Package ‘GPFDA’

February 19, 2015

Title Apply Gaussian Process in Functional data analysis

Version 2.2

Date 2014-09-26

Depends R (>= 3.1), fda.usc, spam, fda

Imports MASS

Author Jian Qing Shi, Yafeng Cheng

Maintainer Yafeng Cheng <yafeng.cheng@nc1.ac.uk>

Description Use functional regression as the mean structure and Gaussian Process as the covariance structure.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-29 23:38:33

R topics documented:

GPFDAd-package .................................................. 2
betaPar .......................................................... 2
cov2 ............................................................. 4
cov.linear ....................................................... 4
cov.pow.ex ..................................................... 6
cov.rat.qu ...................................................... 7
D2 ................................................................. 9
gpfr .............................................................. 10
gpfrpred ........................................................ 13
gppredict ......................................................... 15
gpr ............................................................... 17
mat2fd .......................................................... 19
plot.gpfr ......................................................... 20
plot.gpr ........................................................ 21
xixj ............................................................. 22
xixj_sta ......................................................... 23
**GPFDA-package**  
*Gaussian Process in Functional Data Analysis*

**Description**

uses functional regression to be the mean function, and the Gaussian Process to be the covariance structure.

\[ y_m(t) = m\mu_m(t) + t\tau_m(x) + \epsilon_m(t) \]

Where \( m \) is the \( m^{th} \) data or curve; \( \mu_m \) is from functional regression; and \( \tau_m \) is from Gaussian Process regression with mean 0 covariance matrix \( k(\theta) \).

**Details**

- **Package:** GPFDA
- **Type:** Package
- **Version:** 1.0
- **Date:** 2013-09-30
- **License:**

**Author(s)**

Jian Qing Shi & Yafeng Cheng  
Maintainer: yafeng.cheng@ncl.ac.uk

**References**


**betaPar**  
*Create an fdPar object*

**Description**

Easy setting up for create a fdPar object.
**betaPar**

**Usage**

betaPar(betaList=NULL)

**Arguments**

- **betaList**: A list containing following items: `rtime`: range of time, default to be 0 and 1; `nbasis`: number of basis functions used in smoothing, default to be less or equal to 19; `norder`: the order of the functional curves default to be 6; `bspline`: logical, if True, b-spline is used, otherwise use Fourier basis, default to be True; `Pen`: default to be c(0,0); `lambda`: default to be 1e4; `bivar`: logical, if True, the bivariate basis will be calculated, otherwise normal basis, default to be False; `lambdas`: the smoothing parameter for the penalty of the additional basis, default to be 1e4.

**Details**

All items listed above have default values. If any item is required to change, add that item into the list, otherwise leave it as NULL. For example, if one only wants to change the number of basis functions, do: betaParlist(nbasis=11)

**Value**

- **betaPar**: An fdPar object

**Author(s)**

Jian Qing Shi & Yafeng Cheng

**References**


**See Also**

cov.linear,xixj_sta

**Examples**

```r
library(GPFDAt)
beta1=betaPar()
beta2=betaPar(list(nbasis=7,lambda=0.01))
```
---

### co2 data set for real example.

**Description**

data.frame with two variables, the first one is the response, second one is the time. The original data has 612 samples, but 5 of them are missing, which is removed from our sample.

**Usage**

```
callnorm
```

**Format**

A data frame with 607 observations on the following 2 variables.

**Details**

Data used in the real data example, see demo 'co2'. It is obtained from http://cdiac.esd.orl.gov/ftp/trends/co2/maunaloa.co2. Atmospheric CO2 values (ppmv) derived from in situ air samples collected at Mauna Loa, Hawaii, USA.

---

### cov.linear

**Covariance function. Linear covariance function.**

**Description**

Non-stationary covariance function.

**Usage**

```
cov.linear(hyper, Data, Data.new = NULL)
```

**Arguments**

- `hyper` The hyper parameters. Must be a list with certain names.
- `Data` The input data. Must be a vector or a matrix.
- `Data.new` The data for prediction. Must be a vector or a matrix. Default to be NULL.

**Details**

The names for the hyper parameters should be: "linear.a" for linear covariance function, "pow.ex.w", "pow.ex.v" for power exponential, "rat.qu.s", "rat.qu.a" for rational quadratic, "vv" for white noise. All hyper parameters should be in one list.

