

Package ‘fpca’

April 17, 2009

Version 0.1-1

Date 2008-02-06

Title Restricted MLE for Functional Principal Components Analysis

Author Jie Peng <jie@wald.ucdavis.edu> , Debashis Paul <debashis@wald.ucdavis.edu>

Maintainer Jie Peng <jie@wald.ucdavis.edu>

Depends sm,splines

Description A geometric approach to MLE for functional principal components

License GPL (>= 2)

URL anson.ucdavis.edu/~jie/publication.html

Repository CRAN

Date/Publication 2008-03-24 21:33:20

R topics documented:

example	1
fpca	2
fpca.mle	3
Index	7

 example

An example

Description

A simulated dataset as an example

Usage

```
data(example)
```

Format

example is a list with six components (in the given order):

data.exp data matrix with three columns: column 1–ID, column 2–measurement, column 3–time of measurement.

eigenf true eigenfunctions: generated from cubic Bsplines with five equally spaced knots.

eigenv true eigenvalues: first–1, second–0.66, third–0.52, others–zero.

M.tr true number of basis functions: 5.

r.tr true dimension of the process: 3.

sig.tr true error standard deviation: 0.25.

Details

mean curve of the process is zero; principal component scores and errors are all i.i.d $N(0,1)$; there are 200 subjects, and each has 2~10 measurements uniformly distributed on $[0,1]$; in total there are 1227 measurements

 fpca

The fpca package: summary information

Description

The package implements the Newton Raphson procedure to estimate functional principal components from (sparsely and irregularly observed) longitudinal data described in Peng and Paul (2007).

Details

Missing values are not allowed. Subjects with only one measurement will be automatically excluded. The main function is 'fpca.mle'. A simulated data set can be called by 'data(example)'. Type 'help(example)' to see details. Packages 'sm' and 'splines' are used by this package. The code for EM (as initial estimate) is provided by Professor G. James in USC (with slight modifications).

Author(s)

J. Peng, D. Paul

References

- Peng, J. and Paul, D. (2007). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data (arXiv:0710.5343v1 [stat.ME])
- James, G. M., Hastie, T. J. and Sugar, C. A. (2000) Principal component models for sparse functional data. *Biometrika*, 87, 587-602.
- Yao, F., Mueller, H.-G. and Wang, J.-L. (2005) Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association* 100, 577-590

fpca.mle

*Restricted MLE of Functional Principal Components***Description**

A function to obtain restricted MLE of functional principal components by Newton Raphson algorithm for (sparsely and irregularly observed) longitudinal data. Subjects with only one measurement will be automatically excluded by 'fpca.mle'. Acknowledgements: The code for EM (as initial estimate) is provided by Professor G. James in USC (with slight modifications).

Usage

```
fpca.mle(data.m, M.set, r.set, ini.method="EM", basis.method="bs", sl.v=rep(0.5,10),
grid.l=seq(0,1,0.01), grids=seq(0,1,0.002))
```

Arguments

- | | |
|--------------|--|
| data.m | Matrix with three columns. Each row corresponds to one measurement for one subject. Column One: subject ID (numeric or string); Column 2: measurement (numeric); Column 3: corresponding measurement time (numeric); Missing values are not allowed. |
| M.set | numeric vector with integer values (≥ 4). Its element M is the number of basis functions used in the model for representing the eigenfunctions. |
| r.set | numeric vector with integer values (≥ 1). Its element r is the dimension of the process used in the model. |
| ini.method | string. It specifies the initial method for Newton. Its value is either "EM" (default): EM algorithm by James et al. 2002; or "loc": the local linear method by Yao et al. 2005. |
| basis.method | string. It specifies the basis functions. Its value is either "bs" (default): cubic Bsplines with equally spaced knots; or "ns": natural splines. Given, basis.method, each combination of M and r specifies a model. |

