Rapporto n. 176

Titolo

Smooth Backfitting with R

Autori

Alberto Arcagni, Luca Bagnato

ottobre 2009

Dipartimento di Metodi Quantitativi per le Scienze Economiche ed Aziendali

Università degli Studi di Milano Bicocca

Via Bicocca degli Arcimboldi , 8 - 20126 Milano - Italia

Tel +39/02/64483103- Fax +39/02/64483105

Segreteria di redazione: Andrea Bertolini

Smooth Backfitting with R

by Alberto Arcagni, Luca Bagnato

The Smooth Backfitting Estimator (SBE) for additive models increases the estimation performances of the classical Backfitting Estimator. While for Backfitting many code has been proposed no usable programs for SBE are available. In this paper the package `sBF` for Smooth Backfitting using the Nadaraya-Watson estimator is presented. Some simulations are provided in order to test the proposed program. The manual of package `sBF`, including its functions, is given at the end of the paper.

Introduction

Additive models are a nonparametric multiple regression technique which allows to study the influence of different covariates separately. In particular, an additive model focuses on the estimation of the following regression model:

$$\begin{aligned} &E(Y|\mathbf{X}=\mathbf{x}) = m(\mathbf{x}), \\ &\text{with} \quad m(\mathbf{x}) = m_0 + \sum_{j=1}^{d} m_j\left(x_j\right) \ , \end{aligned} \tag{1}$$

where $\mathbf{X} = (X_1, X_2, \ldots, X_d)$ represents the covariates and Y the dependent variable. For identification it is usually assumed $E\left[m_j\left(X_j\right)\right] = 0$ for all $j = 1,2,\ldots,d$, so $m_0 = E(Y)$. Due to its structure such a model is very flexible and at the same time enables to overcome the problem known as *curse of dimensionality* (Bellman, 1961).

Model (1) was first studied in the context of input-output analysis by Leontief (1947) who called it *additive separable*. Additive models were introduced in the statistics literature at the beginning of the '80s and they led to several important results either practical and theoretical. Buja et al. (1989) and Hastie and Tibshirani (1990) provide a good review about additive models including estimation algorithms like Backfitting (Opsomer and Ruppert, 1997) and Marginal Integration (Linton and Nielsen, 1995). The SBE method, introduced by Mammen et al. (1999), consists of a more sophisticated version of the classical Backfitting but, as shown by Nielsen and Sperlich (2005), the SBE is more efficient, robust and easier to calculate.

Smooth Backfitting

Considering model (1), the Nadaraya-Watson SBE $\{\tilde{m}_0, \tilde{m}_1\left(x_1\right), \ldots, \tilde{m}_d\left(x_d\right)\}$ is defined as the minimizer of the smoothed sum of squares:

$$\begin{aligned} \int \sum_{i=1}^{n} &[Y_i - \bar{m}_0 - \bar{m}_1\left(x_1\right) - \cdots - \bar{m}_d\left(x_d\right)]^2 \cdot \\ &\cdot \prod_{j=1}^{d} K_h\left(x_j - X_{ij}\right) d\mathbf{x} \ , \end{aligned} \tag{2}$$

where $i = 1,\ldots,n$ denotes the observations and $j = 1,\ldots,d$ the covariates (or the directions) that are take in consideration. The minimization runs over the additive functions $\bar{m}_j\left(x_j\right)$ and the constant $\bar{m}_0$ with:

$$\int \bar{m}_j\left(x_j\right) \hat{p}_j\left(\mathbf{x}\right) d\mathbf{x} = 0 \ ,$$

where $\hat{p}_j\left(\mathbf{x}\right) = \int \hat{p}\left(\mathbf{x}\right) d\mathbf{x}_{-j}$ ($\mathbf{x}_{-j}$ denotes the vector $\mathbf{x}$ without the j-th component) is the marginal of the density estimate:

$$\hat{p}(\mathbf{x}) = n^{-1} \sum_{i=1}^{n} \prod_{l=1}^{d} K_h\left(x_l - X_{il}\right) \ . \tag{3}$$

