Package 'graper'

October 16, 2019

Type Package

Title Adaptive penalization in high-dimensional regression and classification with external covariates using variational Bayes

Version 1.0.0

Date 2018-10-26

License GPL (>= 2)

Description This package enables regression and classification on high-dimensional data with different relative strengths of penalization for different feature groups, such as different assays or omic types. The optimal relative strengths are chosen adaptively. Optimisation is performed using a variational Bayes approach.

Depends R (>= 3.6)

Encoding UTF-8

LazyData TRUE

Imports Matrix, Rcpp, stats, ggplot2, methods, cowplot, matrixStats

LinkingTo Rcpp, RcppArmadillo, BH

biocViews Regression, Bayesian, Classification

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, BiocStyle, testthat

VignetteBuilder knitr

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

git_branch RELEASE_3_9

git_last_commit cc518e1

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

Author Britta Velten [aut, cre], Wolfgang Huber [aut]

Maintainer Britta Velten <britta.velten@embl.de>

R topics documented:

`coef.graper` *Get estimated coefficients from a graper object*

Description

Function to obtain estimated coefficients from a fitted graper model.

Usage

```
## S3 method for class 'graper'
coef(object, include_intercept = TRUE, ...)
```

Arguments

`object`	fitted graper model as obtained from `graper`
`include_intercept`	whether to include the estimated intercept value in the output
`...`	other arguments

Value

1-Column matrix of estimated coefficients.

Examples

```
# create data
dat <- makeExampleData()
fit <- graper(dat$X, dat$y, dat$annot)
coef(fit)
```

getPIPs *Get posterior inclusion probabilities per feature*

Description

Function to obtain estimated posterior inclusion probabilities per feature from a fitted graper model.

Usage

```
getPIPs(object)
```

Arguments

object	fitted graper model as obtained from graper

Value

1-Column matrix of estimated posterior inclusion probabilities.

Examples

```
# create data
dat <- makeExampleData()
fit <- graper(dat$X, dat$y, dat$annot)
getPIPs(fit)
```

graper *Fit a regression model with graper*

Description

Fit a regression model with graper given a matrix of predictors (X), a response vector (y) and a vector of group memberships for each predictor in X (annot). For each group a different strength of penalization is determined adaptively.

Usage

```
graper(X, y, annot, factoriseQ = TRUE, spikeslab = TRUE,
    intercept = TRUE, family = "gaussian", standardize = TRUE,
    n_rep = 1, max_iter = 3000, th = 0.01, d_tau = 0.001,
    r_tau = 0.001, d_gamma = 0.001, r_gamma = 0.001, r_pi = 1,
    d_pi = 1, calcELB = TRUE, verbose = TRUE, freqELB = 1,
    nogamma = FALSE, init_psi = 1)
```

Arguments

X	design matrix of size n (samples) x p (features)
y	response vector of size n
annot	factor of length p indicating group membership of each feature (column) in X
factoriseQ	if set to TRUE, the variational distribution is assumed to fully factorize across features (faster, default). If FALSE, a multivariate variational distribution is used.
spikeslab	if set to TRUE, a spike and slab prior on the coefficients (default).
intercept	whether to include an intercept into the model
family	Likelihood model for the response, either "gaussian" for linear regression or "binomial" for logistic regression
standardize	whether to standardize the predictors to unit variance
n_rep	number of repetitions with different random initializations to be fit
max_iter	maximum number of iterations
th	convergence threshold for the evidence lower bound (ELB)
d_tau	hyper-parameters for prior of tau (noise precision)
r_tau	hyper-parameters for prior of tau (noise precision)
d_gamma	hyper-parameters for prior of gamma (coefficients' prior precision)
r_gamma	hyper-parameters for prior of gamma (coefficients' prior precision)
r_pi	hyper-parameters for Beta prior of the mixture probabilities in the spike and slab prior
d_pi	hyper-parameters for Beta prior of the mixture probabilities in the spike and slab prior
calcELB	whether to calculate the evidence lower bound (ELB)
verbose	whether to print out intermediate messages during fitting
freqELB	frequency at which the evidence lower bound (ELB) is to be calculated, i.e. each freqELB-th iteration
nogamma	if TRUE, the normal prior will have same variance for all groups (only relevant for spikeslab = TRUE)
init_psi	initial value for the spike variables

Details

The function trains the graper model given a matrix of predictors (X), a response vector (y) and a vector of group memberships for each predictor in X (annot). For each feature group as specified in annot a penalty factor and sparsity level is learnt.