---
**Value**

| cov.lin | Covariance matrix |

**Author(s)**

Jian Qing Shi & Yafeng Cheng

**References**


**See Also**

cov.pow.ex; cov.rat.qu; gpr; xixj

**Examples**

```r
library(GPFDA)
require(MASS)

set.seed(30)
hp <- list('pow.ex.w'=log(10), 'linear.a'=log(10), 'pow.ex.v'=log(5),
   'vv'=log(1))
c <- seq(0.1, len=40)
idx <- sort(sample(1:40,21))
X <- as.matrix(c[idx])
Y <- mvnrnorm(n=40, mean=c-c, Sigma=(cov.linear(hp, c)+cov.pow.ex(hp, c)))[,1]
   *0.1*sin(c*6)
Y <- as.array(Y[idx])
x <- as.array(seq(0.1, by=0.03))
a <- gpr(X, Y, c('linear'), hp)
b <- gppredict(a, x)

upper <- b$pred.mean+1.96*b$pred.sd
lower <- b$pred.mean-1.96*b$pred.sd
plot(1:100, 1:100, col=0, xlab=range(x[,1]), ylab=range(Y),
   xlim=xlim=range(x[,1]), ylim=range(Y),
   main="Prediction", xlab="input ( x )", ylab="response")
plot(1:100, 1:100, col=0, xlab=range(x[,1]), ylab=range(Y),
   xlim=xlim=range(x[,1]), ylim=range(Y),
   main="Prediction", xlab="input ( x )", ylab="response")
polygon(c(x[,1], rev(x[,1])), c(upper, lower, Y), col = "grey90",
   border = NA)
points(x[,1], Y, pch=4, col=2)
lines(x[,1], Y)
lines(x[,1], b$pred.mean, col=3, lwd=2)
```
Covariance function. Powered exponential covariance function.

Description

Stationary covariance function.

Usage

cov.pow.ex(hyper, Data, Data.new = NULL, gamma = 1)

Arguments

hyper The hyper parameters. Must be a list with certain names.
Data The input data. Must be a vector or a matrix.
Data.new The data for prediction. Must be a vector or a matrix. Default to be NULL.
gamma Power parameter that cannot be estimated by simple non-linear optimization.

Details

The names for the hyper parameters should be: "linear.a" for linear covariance function, "pow.ex.w", "pow.ex.v" for power exponential, "rat.qu.s", "rat.qu.a" for rational quadratic, "vv" for white noise. All hyper parameters should be in one list.

Value

cov.pow.ex Covariance matrix

Author(s)

Jian Qing Shi & Yafeng Cheng

References


See Also

cov.linear; cov.rat.qu; xixj_sta
cov.rat.qu

Examples

```r
library(GPFDA)
require(MASS)

set.seed(30)
hp <- list('pow.ex.w'=log(10),'linear.a'=log(10),'pow.ex.v'=log(5),
'vv'=log(1))
c <- seq(0,1,len=40)
idx <- sort(sample(1:40,21))
X <- as.matrix(c[idx])
Y <- (mvnorm(n=40,mu=c-c, Sigma=(cov.linear(hp,c)+cov.pow.ex(hp,c))[,1]
)*0.1+sin(c*6)
Y <- as.matrix(Y[idx])
x <- as.matrix(seq(0,1,by=0.03))
a <- gpr(X,Y,c('pow.ex'),hp)
b <- gpredict(a,x)

upper=b$pred.mean+1.96*b$pred.sd
lower=b$pred.mean-1.96*b$pred.sd
plot(~100,-100,col=0,xlim=range(x[,1]),ylim=c(min(upper,lower,Y)-
0.1*abs(min(upper,lower,Y)),max(upper,lower,Y)+0.1*abs(max(upper,
lower,Y))),main="Prediction", xlab="input ( x )",ylab="response")
polygon(c(x[,1], rev(x[,1])), c(upper, rev(lower)),col = "grey90",
border = NA)
points(X[,1],Y,pch=4,col=2)
lines(X[,1],Y)
lines(x[,1],b$pred.mean,col=3,lwd=2)
```

cov.rat.qu

Covariance function. Rational quadratic covariance function.

Description

Stationary covariance function.