After some basic algebra and standard theory, the solution related to the minimization problem in (2) can be obtained by solving the following system of equations ($j = 1,\ldots,d$) :

$$\tilde{m}_j = \hat{m}_j\left(x_j\right) - \sum_{k \neq j} \int \tilde{m}_k\left(x_k\right) \frac{\hat{p}_{jk}\left(x_j, x_k\right)}{\hat{p}_j\left(x_j\right)} dx_k - \bar{Y}, \tag{4}$$

$$\int \tilde{m}_j\left(x_j\right) \hat{p}_j\left(x_j\right) dx_j = 0 \ , \tag{5}$$

where

$$\hat{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \prod_{l=1}^{d} K_h\left(x_l - X_{il}\right) Y_i}{\sum_{i=1}^{n} \prod_{l=1}^{d} K_h\left(x_l - X_{il}\right)} \tag{6}$$

is a sort of pre-smoother for the conditional mean in (1) and

$$\hat{p}_{jk}\left(x_j, x_k\right) = n^{-1} \sum_{i=1}^{n} K_h\left(x_j - X_{ij}\right) K_h\left(x_k - X_{ik}\right) \tag{7}$$

is the two dimensional marginal of the full density estimate $\hat{p}(x)$. Using $\frac{\hat{p}_{jk}\left(x_j, x_k\right)}{\hat{p}_j\left(x_j\right)}$ in the equation (4), the curse of dimensionality can be eluded since only one and two dimensional marginal densities must be computed (Nielsen and Sperlich, 2005, pag. 47). The proposed R package follows the iterative algorithm provided by Nielsen and Sperlich (2005). Hereby the algorithm is quickly recalled.

We want to estimate each additive components on a predeterminate grid of point $i = 1,\ldots,M$. Thus the generic point where the functions $m_j(\cdot)$, $j =$

$1,\ldots,d$, will be estimated is $x_i^0 = \left(x_{i1}^0,\ldots,x_{id}^0\right)$ where x_{ij}^0 belong to the support of X_j for all j. Suppressing the index i of x_{ij}^0, $j = 1,\ldots,d$, the algorithm works as follows:

1. Set $r = 0$, and calculate $\tilde{m}_0 = n^{-1}\sum_{i=1}^n Y_i$ and simultaneously calculate the functions $\hat{m}_j(\cdot)$ and $\frac{\hat{p}_{jk}\left(x_j^0,x_k^0\right)}{\hat{p}_j\left(x_j^0\right)}$, $k \neq j$, then set $m^{(r)} := \hat{m}_j$.

2. For $j = 1,\ldots,d$ calculate for all the points x_j^0

$$\begin{aligned}\tilde{m}_j^{(r+1)}\left(x_j^0\right) &= \hat{m}_j\left(x_j^0\right) - \tilde{m}_0 + \\ &- \sum_{k<j}\int \tilde{m}_k^{(r+1)}(x_k)\frac{\hat{p}_{jk}\left(x_j^0,x_k\right)}{\hat{p}_j\left(x_j^0\right)}dx_k + \\ &- \sum_{k>j}\int \tilde{m}_k^{(r)}(x_k)\frac{\hat{p}_{jk}\left(x_j^0,x_k\right)}{\hat{p}_j\left(x_j^0\right)}dx_k\,.\end{aligned}$$

3. If the convergence criterion

$$\frac{\sum_{i=1}^M\left[\tilde{m}_j^{(r+1)}\left(x_{ij}^0\right) - \tilde{m}_j^{(r)}\left(x_{ij}^0\right)\right]^2}{\sum_{i=1}^M \tilde{m}_j^{(r)}\left(x_{ij}^0\right)^2 + \epsilon} < \epsilon \qquad (8)$$

is fulfilled then stop; otherwise set r to $r+1$ and go to step 2.

