Package 'netReg'

October 21, 2020

Type Package

Title Network-Regularized Regression Models

Version 1.13.1

Maintainer Simon Dirmeier <simon.dirmeier@web.de>

Description netReg fits linear regression models using network-penalization.
Graph prior knowledge, in the form of biological networks, is being
incorporated into the loss function of the linear model.
The networks describe biological relationships such as co-regulation or dependency of the same
transcription factors/metabolites/etc. yielding a part sparse and part
smooth solution for coefficient profiles.

URL https://github.com/dirmeier/netReg

BugReports https://github.com/dirmeier/netReg/issues

Depends R (>= 3.4), tensorflow (>= 1.14.0), tfprobability (>= 0.7.0)

biocViews Software, StatisticalMethod, Regression, FeatureExtraction,
Network, GraphAndNetwork

License GPL-3

Encoding UTF-8

Suggests BiocStyle, testthat, knitr, rmarkdown, lintr, styler,
LaplacesDemon, grplasso

VignetteBuilder knitr

RoxygenNote 7.0.2

SystemRequirements C++11

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, stats, reticulate, nloptr, methods

NeedsCompilation yes

PackageStatus Deprecated

git_url https://git.bioconductor.org/packages/netReg

git_branch master

git_last_commit cb32d73

git_last_commit_date 2020-06-01

Date/Publication 2020-10-20

Author Simon Dirmeier [aut, cre]

R topics documented:

`netReg-package` *The* netReg *package*

Description

netReg is a package for generalized linear regression that includes prior graphs in the models objective function.

Details

netReg uses *Armadillo* and *TensorFlow* for fast matrix computations and optimization.

Author(s)

Simon Dirmeier | <simon.dirmeier@gmx.de>

References

Dirmeier, Simon and Fuchs, Christiane and Mueller, Nikola S and Theis, Fabian J (2018), netReg: Network-regularized linear models for biological association studies.
Bioinformatics

Abadi, Martín et al. (2016), Tensorflow: A system for large-scale machine learning.
12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)

Powell M.J.D. (2009), The BOBYQA algorithm for bound constrained optimization without derivatives.
http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf Eddelbuettel, Dirk and Sanderson, Conrad (2014), RcppArmadillo: Accelerating R with high-performance C++ linear algebra.
Computational Statistics & Data Analysis

cv.edgenet *Find the optimal shrinkage parameters for edgenet*

Description

Finds the optimal regulariztion parameters using cross-validation for edgenet. We use the BOBYQA algorithm to find the optimial regularization parameters in a cross-validation framework.

Usage

```
cv.edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = NA_real_,
  psigx = NA_real_,
  psigy = NA_real_,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian,
  optim.maxit = 100,
  optim.thresh = 0.01,
  nfolds = 10
)

## S4 method for signature 'matrix,numeric'
cv.edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = NA_real_,
  psigx = NA_real_,
  psigy = NA_real_,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian,
  optim.maxit = 100,
  optim.thresh = 0.01,
  nfolds = 10
)

## S4 method for signature 'matrix,matrix'
cv.edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
```

```
  lambda = NA_real_,
  psigx = NA_real_,
  psigy = NA_real_,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian,
  optim.maxit = 100,
  optim.thresh = 0.01,
  nfolds = 10
)
```

Arguments

X	input matrix, of dimension (n x p) where n is the number of observations and p is the number of covariables. Each row is an observation vector.
Y	output matrix, of dimension (n x q) where n is the number of observations and q is the number of response variables Each row is an observation vector.
G.X	non-negativ affinity matrix for X, of dimensions (p x p) where p is the number of covariables. Providing a graph G.X will optimize the regularization parameter psi.gx. If this is not desired just set G.X to NULL.
G.Y	non-negativ affinity matrix for Y, of dimensions (q x q) where q is the number of responses Y. Providing a graph G.Y will optimize the regularization parameter psi.gy. If this is not desired just set G.Y to NULL.
lambda	numerical shrinkage parameter for LASSO. Per default this parameter is set to NA_real_ which means that lambda is going to be estimated using cross-validation. If any numerical value for lambda is set, estimation of the optimal parameter will *not* be conducted.
psigx	numerical shrinkage parameter for graph-regularization of G.X. Per default this parameter is set to NA_real_ which means that psigx is going to be estimated in the cross-validation. If any numerical value for psigx is set, estimation of the optimal parameter will *not* be conducted.
psigy	numerical shrinkage parameter for graph-regularization of G.Y. Per default this parameter is set to NA_real_ which means that psigy is going to be estimated in the cross-validation. If any numerical value for psigy is set, estimation of the optimal parameter will *not* be conducted.
thresh	numerical threshold for the optimizer
maxit	maximum number of iterations for the optimizer (integer)
learning.rate	step size for Adam optimizer (numerical)
family	family of response, e.g. *gaussian* or *binomial*
optim.maxit	the maximum number of iterations for the optimization (integer). Usually 1e4 is a good choice.
optim.thresh	numerical threshold criterion for the optimization to stop. Usually 1e-3 is a good choice.
nfolds	the number of folds to be used - default is 10