<code>sl.v</code>	numeric vector. An ordinary Newton step has length 1 which could be too large in the initial few steps. This vector specifies the step length for the first K steps, where K is the length of <code>sl.v</code> . If $K \geq \text{max.step}$ (see below), then <code>sl.v</code> will be truncated at <code>max.step</code> . If $K < \text{max.step}$, then the steps after the K-th step will have length 1. The default value of <code>sl.v</code> sets the first 10 steps of Newton to be of length 0.5.
<code>max.step</code>	integer. It is the maximum number of iterations Newton will run. Newton will be terminated if <code>max.step</code> number of iterations has been achieved even if it has not converged.
<code>grid.l</code>	numeric vector ranging from 0 to 1. This specifies the grid used by the local linear method (when "loc" is the initial method). Note that, due to the "sm" package used for fitting "loc", this grid can not be too dense.
<code>grids</code>	numeric vector ranging from 0 to 1. This specifies the grid used by EM (when EM is the initial method) and Newton. Note that, for both <code>grid.l</code> and <code>grids</code> , the denser the grid is, the more computation is needed.

Details

'fpca.mle' uses the Newton-Raphson algorithm on a `Stiefel` manifold to obtain the restricted maximum likelihood estimator of the functional principal components from longitudinal data. It also performs model selection over (M and r) based on an approximate leave-one-curve-out cross validation score.

Value

A list with components

<code>selected_model</code>	table. the selected M (number of basis functions) and r (dimension of the process).
<code>eigenfunctions</code>	numeric matrix. The estimated eigenfunctions under the selected model, evaluated at "grid" (see below).
<code>eigenvalues</code>	numeric vector. The estimated eigenvalues under the selected model.
<code>error_var</code>	numeric value. The estimated error variance under the selected model.
<code>fitted_mean</code>	numeric vector. The estimated mean curve (by local linear fitting) evaluated at "grid".
<code>grid</code>	numeric vector ranging from L1 and L2. This is the grid of time points rescaled to fit the actual time domain of the data, where L1 is the earliest time point, and L2 is the latest time point in the data.
<code>cv_scores</code>	numeric matrix. Approximate cv score for each combination of M and r.
<code>converge</code>	numeric matrix. Indicates the convergence of Newton for each combination of M and r. If an entry is less than 1e-3, it indicates that Newton converged under the corresponding model; otherwise it has not converged.

Author(s)

J. Peng, D. Paul

References

- Peng, J. and Paul, D. (2007). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data (arXiv:0710.5343v1 [stat.ME])
- James, G. M., Hastie, T. J. and Sugar, C. A. (2000) Principal component models for sparse functional data. *Biometrika*, 87, 587-602.
- Yao, F., Mueller, H.-G. and Wang, J.-L. (2005) Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association* 100, 577-590

Examples

```
##load data
data(example)

## candidate models
M.set<-c(4,5,6)
r.set<-c(2,3,4)

##parameters for fpca.mle
ini.method="EM"
basis.method="bs"
sl.v=rep(0.5,10)
max.step=50
grid.l=seq(0,1,0.01)
grids=seq(0,1,0.002)

##fit candidate models by fpca.mle

data.exp<-example[[1]]
result<-fpca.mle(data.exp, M.set,r.set,ini.method, basis.method,sl.v,max.step,
grid.l,grids)

##rescaled grid

grids.new<-result[[6]]

##model selection result: the true model M=5, r=3 is selected with smallest CV score among a

M<-result[[1]][1]
r<-result[[1]][2]

##compare estimated eigenvalues with the truth

result[[3]]      ## estimated
example[[3]]     ## true

##compare estimated error variance with the truth

result[[4]]      ## estimated
example[[6]]^2   ## true
```

```
##plot: compare estimated eigenfunctions with the truth

eigenf<-example[[2]] ##true
par(mfrow=c(2,2))
for(i in 1:r){
plot(grid.new,result[[2]][i,],ylim=range(result[[2]]),xlab="time",ylab=paste("eigenfunction",i))
points(grid, eigenf[i,],col=5,type="o")
}

##plot: compare estimated mean curve with the truth

plot(grid.new,result[[5]],xlab="time",ylab="mean curve",ylim=range(result[[2]]))
points(grid,numeric(length(grid)),col=5)

par(mfrow=c(1,1))

##look at the CV scores and convergence for each model: note that model (M=5, r=4) does not

result[[7]] ##CV
result[[8]] ##convergence
```

Index

*Topic **datasets**

example, [1](#)

*Topic **methods**

fpca.mle, [3](#)

*Topic **package**

fpca, [2](#)

example, [1](#)

fpca, [2](#)

fpca.mle, [3](#)