Program with R

The code we wrote is composed of two functions. The main function is called *sBF* and reproduces the algorithm as described in the previous section. The second function, *K*, is instrumental to the main function, and returns different kernel weighting functions. We start describing the *K* function. The function is defined as follows:

```
K(u, method = "gaussian")
```

The domain of the kernel functions is centered at the origin and generally the weight value returned by the kernel decreases while the distance `u` from the origin increases. The `method` parameter defines the kernel function to use. The default value, `gaussian`, applies the Normal distribution to define the weights. Other possible methods (Silverman, 1986) are: `unifrom`, `epanechnikov`, `biweight`, and `triweight`. These methods are generated by the same function changing a shape parameter.

The definition of the *sBF* function is

```
sBF(dat,
  depCol = 1,
  m = 100,
  windows = rep(20, ncol(dat)-1),
  bw = NULL,
  method = "gaussian",
  mx = 100,
  epsilon = 0.0001,
  PP = NULL,
  G = NULL
)
```

`dat` is a matrix or a data frame containing the observations by row and `depCol` reports the column position of the dependent variable (first column by default). The number of covariates, `d`, is defined by the number of columns of `dat` minus one. Nonparametric smoothing techniques usually require a `d`-dimensional grid on which the algorithms calculate the regressed functions: `m` is the number of equispaced points for any dimension of the grid. Thus we set a matrix `G` (`m` rows and `d` columns) where each column represents a grid related to a single univariate function $m_j(\cdot)$, $j = 1,\ldots,d$. Using matrix `G`, the `d`-dimensional grid (with m^d points), where the estimates of (1) are calculated, can be defined. Higher values of `m` determine more accurate estimates but longer computational time.

Bandwidth is an important parameter in smoothing techniques. It can be chosen in two different ways: through the argument `bw` or defining the number of windows into the range of the values of any independent variable through the argument `windows` (equal to 20 by default). Bandwidth is the width of the windows. Both the parameters `bw` and `windows` can be single values, then every smoother has the same bandwidth, or they can be vectors of length `d` to specify different bandwidths for any direction. Higher values of the bandwidth provide smoother estimates.

The parameter `method` defines the kernel function that will be used and it can get the same values as the *K* function's argument.

The iterative algorithm, described in the previous section, converges if the condition (8) is verified. `epsilon` defines the ϵ parameter in that condition. If the algorithm does not converge it will stop when the maximum number of iterations `mx` is achieved (equal to 100 by default).

sBF function calculates the matrix `PP` of the joint probabilities (7). Calculating `PP` takes a long computational time. In applications it could be useful using the same `PP` matrix for different estimates, e.g. to evaluate the impact of different bandwidths and develop algorithms to select optimal bandwidths (see, for example Nielsen and Sperlich, 2005, page 52). This reasoning applies also to the grid `G`. This is why the possibility to input matrices `PP` and `G` as parameters is given.

The *sBF* function returns an object of type list containing estimates and information related to the algorithm.

`mxhat` is a matrix `m×d` containing the estimated univariate functions (1) by column on each point of the grid (returned as `grid`), and `m0` the estimated value of m_0. By using `grid`, `mxhat` and `m0` it is possible to obtain estimated values also outside the grid and adopt, for example, some interpolation criteria. The function returns also a boolean variable `conv`, which indicates whether the iterations have converged and the number of the iterations `nit`. The function also returns the matrix of the joint probabilities (7) as object `PP` and the bandwidths as object `bw`.

The most interesting part of our code relates to the calculation of the conditional probabilities in (4). After obtaining matrix `PP`, the conditional probabilities are derived and grouped in matrices with dimensions depending on the number of the covariates and on the number of the grid points.

A large number of iterations should be done to calculate joint and conditional probabilities. Considering that `d` is the number of covariates we have to calculate these probabilities for $\binom{d}{2}$ couples of variables. It is noticeable that the number of calculation is very high whereas these probabilities must be evaluated on each pair of grid points (m^2). Instead of loops and multidimensional arrays, the use of matrices $m^2 \times \binom{d}{2}$ allowed us to exploit matrix and Kronecker products to performing the iterative algorithm steps.