Value

An object of class `cv.edgenet`

`parameters`	the estimated, optimal regularization parameters
`lambda`	optimal estimated value for regularization parameter lambda (or, if provided as argument, the value of the parameter)
`psigx`	optimal estimated value for regularization parameter psigx (or, if provided as argument, the value of the parameter)
`psigy`	optimal estimated value for regularization parameter psigy (or, if provided as argument, the value of the parameter)
`estimated.parameters`	names of parameters that were estimated
`family`	family used for estimated
`fit`	an `edgenet` object fitted with the optimal, estimated paramters
`call`	the call that produced the object

Examples

```
X <- matrix(rnorm(100 * 10), 100, 10)
b <- matrix(rnorm(100), 10)
G.X <- abs(rWishart(1, 10, diag(10))[, , 1])
G.Y <- abs(rWishart(1, 10, diag(10))[, , 1])
diag(G.X) <- diag(G.Y) <- 0

# estimate the parameters of a Gaussian model
Y <- X %*% b + matrix(rnorm(100 * 10), 100)

## dont use affinity matrices and estimate lambda
fit <- cv.edgenet(
  X = X, Y = Y, family = gaussian,
  maxit = 1, optim.maxit = 1
)
## only provide one matrix and estimate lambda
fit <- cv.edgenet(
  X = X, Y = Y, G.X = G.X, psigx = 1, family = gaussian,
  maxit = 1, optim.maxit = 1
)
## estimate only lambda with two matrices
fit <- cv.edgenet(
  X = X, Y = Y, G.X = G.X, G.Y, psigx = 1, psigy = 1,
  family = gaussian, maxit = 1, optim.maxit = 1
)
## estimate only psigx
fit <- cv.edgenet(
  X = X, Y = Y, G.X = G.X, G.Y, lambda = 1, psigy = 1,
  family = gaussian, maxit = 1, optim.maxit = 1
)
## estimate all parameters
fit <- cv.edgenet(
  X = X, Y = Y, G.X = G.X, G.Y,
  family = gaussian, maxit = 1, optim.maxit = 1
)
## if Y is vectorial, we cannot use an affinity matrix for Y
```

```
fit <- cv.edgenet(
  X = X, Y = Y[, 1], G.X = G.X,
  family = gaussian, maxit = 1, optim.maxit = 1
)
```

---

edgenet *Fit a graph-regularized linear regression model using edge-based regularization.*

Description

Fit a graph-regularized linear regression model using edge-penalization. The coefficients are computed using graph-prior knowledge in the form of one/two affinity matrices. Graph-regularization is an extension to previously introduced regularization techniques, such as the LASSO. See the vignette for details on the objective function of the model: `vignette("edgenet", package="netReg")`

Usage

```
edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = 1,
  psigx = 1,
  psigy = 1,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)

## S4 method for signature 'matrix,numeric'
edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = 1,
  psigx = 1,
  psigy = 1,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)

## S4 method for signature 'matrix,matrix'
edgenet(
  X,
```

```
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = 1,
  psigx = 1,
  psigy = 1,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)
```

Arguments

X	input matrix, of dimension (n x p) where n is the number of observations and p is the number of covariables. Each row is an observation vector.
Y	output matrix, of dimension (n x q) where n is the number of observations and q is the number of response variables. Each row is an observation vector.
G.X	non-negativ affinity matrix for X, of dimensions (p x p) where p is the number of covariables
G.Y	non-negativ affinity matrix for Y, of dimensions (q x q) where q is the number of responses
lambda	numerical shrinkage parameter for LASSO.
psigx	numerical shrinkage parameter for graph-regularization of G.X
psigy	numerical shrinkage parameter for graph-regularization of G.Y
thresh	numerical threshold for optimizer
maxit	maximum number of iterations for optimizer (integer)
learning.rate	step size for Adam optimizer (numerical)
family	family of response, e.g. *gaussian* or *binomial*

Value

An object of class edgenet

beta	the estimated (p x q)-dimensional coefficient matrix B.hat
alpha	the estimated (q x 1)-dimensional vector of intercepts
parameters	regularization parameters
lambda	regularization parameter lambda)
psigx	regularization parameter psigx
psigy	regularization parameter psigy
family	a description of the error distribution and link function to be used. Can be a netReg::family function or a character string naming a family function, e.g. gaussian or "gaussian".
call	the call that produced the object

References

Cheng, Wei and Zhang, Xiang and Guo, Zhishan and Shi, Yu and Wang, Wei (2014), Graph-regularized dual Lasso for robust eQTL mapping.
Bioinformatics

