Package 'outerbase'

October 14, 2022

Type Package

Title Outer Product Regression

Version 0.1.0

Date 2022-06-03

Maintainer Matthew Plumlee <mplumlee@northwestern.edu>

Description High-dimensional regression using outer product models. Research on the methods is currently under investigation and published resources will be posted as they are available. As the method is new, the website is the best resource for understanding the principals. Some of the core ideas are based on Plumlee and coauthors' work on analysis of grid-structured experiments described in Plumlee (2014) <doi:10.1080/01621459.2014.900250> and Plumlee, Erickson, Ankenman, Lawrence (2021) <doi:10.1093/biomet/asaa084>. Some additional textbooks for additional information on Gaussian processes are Rasmussen and Williams (2005) <doi:10.7551/mitpress/3206.001.0001> and Gramacy (2022) <doi:10.1201/9780367815493>.

License MIT + file LICENSE

URL https://mattplumlee.github.io/outerbase/, https://github.com/MattPlumlee/outerbase/

BugReports https://github.com/MattPlumlee/outerbase/issues

NeedsCompilation yes

Imports methods, Rcpp, utils, stats

LinkingTo Rcpp, RcppArmadillo

SystemRequirements GNU make

Encoding UTF-8

RoxygenNote 7.2.0

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Author Matthew Plumlee [aut, cre]

Repository CRAN

Date/Publication 2022-06-09 14:10:02 UTC

R topics documented:

`outerbase-package` *outerbase*

Description

High-dimensional regression using outer product models. Research on the methods is currently under investigation and published resources will be posted as they are available. As the method is new, the website is the best resource for understanding the principals. Some of the core ideas are based on Plumlee and coauthors' work on analysis of grid-structured experiments described in Plumlee (2014) doi: 10.1080/01621459.2014.900250 and Plumlee, Erickson, Ankenman, Lawrence (2021) doi: 10.1093/biomet/asaa084. Some additional textbooks for additional information on Gaussian processes are Rasmussen and Williams (2005) doi: 10.7551/mitpress/3206.001.0001 and Gramacy (2022) doi: 10.1201/9780367815493.

Author(s)

Maintainer: Matthew Plumlee <mplumlee@northwestern.edu>

See Also

Useful links:

- https://mattplumlee.github.io/outerbase/
- https://github.com/MattPlumlee/outerbase/
- Report bugs at https://github.com/MattPlumlee/outerbase/issues

---

`BFGS_lpdf` *BFGS lpdf*

Description

A wrapper for codeBFGS_std that is useful for easily calling parameter optimization for this package with as few lines as possible. Note that `om` and `logpdf` will be set to optimal parameters, the return is simply for information.

Usage

```
BFGS_lpdf(
  om,
  logpdf,
  parlist = list(),
  newt = FALSE,
  cgsteps = 100,
  cgtol = 0.001,
  ...
)
```

Arguments

om	an outermod instance
logpdf	a lpdf instance
parlist	an initial point, which are pulled from ‘om‘ and ‘logpdf‘ if not provided
newt	boolean for if Newtons method should be used
cgsteps	max number of cg iterations, if newt=FALSE
cgtol	cg tolerance, if newt=FALSE
...	additional parameters passed to BFGS_std

Value

a list of information from optimization.

BFGS_std *BFGS standard*

Description

Do generic minimization of a function funcw that takes a list parlist using the "Broyden-Fletcher-Goldfarb-Shanno" (BFGS) algorithm. Useful for hyperparameter optimization because it handles infinite returns fairly easily.

Usage

```
BFGS_std(funcw, parlist, B = NULL, lr = 0.1, ..., verbose = 0)
```

Arguments

funcw	An object to optimize
parlist	An initial point as a list
B	An initial Hessian to start from
lr	An initial learning rate to start from
...	additional parameters passed to funcw
verbose	an integer from 0-3 where larger prints more information

Value

a list of information from optimization, with the value stored in parlist

covf *covariance function class*

Description

This is a base class designed to handle the specific features of covariances needed for outerbase. Polymorphism allows for the implied methods to be used across several similar classes.

Value

no returns, this is a class which contains methods

Fields

`covf$hyp` hyperparameters for this specific correlation function

`covf$lowbnd,covf$uppbnd` upper and lower bounds for the inputs to the covariance function.