Usage

cov.rat.qu(hyper, Data, Data.new = NULL)

Arguments

- **hyper**: The hyper parameters. Must be a list with certain names.
- **Data**: The input data. Must be a vector or a matrix.
- **Data.new**: The data for prediction. Must be a vector or a matrix. Default to be NULL.
Details

The names for the hyper parameters should be: "linear.a" for linear covariance function, "pow.ex.w", "pow.ex.v" for power exponential, "rat.qu.s", "rat.qu.a" for rational quadratic, "vv" for white noise. All hyper parameters should be in one list.

Value

cov.rat.qu  Covariance matrix

Author(s)

Jian Qing Shi & Yafeng Cheng

References


See Also

cov.linear; cov.pow.ex; xixj_sta

Examples

```r
library(GPFDAt)
require(MASS)

set.seed(30)
hp <- list('pow.ex.w'=log(10), 'linear.a'=log(10), 'pow.ex.v'=log(5), 'vv'=log(1))
c <- seq(0,1,len=40)
idx <- sort(sample(1:40,21))
X <- as.matrix(c[idx])
Y <- rmvnorm(n=40, mu=c-c, Sigma=(cov.linear(hp,c)+cov.pow.ex(hp,c)))[,1]
Y <- as.matrix(Y[idx])
x <- as.matrix(seq(0,1,by=0.03))
a <- gpr(X,Y,c('rat.qu'))
b <- gppredict(a,x)

upper=b$pred.mean+1.96*b$pred.sd
lower=b$pred.mean-1.96*b$pred.sd
plot(-100,-100,col=0,xlim=range(x[,1]),ylim=c(min(upper,lower,Y)-0.1*abs(min(upper,lower,Y)),max(upper,lower,Y)+0.1*abs(max(upper,lower,Y))),main="Prediction", xlab="input (x)",ylab="response")
polygon(c(x[,1], rev(x[,1])), c(upper, rev(lower)),col = "grey90", border = NA)
points(X[,1],Y,pch=4,col=2)
lines(X[,1],Y)
lines(x[,1],b$pred.mean,col=3,lwd=2)
```
Description

Computer the second derivative of the likelihood function with respect to one of the hyper-parameters, with first and second derivative of the kernel function given.

Usage

\[ \text{D2}(d1, d2, \text{inv.Q}, \text{Alpha.Q}) \]

Arguments

- \( d1 \) First derivative of the kernel function with respect to the required hyper-parameter.
- \( d2 \) Second derivative of the kernel function with respect to the required hyper-parameter.
- \( \text{inv.Q} \) Inverse matrix of the covariance matrix.
- \( \text{Alpha.Q} \) This is \( iQY(iQY)'iQ \). where \( iQ \) is the inverse of the covariance matrix, \( Y \) is the response.

Details

The function is to calculate the second derivative of the normal likelihood, using the first and second derivative of the kernel functions. The first and second derivative need to be pre-defined, for example of customized covariance function, see "demo('co2')".

Value

\( \text{out} \) A number

Author(s)

Jian Qing Shi & Yafeng Cheng

References

gpfr

Gaussian Process in functional data.

Description
Use functional regression to be the mean structure and Gaussian Process to be the covariance structure.

Usage
gpfr(response=NULL,lReg=NULL,fReg=NULL,fbetaList_l=NULL, fyList=NULL,fbetaList_f=NULL, fxList=NULL,fbetaList=NULL, concurrent=TRUE,fbetaList_f=NULL, gpReg=NULL,hyper=NULL,Cov=c('pow.ex','linear'),gamma=1, time=NULL,NewHyper=NULL,accuracy=c('high','normal','low'), trace.iter=5,fitting=FALSE, rPreIdx=FALSE)

Arguments
response The training response. can be an fd object or a matrix with nrow samples, ncol time points
lReg The input variable for functional linear regression with scale covariates. Expected to be a matrix with nrow samples.
fReg The input variable for functional linear regression with functional covariates. Expected to be a matrix with nrow samples, or an fd object, or a list of matrices or fd objects.
fyList The list to control the smoothing of response. See details for more info.
fbetaList_l The list to control the smoothing of beta for functional regression with scale covariates. See details for more info.
fxList The list to control the smoothing of functional covariates for functional regression with functional covariates. See details for more info.
fbetaList_f The list to control the smoothing of beta for functional regression with functional covariates. See details for more info.
fbetaList The list to control the smoothing of functional covariates for functional regression with functional covariates and scale response. Not available for now.
concurrent Logical. If True concurrent functional regression will be carried out, otherwise the full functional regression will be carried out.
gpReg Data for training Gaussian Process. Expecting matrix, fd object, list of matrices or list of fd objects.
Cov Kernel function or covariance function type(s).
hyper Hyper parameter initial value. Default to be NULL.
NewHyper Vector of the names of the new hyper parameters from customized kernel function.
gamma Power parameter used in powered exponential.
time
accuracy
trace.iter
fitting
rPreIdx