Simulations

Two simulations were performed to validate the code. For both simulations the dependent variable is obtained as follows:

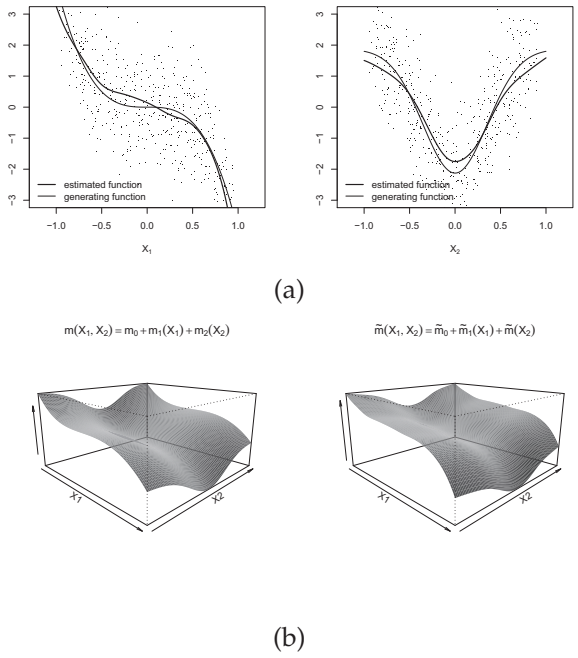
$$Y = \sum_{j=1}^{d} m\left(X_j\right) + \xi\,, \qquad \xi \sim N(0,1)\,, \tag{9}$$

where $X_j := 2.5 \arctan\left(Z_j\right)/\pi$ and the vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_d)$ has multinormal standard distribution and the correlation coefficients between the components are $\rho_{ij} = \rho\ \forall i, j = 1, \ldots, d,\ i \neq j$.

In the first simulation we show a simple model which has two covariates ($d = 2$). The two univariate functions constituting the additive model are the following:

$$m_1\left(X_1\right) = 4X_1^3 \qquad \text{and} \qquad m_2\left(X_2\right) = -4e^{-4X_2^2}\,. \tag{10}$$

The correlation parameter and the sample size are selected respectively equal to $\rho = 0.1$ and $n = 500$.



(a)

(b)

Figure 1: Estimates related to simulation with additive components (10). (a) Estimated functions $\tilde{m}_j(\cdot)$ (bold curve) and generating functions $m_j(\cdot)$ (thin curve) for $j = 1$ (graph on the left) and $j = 2$ (graph on the right). (b) Generating surface (graph on the left) and estimated one (graph on the right).

Figure 1 (a) shows the univariate estimated functions $\tilde{m}_j(\cdot)$, the generating functions $m_j(\cdot)$, $j = 1, 2$, and the scatterplots $(X_1, Y - \tilde{m}_0 - \tilde{m}_2(X_2))$ and $(X_2, Y - \tilde{m}_0 - \tilde{m}_1(X_1))$. Notwithstanding some boundary effects due to data sparseness, the estimated functions well adapt to the generating ones. Such a result seems clearer in Figure 1 (b) where the conditional mean $m(\cdot, \cdot)$ is compared to the estimated function $\tilde{m}(\cdot, \cdot)$.

In the second simulation we replicated the simulated model used by Nielsen and Sperlich (2005) and we compared the results. The additive components to insert in model (9) are the following

$$m\left(X_j\right) = 2 \sin \pi X_j\,, \quad j = 1, \ldots, d\,, \tag{11}$$

where X_j is defined as in the previous simulation. While $\rho = 0.1$ and $n = 500$, for ease in computation we use $d = 50$ instead of $d = 100$ as used in the cited article. The bandwidth choice for each covariate is related to the standard deviation of X_j, i.e. σ_j.