`covf$cov(x1,x2)` returns the covariance matrix between two vectors of inputs `x1` and `x2`

`covf$covdiag(x1)` returns the diagonal of the covariance matrix between `x1` and itself

`covf$cov_gradhyp(x1,x2)` returns a cube of the gradient the `cov` with respect to the covariance hyperparameters

See Also

derived class: `covf_mat25`, `covf_mat25pow`, `covf_mat25ang`

covf_mat25 *Matern covariance function*

Description

```
covf = new(covf_mat25)
```

This is the standard Matern covariance function which has form

$$c(x_1, x_2) = (1 + |h| + h^2/3)\exp(-|h|)$$

where $h = (x_1 - x_2)/\rho$ and ρ=`exp(2*hyp[0])`.

Value

no returns, this is a class which contains methods

See Also

base class: `covf`

covf_mat25ang — *Matern covariance function with angular transform*

Description

`covf = new(covf_mat25ang)`

This is the standard Matern covariance function with a power transformation which has form

$$c(x_1, x_2) = (1 + |h| + h^2/3)\exp(-|h|)$$

where

$$h = \sqrt{(\sin(x_1) - \sin(x_2))^2/\rho_s + (\cos(x_1) - \cos(x_2))^2/\rho_c}.$$

hyp is a two dimensional vector with ρ_s=`exp(2*hyp[0])` and ρ_c=`exp(2*hyp[1])`.

Value

no returns, this is a class which contains methods

See Also

base class: covf

covf_mat25pow — *Matern covariance function with power transform*

Description

`covf = new(covf_mat25pow)`

This is the standard Matern covariance function with a power transformation which has form

$$c(x_1, x_2) = (1 + |h| + h^2/3)\exp(-|h|)$$

where $h = (x_1^\alpha - x_2^\alpha)/\rho$ and hyp is a two dimensional vector with ρ=`exp(2*hyp[0]+0.25*hyp[1])` and α=`exp(0.25*hyp[1])`.

Value

no returns, this is a class which contains methods

See Also

base class: covf

gethyp *Get the hyperparameters*

Description

```
hyp = gethyp(om)
```

Gets the current hyperparameters from an outermod instance. It formats them in a way that makes reading in R easier.

Arguments

om	an outermod instance

Value

a vector of parameters

See Also

outermod

Examples

```
om = new(outermod)
setcovfs(om, c("mat25", "mat25", "mat25"))
hyp = gethyp(om)
print(hyp)
```

getpara *Get the model parameters*

Description

```
para = getpara(logpdf)
```

This function gets the current parameters from an lpdf class instance. It formats them in a way that makes reading in R easier.

Arguments

logpdf	an lpdf class instance

Value

a vector of parameters

listcov *list all covariance functions*

Description

list all covariance functions

Usage

```
listcov()
```

Value

list all names of covariance functions recommend as of this edition. The first is the default.

loglik_gauss *Gaussian errors, large scale*

Description

```
loglik = new(loglik_gauss, om, terms, y, x)
```

This is a standard model which has the form

$$y = \langle \phi(x), \theta \rangle + \varepsilon, \varepsilon \sim N(0, \sigma^2)$$

where $\phi(x)$ is the basis, θ is the coefficient vector, ε is an unseen noise vector. The parameter vector is of length 1 where `para` $= \log(\sigma)$. It is a faster (sometimes) version of `loglik_std` but can only handle diagonal variational inference.

Arguments

`om`	an `outermod` instance to be referred to
`terms`	a matrix of `terms`, must have as many columns as dims in `om`
`y`	a vector of observations
`x`	a matrix of predictors, must have as many columns as dims in `om` and the same number of rows as `y`

Value

no returns, this is a class which contains methods

See Also

base class: `lpdf`

loglik_gda *Gaussian errors with diagonal adjustment*

Description

`loglik = new(loglik_gda, om, terms, y, x)`

This is a standard model which has the form

$$y = \langle \phi(x), \theta \rangle + \delta(x) + \varepsilon, \delta(x) \sim N(0, \lambda g(x)), \varepsilon \sim N(0, \sigma^2)$$

where $\phi(x)$ is the basis, θ is the coefficient vector, $\delta(x)$ is unseen vector corresponding to unmodeled variance $\lambda g(x)$, ε is an unseen noise vector. The parameter vector is of length 2 where $\sigma =$ `exp(para[0])` and $\lambda =$`exp(2*para[1])`.