time          Time used in globle setting for functional objects.
accuracy      Optimization accuracy. Default to be high.
trace.iter    Print the processing of iterations of optimization.
fitting       Is fitting required or not. Default to be F.
rPreIdx       Logical. If True, do random selected index for pre-optimization, otherwise use fixed index.

Details
fyList is a list with items: ‘time’: a sequence of time points default to be 100 points from 0 to 1; ‘nbasis’: number of basis functions used in smoothing, default to be less or equal to 23; ‘norder’: the order of the functional curves default to be 6, ‘bSpline’: logical, if True, b-spline is used, otherwise use Fourier basis, default to be True; ‘Pen’: default to be c(0,0), means that the penalty is on the second order derivative of the curve, since the weight for zero-th and first order derivatives of the curve are zero, ‘lambda’:default to be 1e-4, the smoothing parameter for the penalty.
fxList is a list of lists which are similar to fyList. Because it may contain different information for more than one functional covariates.
fbetaList, fbetaList_l and fbetaList_f are similar to each other. They are also expected to be list of lists. The items in each sub-list are: ‘time’: range of time, default to be 0 and 1; ‘nbasis’: number of basis functions used in smoothing, default to be less or equal to 19; ‘norder: the order of the functional curves default to be 6; ‘bSpline’: logical, if True, b-spline is used, otherwise use Fourier basis, default to be True; ‘Pen’: default to be c(0,0); ‘lambda’:default to be 1e4; ‘bivar’:logical, if True, the bivariate basis will be calculated, otherwise normal basis, default to be False; ‘lambdas’: the smoothing parameter for the penalty of the additional basis, default to be 1e4.

Note that user only write the item they need to change in the list, all items have default settings. See example below.

Value
A list of
hyper Estimated hyper-parameters
I A vector of estimated standard deviation of hyper-parameters
modellist List of models fitted before Gaussian process
CovFun Covariance function
gamma gamma used in Gaussian process powered exponential kernel
init_resp Initial response value
resid_resp Residual after the fitted value from models has been taken out
fitted Fitted value
fitted.sd Standard deviation of the fitted value
ModelType The model applied in the function.
lTrain Training data for functional regression with scalper covariates
fTrain Training data for functional regression with functional covariates
mfTrainfd  List of fd objects that from training data for functional regression with functional covariates
gpTrain  Training data for Gaussian Process
time  Time used in training in Gaussian Process
iuuL  Inverse of covariance matrix for lReg
iuuF  Inverse of covariance matrix for fReg
fittedFM  Fitted value from functional regression
fyList  fyList used in the function

Author(s)
Jian Qing Shi & Yafeng Cheng

References

See Also
gpr

Examples

library(GPFDA)
traindata=vector('list',20)
for(i in 1:20) traindata[[i]]=i
n=50
traindata=lapply(traindata,function(i) {
  x=seq(-3,3,len=n)
  y=sin(x^2)-x+0.2*runif(n,runif(1,-3,3),runif(1,0.5,3))
  xl=0.5*x^3+exp(x)+runif(n,runif(1,-3,3),runif(1,0.5,5))
  x2=cos(x^3)+0.2*runif(n,runif(1,-3,3),runif(1,0.5,5))
  mat=cbind(x,x1,x2,y)
  colnames(mat)=c('time','x1','x2','y')
  scale=t(c(2*(mean(y)>0.25)-1,(var(y)>3.6)*2-1,(sd(y)-sd(x)>1.4)*2-1))
  i=list(mat,scale)
})
lx=do.call('rbind',lapply(traindata,function(i)i[[2]]))
fx1=do.call('rbind',lapply(traindata,function(i)i[[1]][,2]))
fx2=do.call('rbind',lapply(traindata,function(i)i[[1]][,3]))
fy1=do.call('rbind',lapply(traindata,function(i)i[[1]][,4]))
time=old=traindata[[1]][[1]][,1]

