Package ‘sBF’

February 20, 2015

Type Package

Title Smooth Backfitting

Version 1.1.1

Date 2014-12-19

Author A. Arcagni, L. Bagnato

Maintainer Alberto Arcagni <a.arcagni@campus.unimib.it>

Description Smooth Backfitting for additive models using
Nadaraya-Watson estimator

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2014-12-19 17:37:38

NeedsCompilation no

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