Examples

```
X <- matrix(rnorm(100 * 10), 100, 10)
b <- matrix(rnorm(100), 10)
G.X <- abs(rWishart(1, 10, diag(10))[, , 1])
G.Y <- abs(rWishart(1, 10, diag(10))[, , 1])
diag(G.X) <- diag(G.Y) <- 0

# estimate the parameters of a Gaussian model
Y <- X %*% b + matrix(rnorm(100 * 10), 100)
## dont use affinity matrices
fit <- edgenet(X = X, Y = Y, family = gaussian, maxit = 10)
## only provide one matrix
fit <- edgenet(X = X, Y = Y, G.X = G.X, psigx = 1, family = gaussian, maxit = 10)
## use two matrices
fit <- edgenet(X = X, Y = Y, G.X = G.X, G.Y, family = gaussian, maxit = 10)
## if Y is vectorial, we cannot use an affinity matrix for Y
fit <- edgenet(X = X, Y = Y[, 1], G.X = G.X, family = gaussian, maxit = 10)
```

`family` *Family objects for models*

Description

Family objects provide a convenient way to specify the details of the models used by `netReg`. See also `stats::family` for more details.

Usage

```
family(object, ...)

gaussian(link = c("identity"))

binomial(link = c("logit", "probit", "log"))

poisson(link = c("log"))

gamma(link = c("inverse"))

beta(link = c("logit", "probit", "log"))

inverse.gaussian(link = c("1/mu^2"))
```

Arguments

`object`	a object for which the family shoulr be retured (e.g. `edgenet`)
`...`	further arguments passed to methods
`link`	name of a link function

Value

An object of class `netReg.family`

`family`	name of the family
`link`	name of the link function
`linkinv`	inverse link function
`loss`	loss function

Examples

```
gaussian()
binomial("probit")$link
poisson()$linkinv
gamma()$linkinv
beta()$loss
inverse.gaussian()$loss
```

`group.lasso` *Fit a linear regression model the group lasso penalty*

Description

Fit a linear regression model the group LASSO penalty.

Usage

```
group.lasso(
  X,
  Y,
  grps = NULL,
  lambda = 1,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)

## S4 method for signature 'matrix,numeric'
group.lasso(
  X,
  Y,
  grps = NULL,
  lambda = 1,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)

## S4 method for signature 'matrix,matrix'
```

```
group.lasso(
  X,
  Y,
  grps = NULL,
  lambda = 1,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)
```

Arguments

X	input matrix, of dimension (n x p) where n is the number of observations and p is the number of covariables. Each row is an observation vector.
Y	output matrix, of dimension (n x q) where n is the number of observations and q is the number of response variables. Each row is an observation vector.
grps	vector of integers or NA_integer_ of length p that encodes the grouping of variables, e.g., c(1,1,2,2,NA)
lambda	numerical shrinkage parameter
thresh	numerical threshold for optimizer
maxit	maximum number of iterations for optimizer (integer)
learning.rate	step size for Adam optimizer (numerical)
family	family of response, e.g., *gaussian* or *binomial*

Value

An object of class edgenet

beta	the estimated (p x q)-dimensional coefficient matrix B.hat
alpha	the estimated (q x 1)-dimensional vector of intercepts
parameters	regularization parameters
lambda	regularization parameter lambda)
family	a description of the error distribution and link function to be used. Can be a netReg::family function or a character string naming a family function, e.g. gaussian or "gaussian".
call	the call that produced the object

References

Yuan, Ming and Lin, Yi (2006), Model selection and estimation in regression with grouped variables.
Journal of the Royal Statistical Society: Series B

Meier, Lukas and Van De Geer, Sara and Bühlmann, Peter (2008), The group lasso for logistic regression.
Journal of the Royal Statistical Society: Series B

Examples

```
X <- matrix(rnorm(100 * 10), 100, 5)
b <- rnorm(5)
grps <- c(NA_integer_, 1L, 1L, 2L, 2L)

# estimate the parameters of a Gaussian model
Y <- X %*% b + rnorm(100)
fit <- group.lasso(X = X, Y = Y, grps = grps, family = gaussian, maxit = 10)
```

---

yeast *A sample yeast data set for regression*

Description

The yeast data set is a `list` containing three matrices that can be used as an example for using `netReg`. The data have been taken from the references listed below.

Usage

```
data(yeast)
```

Format

A `list` containing three matrices

Details

- `X` (112 x 500)-dimensional binary matrix of 500 genetic markers for 112 yeast samples
- `Y` (112 x 231)-dimensional double matrix of 231 gene expression values for 112 yeast samples
- `GY` (231 x 231)-dimensional adjaceny matrix representing protein-protein interactions for 231 yeast genes

References

Brem, Rachel B., et al. (2005), Genetic interactions between polymorphisms that affect gene expression in yeast.
Nature

Storey, John D., Joshua M. Akey, and Leonid Kruglyak (2005), Multiple locus linkage analysis of genomewide expression in yeast.
PLoS Biology

Index