# comment out because running time is a bit long
gpfrpred

Prediction of the Gaussian Process using functional regression

Description

Predict the new points in Gaussian Process using the training results

Usage

```
gpfrpred(object, testData, NewTime=NULL, lReg=NULL, fReg=NULL, gpReg=NULL, 
GP_predict=TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The result from training with class 'gpfrda'. If missing, function stops running.</td>
</tr>
<tr>
<td>testData</td>
<td>The test data. Must be matrix or fd object.</td>
</tr>
<tr>
<td>NewTime</td>
<td>New time for test data. If NULL, default setting will be applied.</td>
</tr>
<tr>
<td>lReg</td>
<td>The test scale data for functional regression with scale covariates.</td>
</tr>
<tr>
<td>fReg</td>
<td>The test functional data for functional regression with functional covariates.</td>
</tr>
<tr>
<td>gpReg</td>
<td>List of three items. The names of the items must be 'response', 'input', 'time'. For type I prediction, 'response' is the observed response for a new batch, 'input' is the observed functional covariates for a new batch, 'time' is the observation time for the previous two. If NULL, type II prediction will be carried out.</td>
</tr>
<tr>
<td>GP_predict</td>
<td>Logical. If true, GP prediction is carried out, otherwise functional prediction is carried out. Default to be True.</td>
</tr>
</tbody>
</table>

Details

Two types of prediction are supplied. Type one is the new batch has a few observations, type two is the new batch has no observations.

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ypred</td>
<td>matrix of predicted value with confidence interval. First column is the fitted value, second and third are the confidence interval.</td>
</tr>
<tr>
<td>ypred.mean</td>
<td>The mean value of the prediction.</td>
</tr>
<tr>
<td>ypred.sd</td>
<td>The standard deviation of the prediction.</td>
</tr>
<tr>
<td>time</td>
<td>time of test data</td>
</tr>
<tr>
<td>object</td>
<td>all items trained from gpfr if exists</td>
</tr>
</tbody>
</table>
Author(s)
Jian Qing Shi & Yafeng Cheng

References

See Also
gpr

Examples

```r
library(GPFDA)

# code from: demo('gpfrp')

traindata <- vector('list',20)
for(i in 1:20) traindata[[i]]<-i
n <- 50
traindata <- lapply(traindata,function(i) {
x <- seq(-3,3,len=n)
y <- sin(x^2)-x+0.2*rrnorm(n,0,3)
x1 <- 0.5*x^3+exp(x)+rrnorm(n,0,3)
x2 <- cos(x^3)+0.2*rrnorm(n,0,3)
mat <- cbind(x,x1,x2,y)
colnames(mat) <- c('time','x1','x2','y')
scale <- t(c(2*(mean(y)>0.25)-1,(var(y)>3.6)*2-1,(sd(y)-sd(x)>1.4)*2-1))
i <- list(mat,scale)
})