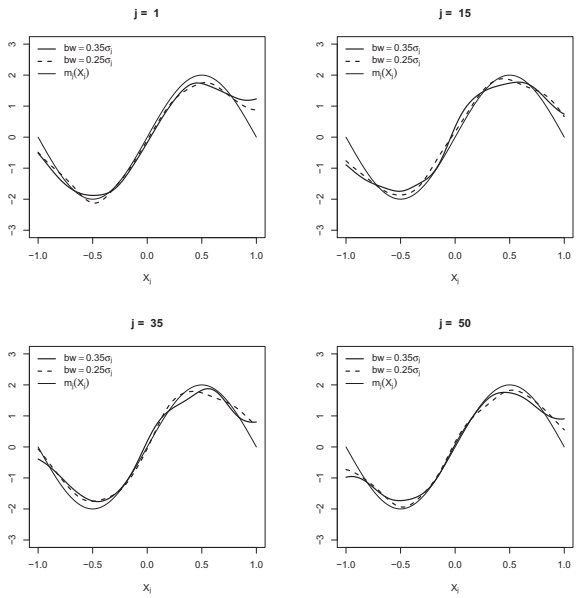


Figure 2: Estimated functions $\tilde{m}_j(\cdot)$ (bold and dashed curve) and generating functions $m_j(\cdot)$ (thin curve), $j = 1, 15, 35, 50$.

It is easy to note that the estimated functions are very close to the generating ones also for high values of j. Such a result confirms that *sBF* function is coherent with SBE and could be applied also using many covariates.

Conclusions

Additive models provide both flexible structure and interpretation capability, thus usable and efficient estimates are needed. Smooth Backfitting Estimator improves the classical Backfitting Estimator but usable programs for its calculation are not available. In this paper we present a R package which takes advantage of the peculiarity of such statistical environment. The program allows to obtain estimates in a short time also when models include many covariates. Simulations show the code validity and can be compared to the results obtained by other authors. Our package provides the building block for further investigation. In particular, it gives the possibility to study some bandwidth choice methods. To conclude the algorithm can be also extended using the local linear estimator instead of the Nadaraya-Watson estimator.

Bibliography

R. Bellman. *Adaptive Control Process*. Princeton University Press, Princeton, 1961.

A. Buja, T. Hastie, and R. Tibshirani. Linear smoothers and additive models (C/R: P510-555). *The Annals of Statistics*, 17:453–510, 1989.

T. Hastie and R. J. Tibshirani. *Generalised additive models*. Chapman and Hall: London, 1990.

W. Leontief. Introduction to a theory of an internal structure of functional relationships. *Econometrika*, 15:361–373, 1947.

O. Linton and J. P. Nielsen. A kernel method of estimating structured nonparametric regression based on marginal integration. *Biometrica*, 82:93–100, 1995.

E. Mammen, O. Linton, and J. Nielsen. The existence and asymptotic properties of a backfitting projection algorithm under weak conditions. *The Annals of Statistics*, 27(5):1443–1490, 1999.

J. P. Nielsen and S. Sperlich. Smooth backfitting in practice. *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, 67(1):43–61, 2005.

J. Opsomer and D. Ruppert. Fitting a bivariate additive model by local polynomial regression. *Annals of Statistics*, (25):186–211, 1997.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall Ltd, 1986.

Alberto Arcagni
University of Milano – Bicocca, Italy
a.arcagni@campus.unimib.it
Luca Bagnato
University of Milano – Bicocca, Italy
luca.bagnato@unimib.it

Package 'sBF'

October 1, 2009

Type Package

Title Smooth Backfitting

Version 1.0

Date 2009-09-24

Author A. Arcagni, L. Bagnato

Maintainer <a.arcagni@campus.unimib.it>, <luca.bagnato@unimib.it>

Description Smooth Backfitting for additive models using Nadaraya-Watson estimator

License GPL (>= 2)

LazyLoad yes

R topics documented:

`sBF-package` *Smooth Backfitting Estimator Package*

Description

Smooth Backfitting Estimator

Details

Package:	sBF
Type:	Package
Version:	1.0
Date:	2009-09-24
License:	GPL (>= 2)

Author(s)

A. Arcagni a.arcagni@campus.unimib.it and L. Bagnato luca.bagnato@unimib.it

References

T. Hastie and R. J. Tibshirani. *Generalised additive models*. Chapman and Hall: London, 1990.

E. Mammen, O. Linton, and J. Nielsen. *The existence and asymptotic properties of a backfitting projection algorithm under weak conditions*. The Annals of Statistics, 27(5):1443-1490, 1999.