By default it uses a Spike-and-Slab prior on the coefficients and uses a fully factorized variational distribution in the inference. This provides a fast way to train the model. Using spikeslab=FALSE a ridge regression like model can be fitted using a normal instead of the spike and slab prior. Setting factoriseQ = FALSE gives a more exact inference scheme based on a multivariate variational distribution, but can be much slower.

As the optimization is non-convex is can be helpful to use multiple random initializations by setting n_rep to a value larger 1. The returned model is then chosen as the optimal fit with respect to the evidence lower bound (ELB).

Depending on the response vector a linear regression model (family = "gaussian") or a logistic regression model (family = "binomial") is fitted. Note, that the implementation of logistic regression is still experimental.

Value

A graper object containing

EW_beta estimated model coefficientsin liner/logistic regression

EW_s estimated posterior-inclusion probabilities for each feature

intercept estimated intercept term

annot annotation vector of features to the groups as specified when calling graper

EW_gamma estimated penalty factor per group

EW_pi estimated sparsity level per group (from 1 (dense) to 0 (sparse))

EW_tau estimated noise precision

sigma2_tildebeta_s1, EW_tildebeta_s1, alpha_gamma, alpha_tau, beta_tau, Sigma_beta, alpha_pi, beta_pi parameters of the variational distributions of beta, gamma, tau and pi

ELB final value of the evidence lower bound

ELB_trace values of the evidence lower bound for all iterations

Options other options used when calling graper

Examples

```
# create data
dat <- makeExampleData()

# fit a sparse model with spike and slab prior
fit <- graper(dat$X, dat$y, dat$annot)
fit # print fitted object
beta <- coef(fit, include_intercept = FALSE) # model coeffients
pips <- getPIPs(fit) # posterior inclusion probabilities
pf <- fit$EW_gamma # penalty factors per group
sparsities <- fit$EW_pi # sparsity levels per group

# fit a dense model without spike and slab prior
fit <- graper(dat$X, dat$y, dat$annot, spikeslab = FALSE)

# fit a dense model using a multivariate variational distribution
fit <- graper(dat$X, dat$y, dat$annot, factoriseQ = TRUE, spikeslab = FALSE)
```

---

makeExampleData *Simulate example data from the graper model*

Description

Simulate data from the graper model with groups of equal size and pre-specified parameters gamma, pi and tau.

Usage

```
makeExampleData(n = 100, p = 200, g = 4, gammas = c(0.1, 1, 10,
    100), pis = c(0.5, 0.5, 0.5, 0.5), tau = 1, rho = 0,
    response = "gaussian", intercept = 0)
```

Arguments

n	number of samples
p	number of features
g	number of groups
gammas	vector of length g, specifying the slab precision of the prior on beta per group
pis	vector of length g, specifying the probability of s to be 1 (slab)
tau	noise precision
rho	correlation of design matrix (Toeplitz structure)
response	"gaussian" for continuous response from a linear regression model, "bernoulli" for a binary response from a logistic regression model.
intercept	model intercept (default: 0)

Value

list containing the design matrix X, the response y, the feature annotation to groups annot as well as the different parameters in the Bayesian model and the correlation strength rho

Examples

```
dat <- makeExampleData()
```

---

makeExampleDataWithUnequalGroups

Simulate example data from the graper model with groups of unequal size

---

Description

Simulate data from the graper model with groups of unequal size and pre-specified parameters gamma, pi and tau.