n <- 800 # test input
x <- seq(-3,3,len=n)
y <- sin(x^2)-x+0.2*rrnorm(n,0,3)
x1 <- 0.5*x^3+exp(x)+rrnorm(n,0,3)
x2 <- cos(x^3)+0.2*rrnorm(n,0,3)
mat <- cbind(x,x1,x2,y)
colnames(mat) <- c('time','x1','x2','y')
scale <- t(c(2*(mean(y)>0.25)-1,(var(y)>3.6)*2-1,(sd(y)-sd(x)>1.4)*2-1))
# testdata[[1]]=vector('list',3)
n <- 100 # test new points
xt <- seq(1,3,len=n)
yt <- sin(xt^2)-xt+0.2*rrnorm(n,0,3)
xt1 <- 0.5*xt^3+exp(xt)+rrnorm(n,0,3)
xt2 <- cos(xt^3)+0.2*rrnorm(n,0,3)
mat_t <- cbind(xt,xt1,xt2)
colnames(mat_t) <- c('time','xt1','xt2')
td <- list(mat,scale,mat_t)
```
gppredict

Prediction of the Gaussian Process

Description
Predict the new points in Gaussian Process using the training results or manual input

Usage
```
gppredict(train=NULL, Data.new=NULL, hyper=NULL, Data=NULL, Y=NULL, Cov=NULL, gamma=NULL, lrm=NULL, mean=0)
```

Arguments

- **train**: The result from training which is a 'gpr' object. Default to be NULL. If NULL, do training based on the other given arguments; if TRUE, other arguments (except for Data.new) will replaced by NULL; if FALSE, only do prediction based on the other given arguments.
- **Data.new**: The test data. Must be a vector or a matrix.
hyper  Hyper-parameter estimated from training. Can use manual input. Default to be NULL.

Data  The data from training. Must be a vector or a matrix. Default to be NULL.

Y  The response from training. Must be a vector or a matrix. Default to be NULL.

Cov  Names of covariance functions used. Default to be NULL.

gamma  Parameter used in power exponential covariance function. Default to be NULL.

lrm  The linear trend from learning. Default to be lrm. If lrm exists from learning, NULL will be replaced.

mean  Is the mean taken out when analysis? Default to be 0, which assumes the mean is zero. If assume mean is a constant, mean=1; if assume mean is a linear trend, mean='t'.

Details

Use the result from training to predict the value for new points.

Value

 CovFun  Covariance function type
 fitted  Fitted value of training data
 fitted.sd  Standard deviation of the fitted value of training data
 gamma  Parameter used in powered exponential covariance function
 hyper  Hyper-parameter estimated from training data
 I  Variance of the estimated hyper-parameters
 pred.mean  Estimated prediction mean
 pred.sd  Estimated prediction variance
 train.x  Training covariates
 train.y  Training response, may be transformed, for prediction use only
 train.yori  Original training response

Author(s)

Jian Qing Shi & Yafeng Cheng

References


See Also

gpr
Examples

library(GPFDA)
library(MASS) ## used to generate data
hp <- list('pow.ex.w'=log(10), 'linear.a'=log(10), 'pow.ex.v'=log(5), 'vv'=log(1))
c <- seq(0,1,len=40)
idx <- sort(sample(1:40,21))
X <- as.matrix(c[idx])
Y <- (mvnorm(n=40,mu=c-c,Sigma=(cov.linear(hp,c)+cov.pow.ex(hp,c)))[1,1]) +
    sin(c*6)
Y <- as.matrix(Y[idx])
x <- as.matrix(seq(0,1,by=0.03))
a <- gpr(X,Y,c('linear','pow.ex'))
b <- gpredict(a,x)

---

### gpr

Gaussian Process regression for single curve

Description

Gaussian Process regression for single curve with train data.

Usage

gpr(Data, response, Cov=c('linear','pow.ex'), hyper=NULL, NewHyper=NULL,
    mean=0, gamma=1, itermax=100, reltol=8e-10, trace=0)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>The input data from train data. Matrix or vectors are both acceptable. Some data.frames are not acceptable.</td>
</tr>