Arguments

`om`	an `outermod` instance to be referred to
`terms`	a matrix of `terms`, must have as many columns as dims in `om`
`y`	a vector of observations
`x`	a matrix of predictors, must have as many columns as dims in `om` and the same number of rows as `y`

Value

no returns, this is a class which contains methods

See Also

base class: `lpdf`

loglik_std *Gaussian errors*

Description

`loglik = new(loglik_std, om, terms, y, x)`

This is a standard model which has the form

$$y = \langle \phi(x), \theta \rangle + \varepsilon, \varepsilon \sim N(0, \sigma^2)$$

where $\phi(x)$ is the basis, θ is the coefficient vector, ε is an unseen noise vector. The parameter vector is of length 1 where `para` $= \log(\sigma)$. It is a slower (sometimes) version of `loglik_gauss` but allows for complete marginal inference.

Arguments

om	an outermod instance to be referred to
terms	a matrix of terms, must have as many columns as dims in om
y	a vector of observations
x	a matrix of predictors, must have as many columns as dims in om and the same number of rows as y

Value

no returns, this is a class which contains methods

See Also

base class: lpdf

---

logpr_gauss *Gaussian prior*

Description

```
logpr = new(logpr_gauss, om, terms)
```

This is a standard model of coefficients which has them as drawn independently from

$$\theta_i \sim N(0, \rho c_i)$$

where c_i is the variance supplied by om for the ith term. The parameter vector is of length 1 where $\rho = \texttt{exp(para[0])}$.

Arguments

om	an outermod instance to be referred to
terms	a matrix of terms, must have as many columns as dims in om

Value

no returns, this is a class which contains methods

See Also

base class: lpdf

lpdf *Log probability density function class*

Description

This is a base class designed to handle the learning of the underlying coefficients, hyperparameters, and parameters associated with a specific learning instance. Polymorphism allows for the implied methods to be used across several similar classes.

Value

no returns, this is a class which contains methods

Fields

`lpdf$val` current value

`lpdf$para` current model parameters

`lpdf$coeff` current coefficients

`lpdf$compute_val` on calling `update`, compute value and store in `val`

`lpdf$grad` current gradient with respect to coefficients

`lpdf$gradhyp` current gradient with respect to covariance hyperparameters

`lpdf$gradpara` current gradient with respect to model parameters

`lpdf$compute_grad` on calling `update`, compute gradient with respect to coefficients and store in `grad`

`lpdf$compute_gradhyp` on calling `update`, compute gradient with respect to covariance hyperparameters and store in `gradhyp`

`lpdf$compute_gradpara` on calling `update`, compute gradient with respect to model parameters and store in `gradpara`

`lpdf$update(coeff)` update using new coefficients

`lpdf$optcg(tol,epoch)` do optimization with respect to coefficients via conjugate gradient

`lpdf$optnewton()` do optimization via matrix inversion, one Newton step

`lpdf$updateom()` update based on recent version of `outermod`

`lpdf$updatepara(para)` update using new model parameters

`lpdf$updateterms(terms)` update using new `terms`

`lpdf$hess()` returns the hessian with respect to coefficients

`lpdf$hessgradhyp()` returns gradient of `hess()` with respect to covariance hyperparameters

`lpdf$hessgradpara()` returns the gradient of `hess()` with respect to model parameters

`lpdf$diaghess()` returns the diagonal of the hessian with respect to coefficients

`lpdf$diaghessgradhyp()` returns the gradient of `diaghess()` with respect to covariance hyperparameters

`lpdf$diaghessgradpara()` returns the gradient of `diaghess()` with respect to model parameters

`lpdf$paralpdf(para)` compute the log-prior on the parameters, useful for fitting

`lpdf$paralpdf_grad(para)` gradient of `paralpdf(para)`

See Also

container class: lpdfvec

derived classes: loglik_std, loglik_gauss, loglik_gda, logpr_gauss

---

lpdfvec *Vector of* lpdf *instances*

Description

```
logpdf = new(lpdfvec, loglik, logpr)
```

This is a class where each instance contains two lpdf instances and can be manipulated as a single instance. It presumes both are based on the same outermod instance, thus they share hyperparameters. However the model parameters are concatenated. Currently also includes variations on marginal adjustments.

Currently it is designed only for a pair, but the ordering is arbitrary.

