helpfiles (the content of the folder `helpHTML`) to the folder `.Data\__Hhelp`. Source the code (`Msplinesfun6.0.s`) within S-PLUS.

- Mspline/Ins3.4
- Mspline/Ins6.0
- Mspline/Msplinesfun3.4.s
- Mspline/Msplinesfun6.0.s
- Mspline/simulated.example.dump
- Mspline/votfraud.dat
- Mspline/help3.4/e.psi.d
- Mspline/help3.4/frob.d
- Mspline/help3.4/matS.d
- Mspline/help3.4/my.smooth.spline.d
- Mspline/help3.4/opt.RCp.d
- Mspline/help3.4/opt.cv.d
- Mspline/help3.4/psihuber.d
- Mspline/help6.0/e.psi.sgml
- Mspline/help6.0/frob.sgml
- Mspline/help6.0/matS.sgml
- Mspline/help6.0/my.smooth.spline.sgml
- Mspline/help6.0/opt.RCp.sgml
- Mspline/help6.0/opt.cv.sgml
- Mspline/help6.0/psihuber.sgml
- Mspline/helpHTML/e.psi.html
- Mspline/helpHTML/frob.html
- Mspline/helpHTML/matS.html
- Mspline/helpHTML/my.smooth.spline.html
- Mspline/helpHTML/opt.RCp.html
- Mspline/helpHTML/opt.cv.html
- Mspline/helpHTML/psihuber.html

Table 1: Files in the archive `Mspline.tar`.

Figure 1: Classical and robust fit for the simulated example.

3.2 Computational details

The core of the function `frob` is based on the S-PLUS built-in function `smooth.spline`, which is used iteratively on pseudo-values to obtain the M-spline fit (for a fixed value of the smoothing parameter). `frob` also computes the value of the robust cross-validation criterion and of the robust C_p criterion for a given value of the smoothing parameter. The functions `opt.cv` and `opt.RCp` handle the numerical optimization of the robust cross-validation and robust C_p respectively, by making use of the S-PLUS function `optimize` and the output of `frob`.

The help files of the three main functions give detailed explanations about all their arguments and returned values. See also the tutorial examples of the next section.

Search interval

The optimization of the cross-validation and C_p criterion is performed over a search interval specified by the user. It is difficult to give a general guideline as of a "reasonable" choice of this interval because the smoothing parameter as defined in (3) in Cantoni and Ronchetti (2001) depends also on the range of the explanatory variable (e.g. proportional to $(\text{range}\, x)^3$). From the practical point of view, it is worth defining a short search interval, and widen it if the bounds are reached (in which case a warning message is produced by the code).

Restrictions

If the classical C_p method (`ind.chuber = Inf`, see Section 4.1 below) is used with highly contaminated data, it can happen that the optimization procedure fails because the objective function looses its convexity, see the discussion of Figure 3 in Cantoni and Ronchetti (2001). In this case, the code gives an error message.

4 Examples

In this section we illustrate the use of our routines on a simulated example (the same as used in Figure 1 in Cantoni and Ronchetti 2001) and on a real example taken from Simonoff (1996).

4.1 Simulated example

For the simulated example, we first restore the dump data file:

```
> source("simulated.example.dump")
```

Figure 2: Classical and robust fit for the "voting fraud" dataset.

The dataframe is now available in the object `simex`. We obtain the robust optimal fit in the sense of best robust cross-validation criterion by the command:

```
> rcv.simex <- opt.cv(simex$xx,simex$yy,bornes.opt=c(0,0.2))
```

where the default for the constant of the Huber function (`ind.chuber = 1.345`) is used here and were `bornes.opt` gives the interval over which the optimization is to take place. The object `rcv.simex` is a list containing the optimal value of the smoothing parameter (`rcv.simex$lambda.opt`), the corresponding value of the robust cross-validation criterion (`rcv.simex$cv.opt`) and all the information about the final fit (`rcv.simex$ajust.opt`, same as the output of `frob`).

Note that classical results can be reproduced by setting the value of the Huber constant to ∞:

```
> ccv.simex <-
      opt.cv(simex$xx,simex$yy,bornes.opt=c(0,0.2),ind.chuber=Inf)
```

One can then produce a graphical output with the following series of command lines (see Figure 1):

```
> motif()
> plot(simex$xx,simex$yy,xlab="xx",ylab="yy")
> lines(rcv.simex$ajust.opt$x,rcv.simex$ajust.opt$estim)
> lines(ccv.simex$ajust.opt$x,ccv.simex$ajust.opt$estim,lty=3)
> legend(0.4,5,c("Robust spline","Classical spline"), lty = c(1,3))
```

From Figure 1 we see that the value of the smoothing parameter obtained by classical cross-validation produces a fit that clearly over-smoothes the data without describing the behavior of the majority of them. This value is about 200 times larger than that obtained by the robust version of cross-validation and this affects the fit everywhere, and not only where the outlying points appears.

4.2 Voting fraud

This data set is about Democratic over Republican pluralities of voting machine and absentee votes for 22 Philadelphia County elections.

We first read the data from the data file:

```
> votfraud <- read.table("votfraud.dat",header=T)
```

and then run the optimization with respect to the C_p criterion in the robust and in the classical framework:

```
> RCp.votfraud <-
  opt.RCp(votfraud$Machine,votfraud$Absentee,bornes.opt=c(10e8,10e10))
> Cp.votfraud <-
  opt.RCp(votfraud$Machine,votfraud$Absentee,bornes.opt=c(10e8,10e10),
  ind.chuber=Inf)
```

We then fit the data points and the two curves:

```
> plot(votfraud$Machine,votfraud$Absentee,xlab="Democrat -
   Republican machine vote", ylab="Democrat - Republican absentee vote")
> lines(RCp.votfraud$ajust.opt$x,RCp.votfraud$ajust.opt$estim)
> lines(Cp.votfraud$ajust.opt$x,Cp.votfraud$ajust.opt$estim,lty=3)
> legend(10000,1100,c("Robust spline","Classical spline"), lty = c(1,3))
```

The solid fit of Figure 2 (obtained with the robust C_p criterion) describes better the relationship between machine vote and absentee vote for the majority of the data. It automatically picks the general behavior without being perturbated by the two outlying points we see in the picture. The dashed line obtained with the classical C_p technique misses the goal of describing the general pattern of this relationship, being very sensitive to the particular behavior of the upper points.

References

Cantoni, E. and E. Ronchetti (2001). Resistant selection of the smoothing parameter for smoothing splines. *Statistics and Computing 11*, 141–146.

Huber, P. J. (1979). Robust smoothing. In R. L. Launer and G. N. Wilkinson (Eds.), *Robustness in Statistics*, pp. 33–48. New York; London: Academic Press.

Simonoff, J. S. (1996). *Smoothing Methods in Statistics*. Berlin/New York: Springer-Verlag.