<tr>
<td>response</td>
<td>The response data from train data. Matrix or vectors are both acceptable. Some data.frames are not acceptable.</td>
</tr>
<tr>
<td>Cov</td>
<td>The kernel functions or covariance functions to use. Default is the sum of 'linear' and 'power exponentiation'.</td>
</tr>
<tr>
<td>hyper</td>
<td>The hyper parameters. Default is NULL. If not NULL, then must be a list with certain names.</td>
</tr>
<tr>
<td>NewHyper</td>
<td>Vector of the names of the new hyper parameters from customized kernel function. The names of the hyper-parameters must have the format: xxxxxxx.x, i.e. '6 digit' plus 'a dot' plus '1 digit'. This is required for both 'hyper' and 'NewHyper'</td>
</tr>
<tr>
<td>mean</td>
<td>Is the mean taken out when analysis? Default to be 0, which assumes the mean is zero. If assume mean is a constant, mean=1; if assume mean is a linear trend, mean='t'.</td>
</tr>
<tr>
<td>gamma</td>
<td>Power parameter used in powered exponential kernel function.</td>
</tr>
</tbody>
</table>
itermax  
Number of maximum iteration in optimization function. Default to be 100. Normally the number of optimization steps is around 20. If reduce 'reltol', the iterations needed will be less.

reltol  
Relative convergence tolerance. Smaller reltol means more accurate and slower to converge.

trace  
The value of the objective function and the parameters is printed every trace’th iteration. Defaults to 0 which indicates no trace information is to be printed.

Details

The most important function in the package, for fitting the GP model and store everything necessary for prediction. The optimization used in the function is 'nlminb'. Optimization might break down if the noise for the curve are too far away from normal. Jitter, LU decomposition and sparse matrix inverse are used to ensure the matrix inverse can always get an answer.

The names for the hyper parameters should be: "linear.a" for linear covariance function, "pow.ex.w", "pow.ex.v" for power exponential, "rat.qu.s", "rat.qu.a" for rational quadratic, "vv" for white noise. All hyper parameters should be in one list.

Value

<table>
<thead>
<tr>
<th>CovFun</th>
<th>Covariance function type</th>
</tr>
</thead>
<tbody>
<tr>
<td>fitted.mean</td>
<td>Fitted value of training data</td>
</tr>
<tr>
<td>fitted.sd</td>
<td>Standard deviation of the fitted value of training data</td>
</tr>
<tr>
<td>gamma</td>
<td>Parameter used in powered exponential covariance function</td>
</tr>
<tr>
<td>hyper</td>
<td>Hyper-parameter estimated from training data</td>
</tr>
<tr>
<td>I</td>
<td>Variance of the estimated hyper-parameters</td>
</tr>
<tr>
<td>train.x</td>
<td>Training covariates</td>
</tr>
<tr>
<td>train.y</td>
<td>Training response, may be transformed, for prediction use only</td>
</tr>
<tr>
<td>train.y0ri</td>
<td>Original training response</td>
</tr>
<tr>
<td>Q</td>
<td>Covariance matrix</td>
</tr>
<tr>
<td>inv</td>
<td>Inverse of the covariance matrix</td>
</tr>
<tr>
<td>mean</td>
<td>The mean assumed in the analysis</td>
</tr>
<tr>
<td>lrm</td>
<td>'lm' object if mean is a linear regression. NULL otherwise.</td>
</tr>
<tr>
<td>conv</td>
<td>0 means converge; 1 otherwise.</td>
</tr>
<tr>
<td>hyper0</td>
<td>starting point of the hyper-parameters</td>
</tr>
</tbody>
</table>

Author(s)

Jian Qing Shi & Yafeng Cheng

References

**See Also**

`gppredict; cov.linear; cov.pow.ex; cov.rat.qu; gpfr`

**Examples**