Arguments

loglik	one reference to a lpdf instance
logpr	another reference to a lpdf instance that shares outermod with loglik

Value

no returns, this is a class which contains methods

Fields

lpdfvec$domarg A boolean that controls if marginal adjustment is done

See Also

base class: lpdf

lpdf$optcg *Optimization via Conjugate Gradient*

Description

`lpdf$optcg(tol,epoch)`

This optimizes the coefficient vector `coeff` using conjugate gradient. It currently is designed only for quadratic `lpdf` instances.

Arguments

`tol`	A positive double representing tolerance, default is `0.001`.
`epoch`	A positive integer representing the maximum number of steps conjugate gradient will take.

Value

nothing is returned, the class instance is updated

See Also

`lpdf`

lpdf$optnewton *Optimization via Newton's Method*

Description

`lpdf$optnewton()`

This optimizes the coefficient vector `coeff` using Newton's Method. It currently is designed only for quadratic `lpdf` instances. It should take a single step.

Value

nothing is returned, the class instance is updated

See Also

`lpdf`

obfit *Outerbase model fit*

Description

This function fits an outerbase model for prediction and hides most of the actual object-oriented aspects of the package.

Usage

```
obfit(
  x,
  y,
  numb = 100,
  verbose = 0,
  covnames = NULL,
  hyp = NULL,
  numberopts = 2,
  nthreads = NULL
)
```

Arguments

x	a n by d sized matrix of inputs
y	a n length vector of outputs
numb	size of basis to use
verbose	0-3, how much information on optimization to print to console
covnames	a d length vector of covariance names
hyp	initial covariance hyperparameters
numberopts	number of optimizations done for hyperparameters, must be larger than 1
nthreads	number of threads used in learning

Value

Saving important model information to be used with obpred

obpred *Prediction from outerbase*

Description

This function allows for turning an obmodel into predictions with mean and variance.

Usage

```
obpred(obmodel, x)
```

Arguments

obmodel	output from obfit
x	a new m by d sized matrix of inputs

Value

A list with mean and var at new x

See Also

obfit

obtest_borehole3d *Three dim borehole example*

Description

A three dimensional Borehole function used in illustrations.

Usage

```
obtest_borehole3d(x)
```

Arguments

x	a n by 3 vector of inputs

Value

a length n vector of outputs

obtest_borehole8d *Eight dim borehole example*

Description

An eight dimensional Borehole function used in illustrations.

Usage

```
obtest_borehole8d(x)
```

Arguments

x	a n by 8 vector of inputs

Value

a length n vector of outputs

outerbase *Outer product-type basis*

Description

```
ob = new(outerbase, om, x)
```

Class that handles the basis for a given set of points `x`.

Arguments

x	a matrix of predictors, must have as many columns as dims in `om`

Value

no returns, this is a class which contains methods

Fields

`nthreads` number of threads for `omp` to use

`outerbase$getbase(k)` to get each dimensions basis functions

`outerbase$getmat(terms)` to get the basis matrix at `terms`

`outerbase$build()` to (re)build the basis instance

`outerbase$matmul(terms,a)` matrix multiply without building the basis matrix

`outerbase$tmatmul(terms,a)` transpose matrix multiply without building the basis matrix

See Also

outermod the core element that controls outerbase

Examples

```
om = new(outermod)
setcovfs(om, c("mat25", "mat25", "mat25"))
setknot(om,
         list(seq(0,1,by=0.025),seq(0,1,by=0.025),seq(0,1,by=0.025)))
x = matrix(runif(10*3),ncol=3)
ob = new(outerbase, om, x)
terms = om$selectterms(40)
basismat = ob$getmat(terms)
```

---

outerbase$build *Builds the outerbase*

Description

```
outerbase$build()
```

Build (or re-build) a basis based on the recent evaluation of outermod.

Value

nothing is returned, the class instance is updated

See Also

outerbase

---

outerbase$getbase *Get base functions*

Description

```
basis_func = outerbase$getbase(k)
```

Returns the basis for dimension k. Designed mostly for visualization.

Arguments

k	An integer from that corresponds to the dimension.

Value

a matrix of evaluated basis functions

See Also

outerbase

`outerbase$getmat` *Get basis matrix*

Description

```
basismat = outerbase$getmat(terms)
```

Returns the basis matrix for a given set of `terms`.