J. P. Nielsen and S. Sperlich. *Smooth backfitting in practice*. Journal of the Royal Statistical Society, Series B: Statistical Methodology, 67(1):43-61, 2005.

See Also

sBF, K.

K *Kernel weighting function*

Description

Instrumental to the sBF function. It returns weights used in the Nadaraya-Watson estimator.

Usage

```
K(u, method = "gaussian")
```

Arguments

u	distance from the origin.
method	type of kernel function. The default value is gaussian, other possible methods are: unifrom, epanechnikov, biweight, and triweight.

Details

The domain of the kernel functions is centered at the origin and generally the weight value returned by the kernel decreases while the distance u from the origin increases.

References

Silverman, B. W. (1986) *Density Estimation*. London: Chapman and Hall.

See Also

sBF-package, sBF.

`sBF` *Smooth Backfitting Estimator*

Description

Smooth Backfitting for additive models using Nadaraya-Watson estimator.

Usage

```
sBF(dat, depCol = 1, m = 100, windows = rep(20, ncol(dat) - 1),
bw = NULL, method = "gaussian", mx = 100, epsilon = 1e-04,
PP = NULL, G = NULL)
```

Arguments

`dat`	matrix of data.
`depCol`	column of `dat` matrix in which the dependent variable is positioned.
`m`	number of grid points. Higher values of `m` imply better estimates and loger computational time.
`windows`	number of windows. (covariate range width)/`windows` provide the bandwidths for the kernel regression smoother.
`bw`	bandwidths for the kernel regression smoother.
`method`	kernel method. See function `K`.
`mx`	maximum iterations number.
`epsilon`	convergence limit of the iterative algorithm.
`PP`	matrix of joint probabilities.
`G`	grid on which univariate functions are estimated.

Details

Bandwidth can be chosen in two different ways: through the argument `bw` or defining the number of `windows` into the range of the values of any independent variable through the argument windows (equal to 20 by default). Bandwidth is the width of the windows. Both the parameters `bw` and `windows` can be single values, then every smoother has the same bandwidth, or they can be vectors of length equal tu the covariates number to specify different bandwidths for any direction. Higher values of the bandwidth provide smoother estimates.

In applications it could be useful using the same `PP` matrix for different estimates, e.g. to evaluate the impact of different bandwidths and develop algorithms to select optimal bandwidths (see, for example *Nielsen and Sperlich, 2005, page 52*). This reasoning applies also to the grid `G`. This is why the possibility to input matrices `G` and `PP` as parameters is given. The program creates `G` and `PP` if they are not inserted.

Value

`mxhat`	estimated univariate functions on the `grid` points.
`m0`	estimated constant value in the additive model.
`grid`	the grid.

conv	boolean variable indicating whether the convergence has been achieved.
nit	number of iterations performed.
PP	matrix of joint probabilities.
bw	bandwidths used for the kernel regression smoother.

See Also

sBF-package, K.

Examples

```
X <- matrix(rnorm(1000), ncol=2)
MX1 <- X[,1]^3
MX2 <- sin(X[,2])
Y <- MX1 + MX2
data <- cbind(Y, X)

est <- sBF(data)

par(mfrow=c(1, 2))
plot(est$grid[,1],est$mxhat[,1], type="l",
     ylab=expression(m[1](x[1])), xlab=expression(x[1]))
curve(x^3, add=TRUE, col="red")
plot(est$grid[,2],est$mxhat[,2], type="l",
     ylab=expression(m[2](x[2])), xlab=expression(x[2]))
curve(sin(x), add=TRUE, col="red")
par(mfrow=c(1, 1))
```

Index