Usage

```
makeExampleDataWithUnequalGroups(n = 100, pg = c(100, 100, 10, 10),
    gammas = c(0.1, 10, 0.1, 10), pis = c(0.5, 0.5, 0.5, 0.5), tau = 1,
    rho = 0, response = "gaussian", intercept = 0)
```

Arguments

n	number of samples
pg	vector of length g (desired number of groups) with number of features per group
gammas	vector of length g, specifying the slab precision of the prior on beta per group
pis	vector of length g, specifying the probability of s to be 1 (slab)
tau	noise precision (only relevant for gaussian response)
rho	correlation of design matrix (Toeplitz structure)
response	"gaussian" for continuous response from a linear regression model, "bernoulli" for a binary response from a logistic regression model.
intercept	model intercept (default: 0)

Value

list containin the design matrix `X`, the response `y`, the feature annotation to groups `annot` as well as the different parameters in the Bayesian model and the correlation strength rho

Examples

```
dat <- makeExampleDataWithUnequalGroups()
```

`plotELBO` *Plot evidence lower bound*

Description

Function to plot the evidence lower bound (ELBO) over iterations to monitor the convergence of the algorithm.

Usage

```
plotELBO(fit)
```

Arguments

`fit`	fit as produced by `graper`

Value

a ggplot object

Examples

```
dat <- makeExampleData()
fit <- graper(dat$X, dat$y, dat$annot)
plotELBO(fit)
```

`plotGroupPenalties` *Plot group-wise penalties*

Description

Function to plot the group-wise penalty factors (gamma) and sparsity levels.

Usage

```
plotGroupPenalties(fit)
```

Arguments

`fit`	fit as produced by `graper`

Value

a ggplot object

Examples

```
dat <- makeExampleData()
fit <- graper(dat$X, dat$y, dat$annot)
plotGroupPenalties(fit)
```

`plotPosterior` *Plot posterior distributions*

Description

Function to plot the posterior of the model parameters obtained by graper from the variational inference framework.

Usage

```
plotPosterior(fit, param2plot, beta0 = NULL, gamma0 = NULL,
    tau0 = NULL, pi0 = NULL, s0 = NULL, jmax = 2, range = NULL)
```

Arguments

`fit`	fit as produced by `graper`
`param2plot`	which parameter of the graper model to plot (gamma, beta, tau or s)
`beta0`	true beta (if known)
`gamma0`	true gamma (if known)
`tau0`	true tau (if known)
`pi0`	true pi (if known)
`s0`	true s (if known)
`jmax`	maximal number of components per group to plot (for beta and s)
`range`	plotting range (x-axis)

Value

a ggplot object

Examples

```
dat <- makeExampleData()
fit <- graper(dat$X, dat$y, dat$annot)
plotPosterior(fit, param2plot="gamma")
```

predict.graper *Predict response on new data*

Description

Function to predict the response on a new data set using a fitted graper model.

Usage

```
## S3 method for class 'graper'
predict(object, newX, type = c("inRange", "response",
"link"), ...)
```

Arguments

object	fitted graper model as obtained from graper
newX	Predictor matrix of size n_test (number of new test samples) x p (number of predictors) (same feature structure as used in graper)
type	type of prediction returned, either: • **response**:returns the linear predictions for linear regression and class probabilities for logistic regression • **link**:returns the linear predictions • **inRange**:returns linear predictions for linear and class memberships for logistic regression
...	other arguments

Value

A vector with predictions.

Examples

```
# create data
dat <- makeExampleData()
ntrain <- dat$n/2
fit <- graper(dat$X[seq_len(ntrain),],
dat$y[seq_len(ntrain)], dat$annot)
ypred <- predict(fit, dat$X[seq_len(ntrain) + dat$n/2,])

dat <- makeExampleData(response="bernoulli")
ntrain <- dat$n/2
fit <- graper(dat$X[seq_len(ntrain),],
dat$y[seq_len(ntrain)], dat$annot, family = "binomial")
ypred <- predict(fit, dat$X[seq_len(ntrain) + dat$n/2,])
```

`print.graper` *Print a graper object*

Description

Function to print a fitted graper model.

Usage

```
## S3 method for class 'graper'
print(x, ...)
```

Arguments

`x`	fitted graper model as obtained from `graper`
`...`	additional print arguments

Value

Print output.

Examples

```
# create data
dat <- makeExampleData()
fit <- graper(dat$X, dat$y, dat$annot)
print(fit)
```

Index