Arguments

`terms`	a matrix of terms

Value

a matrix of evaluated basis functions based on `terms`.

See Also

outerbase

`outerbase$matmul` *Matrix multiply*

Description

```
b = outerbase$matmul(terms, a)
```

Multiplies the basis times a vector without building the basis matrix.

Arguments

`terms`	a matrix of `terms`
`a`	a vector of length the same as the rows in `terms`

Value

a vector resulting from the matrix multiplication

See Also

outerbase

outerbase$tmatmul *Transpose Matrix multiply*

Description

```
b = outerbase$tmatmul(terms, a)
```

Multiplies the transpose of the basis times a vector without building the basis matrix.

Arguments

terms	a matrix of terms
a	a vector of length the same as the rows in outerbase

Value

a vector resulting from the matrix multiplication

See Also

outerbase

outermod *Outer product-type model*

Description

This is a class used to construct outerbase class instances. It stores key information for constructing a basis.

Value

no returns, this is a class which contains methods

Fields

outermod$updatehyp(hyp) update hyperparameters

outermod$selectterms(numterms) find best numterms terms

outermod$getvar(terms) find variances of coefficients associated with terms

See Also

outerbase the main product from an outermod

setcovfs, setknot, gethyp

Examples

```
om = new(outermod)
setcovfs(om, c("mat25", "mat25", "mat25"))
setknot(om,
         list(seq(0,1,by=0.01),seq(0,1,by=0.01),seq(0,1,by=0.01)))
terms = om$selectterms(40)
coeffvar =om$getvar(terms)
hyp = gethyp(om)
hyp[1:2] = 0.5
om$updatehyp(hyp)
coeffvar = om$getvar(terms)
```

---

outermod$getvar *Get variance of coefficients*

Description

```
coeffvar = outermod$getvar(terms)
```

Returns the variance of the coefficients associated with terms.

Arguments

terms	a matrix of terms

Value

a vector of variances of each coefficient

See Also

outermod

---

outermod$selectterms *Select optimal terms*

Description

```
terms = om$selectterms(numterms)
```

Returns the best numterms given outermod currently using maximum variance criteria.

Arguments

numterms	number of basis terms desired

Value

a matrix of terms

See Also

outermod

---

outermod$updatehyp *Update hyperparameters*

Description

outermod$updatehyp(hyp)

Updates the hyperparameters for the instance of outermod.

Arguments

hyp	A vector of hyperparameters

Value

no value is returned, the class instance is updated

See Also

outermod

---

predictor *prediction class*

Description

pred = new(predictor, loglik)

This is a base class design to allow for coherent building of predictions across multiple models. Unlike many base classes in this package, it is meant to be directly used.

Arguments

loglik	An lpdf instance, specifically that starts with loglik, to build the predictor

Value

no returns, this is a class which contains methods

Fields

`predictor$update(x)` update the current input to `x` for prediction

`predictor$mean()` return the vector of means for the prediction

`predictor$var()` return the vector of variances for the prediction

`predictor$setnthreads(k)` specifics `k` as the number of threads to use

`setcovfs` *Set covariance functions*

Description

`setcovfs(om, covnames)`

Sets the covariance functions for an outermod class instance. This is first thing one does when creating an outermod instance.

Arguments

`om`	an `outermod` instance
`covnames`	a vector of strings of the covariance functions

Value

no value is returned, `om` is updated

See Also

`outermod`

Examples

```
om = new(outermod)
setcovfs(om, c("mat25", "mat25", "mat25"))
setcovfs(om, c("mat25", "mat25pow", "mat25", "mat25ang"))
```

setknot *Set knot points*

Description

`setknot(om, knotslist)`

Sets the knot points of `om` to `knotslist` to estimate the eigenfunctions and eigenvalues. It will naturally check if the knot points have the same dimension as the covariance functions. It will also check if the knot points are within reasonable bounds for the covariance functions.

Arguments

`om`	an `outermod` instance
`knotslist`	a list of one dimensional vectors

Value

no value is returned, `om` is updated

See Also

`outermod`, `setcovfs`

Examples

```
om = new(outermod)
setcovfs(om, c("mat25", "mat25", "mat25"))
knotslist = list(seq(0,1,by=0.01),seq(0,1,by=0.01),seq(0,1,by=0.01))
setknot(om, knotslist)
```

Index