```r
library(GPFDAb)
library(MASS) ## used to generate data
hp <- list('pow.ex.w'=log(10),'linear.a'=log(10),'pow.ex.v'=log(5),
'vv'=log(1))
c <- seq(0,1,len=40)
idx <- sort(sample(1:40,21))
X <- as.matrix(c[idx])
Y <- (mvnorm(n=40,mu=c-c,Sigma=(cov.linear(hp,c)*cov.pow.ex(hp,c)))[,1]
)*0.1+sin(0.6)
Y <- as.matrix(Y[idx])
x <- as.matrix(seq(0,1,by=0.05))
a <- gpr(X,Y,c('linear','pow.ex'))

## NOT RUN
## Further codes to provide predictions and plot can be found in demos, for example
## > demo('gpr_ex1')
## END
```

---

**Create an fd object from a matrix**

**Description**

Easy setting up for creating an fd object

**Usage**

```r
mat2fd(mat,fdList=NULL)
```

**Arguments**

- `mat` Input data, should be a matrix with ncol time points and nrow replications or samples.
- `fdList` A list with following items: ‘time’: a sequence of time points default to be 100 points from 0 to 1; ‘nbasis’: number of basis functions used in smoothing, default to be less or equal to 23; ‘norder’: the order of the functional curves default to be 6, ‘bSpline’: logical, if True, b-spline is used, otherwise use Fourier basis, default to be True; ‘Pen’: default to be c(0,0), means that the penalty is on the second order derivative of the curve, since the weight for zero-th and first order derivatives of the curve are zero, ‘lambda’:default to be 1e-4, the smoothing parameter for the penalty.
Details

All items listed above have default values. If any item is required to change, add that item into the list, otherwise leave it as NULL. For example, if one only wants to change the number of basis functions, do: mat2fdSomeMatrix, list(nbasis=21)

Value

matfd  An fd object

Author(s)

Jian Qing Shi & Yafeng Cheng

References


See Also

cov.linear, xixj_sta

Examples

ry=rnorm(20, sd=10)
y1=matrix(0, ncol=100, nrow=20)
for(i in 1:20)  y1[i,]=sin(seq(-1, pi, len=100))*ry[i]

y1fd=mat2fd(y1)
y1fd=mat2fd(y1, list(lambda=1))

plot.gpfr  Plot Gaussian Process regression with functional mean for either training or predicting

Description

Plot Gaussian Process with functional mean for training or predicting with 'gpfr' class object.

Usage

## S3 method for class 'gpfr'
plot(x, ..., type=c('raw', 'fitted', 'prediction'))

Arguments

x  The 'gpr' object from either training or predicting of the Gaussian Process.

...  Other arguments from general 'plot' function, such as: 'axes', etc.

type  Function provides three types of plots: raw, fitted and prediction.
plot.gpr

Author(s)

Jian Qing Shi & Yafeng Cheng

See Also

gpfr; gpfrpred; plot; plot.gpr

plot.gpr

Plot Gaussian Process training or predicting only for 'gpr' class object.

Usage

## S3 method for class 'gpr'
plot(x, ..., fitted = FALSE, col.no = 1)

Arguments

x The 'gpr' object from either training or predicting of the Gaussian Process.

... Other arguments from general 'plot' function, such as: 'axes', etc.

fitted Plot fitted value or not. Default to be FALSE, which is to plot the predictions.

col.no Column number of the input matrix. If the input matrix has more than one columns, than one of them will be used in the plot. Default to be the first one.

Author(s)

Jian Qing Shi & Yafeng Cheng

See Also

gppredict; gpr; plot
xixj  

*Linear kernel function component.*

**Description**

Component to build a linear kernel function or similar.

\[ M = \sum a_i \ast x_i' \ast x_i^T \]

where \( x_i \) is the \( i^{th} \) column of the input matrix; \( a_i \) is the \( i^{th} \) element of the weight vector. Note that \( x \) and \( x' \) might be different. It is for non-stationary kernel functions.

**Usage**

\[ \text{xixj(mat, mat.new=NULL, a=NULL)} \]

**Arguments**

- `mat` Input data, could be a matrix or a vector.
- `mat.new` Second input data, could be a vector or a matrix. Default to be NULL. If NULL, `mat.new=mat`.
- `a` Weight to be add on each column of the matrix.

**Details**

When all ‘\( a \)’ are 1, this is simply \( \text{mat} \ast \text{t(mat.new)} \). If one wants to involve linear kernel components in customized covariance matrix, this function will be used in derivatives of the kernel function. See examples in `demo('co2')`.

**Value**

- `out` A symmetric matrix used to build the linear kernel or similar

**Author(s)**

Jian Qing Shi & Yafeng Cheng

**References**


**See Also**

`cov.linear`, `xixj_sta`
Stationary kernel function component.

Description

Component of the distance to build a stationary kernel function or similar.

\[ M = \sum w_i (x_i' - x_i^T)^{\text{power}} \]

where \( x_i \) is the \( i^{th} \) column of the input matrix; \( w_i \) is the \( i^{th} \) element of the weight vector. Note that \( x \) and \( x' \) might be different.

Usage

\[ \text{xixj_sta(mat, mat.new=NULL, w=NULL, power=NULL)} \]

Arguments

- **mat** Input data, could be a matrix or a vector.
- **mat.new** Second input data, could be a vector or a matrix. Default to be NULL. If NULL, \( \text{mat.new} = \text{mat} \).
- **w** Weight to be add on each column of the matrix.
- **power** Argument 'power' X 2 will be the power to put on the distance. Default power is 1, which means \( \text{distance}^2 \). The range of the power to put on the distance is 0 to 2, thus argument 'power' is from 0 to 1.

Details

If one wants to involve stationary kernel components in customized covariance matrix, this function will be used in derivatives of the kernel function. See examples in demo('co2').

Value

- **out** A symmetric matrix used to build the linear kernel or similar

Author(s)

Jian Qing Shi & Yafeng Cheng

References


See Also

- **cov.linear**
Index

*Topic functional, Gaussian Process

GPFD{A} package, 2

betaPar, 2

cov2, 4
cov.linear, 3, 4, 6, 8, 19, 20, 22, 23
cov.pow.ex, 5, 6, 8, 19
cov.rat.qu, 5, 6, 7, 19

D2, 9

GPFD{A} (GPFD{A} package), 2
GPFD{A} package, 2
gpfr, 10, 19, 21
gpfrpred, 13, 21
gppredict, 15, 19, 21
gpr, 5, 12, 14, 16, 17, 21

mat2fd, 19

plot, 21
plot.gpfr, 20
plot.gpr, 21, 21

xixj, 5, 22
xixj_sta, 3, 6, 8, 20, 22, 23