Package ‘gren’

July 30, 2018

Type Package
Title Adaptive Group-Regularized Logistic Elastic Net Regression
Version 0.0.1
Date 2018-07-19
Author Magnus M. Münch
Maintainer Magnus M. Münch <m.munch@vumc.nl>
Description Allows the user to incorporate multiple sources of co-data (e.g., previously obtained p-values, published gene lists, and annotation) in the estimation of a logistic regression model to enhance predictive performance and feature selection, as described in Münch, Peeters, van der Vaart, and van de Wiel (2018) <arXiv:1805.00389>.
License GPL (>= 2)
Imports Rcpp (>= 0.12.14), glmnet, Iso, pROC
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
VignetteBuilder knitr
URL https://github.com/magnusmunch/gren/
Suggests knitr, rmarkdown
Encoding UTF-8
RoxygenNote 6.0.1
LazyData true
Repository CRAN
Date/Publication 2018-07-30 13:10:03 UTC

R topics documented:

  gren-package .................................................. 2
cv.gren .......................................................... 3
dataCervical ...................................................... 5
dataColon ......................................................... 6
Adaptive group-regularized logistic elastic net regression

Description

The package allows the user to incorporate multiple sources of co-data (e.g., previously obtained p-values, published gene lists, and annotation) in the estimation of a logistic regression model to enhance predictive performance.

Details

The main function of the package is `gren`, which estimates a group-regularized elastic net regression model. The following functions are convenience functions:

- `cv.gren` estimates performance measures by efficient cross-validation.
- `coef.gren` S3 method to retrieve model parameters from a `gren` fit.
- `predict.gren` S3 method to get predictions for new data from a `gren` fit.
- `denet` density function of the elastic net prior distribution.
- `renet` generate samples from the elastic net prior distribution.

Author(s)

Magnus M. Münch Maintainer: Magnus M. Münch <m.munch@vumc.nl>

References


See Also

- `cv.gren`

Examples

```r
## Create data
p <- 1000
n <- 100
set.seed(2018)
x <- matrix(rnorm(n*p), ncol=p, nrow=n)
beta <- c(rnorm(p/2, 0, 0.1), rnorm(p/2, 0, 1))
```
m <- rep(1, n)
y <- rbinom(n, m, as.numeric(1/(1 + exp(-x %*% as.matrix(beta)))))
partitions <- list(groups=rep(c(1, 2), each=p/2))

## estimate model
fit.gren <- gren(x, y, m, partitions=partitions)

---

**cv.gren**

*Performance cross-validation of group-regularized logistic elastic net regression*

**Description**

Function that cross-validations for performance estimation of gren models.

**Usage**

```r
cv.gren(x, y, m=rep(1, nrow(x)), unpenalized=NULL, partitions=NULL, alpha=0.5, lambda=NULL, intercept=TRUE, monotone=NULL, psel=TRUE, compare=TRUE, posterior=FALSE, nfolds=nrow(x), foldid=NULL, trace=TRUE, control=list(epsilon=0.001, maxit=500, maxit.opt=1000, maxit.vb=100), keep.pred=TRUE, fix.lambda=FALSE, nfolds.out=nrow(x), foldid.out=NULL, type.measure=c("auc", "deviance", "class.error"))
```

**Arguments**

- **x**: See gren.
- **y**: See gren.
- **m**: See gren.
- **unpenalized**: See gren.
- **partitions**: See gren.
- **alpha**: See gren.
- **lambda**: See gren.
- **intercept**: See gren.
- **monotone**: See gren.
- **psel**: See gren.
- **compare**: See gren.
- **posterior**: See gren.
- **nfolds**: See gren.
- **foldid**: See gren.
- **trace**: if TRUE, progress of the cross-validation is printed.
- **control**: See gren.
cv.gren

keep.pred logical. If TRUE the cross-validated predictions are saved.
fix.lambda logical. If TRUE lambda is cross-validated only once and used in every fold.
nfolds.out numeric that gives the number of folds to use in the performance cross-validation. Default is nrow(x).
foldid.out optional numeric vector of length nrow(x) with the performance cross-validation fold assignments of the observations.
type.measure numeric that gives the performance measures to calculate. Currently possible are AUC, deviance, and misclassification error.

Details

cv.gren is a convenience function that gives cross-validated predictions. Performance measures are optionally calculated with these predictions. cv.gren is more efficient than simply looping over the folds, since it uses the final estimates of the previous fold as starting values for the next fold. This substantially reduces computation time.

Value

Function returns a list of length two with the following components:

- groupreg list with the cross-validated predictions of length nrow(x) and possibly performance metrics of the group-regularized model.
- regular list with the cross-validated predictions of length nrow(x) and possibly performance metrics of the regular model.

Author(s)

Magnus M. Münch <m.munch@vumc.nl>

References


See Also

gren, predict.gren, coef.gren

Examples

```r
## Create data
p <- 1000
n <- 100
set.seed(2018)
x <- matrix(rnorm(n*p), ncol=p, nrow=n)
beta <- c(rnorm(p/2, 0, 0.1), rnorm(p/2, 0, 1))
m <- rep(1, n)
y <- rbinom(n, m, as.numeric(1/(1 + exp(-x %*% as.matrix(beta)))))
```
```r
partitions <- list(groups=rep(c(1, 2), each=p/2))

## calculate cross-validated predictions and performance measures
fit.cv.gren <- cv.gren(x, y, m, partitions=partitions, fix.lambda=TRUE)
```

**Description**

The three objects are: mirCerv: 772 sequenced microRNAs for 56 samples; respCerv: binary response coded as healthy and pre-cursor lesions for cervical cancer; and mirCons: conservation status of the microRNAs in three levels coded as only found in humans, found in most mammals, and found in most vertebrates.

**Usage**

```r
data(dataCervical)
```

**Format**

The formats are:

- **mirCerv**: matrix [1:56, 1:772]
- **respCerv**: Factor w/ 2 levels "CIN3","Normal"
- **mirCons**: Factor w/ 3 levels "NotCons", "Mammals", "Broadly"

**Details**

This data is used for illustration in the vignette of the `gren` package.

**Source**


**References**


**Examples**

```r
data(dataCervical)
str(mirCerv)
str(respCerv)
str(mirCons)
```
**dataColon**

Contains four R-objects, including the data and the binary response

---

**Description**

The four objects are: mirCol: 2114 sequenced microRNAs for 88 samples; unpenCol: 4 clinical covariates for 88 samples; respCol: binary response coded as progressive disease and benefitted from therapy; mirExpr: expression levels of microRNAs from previous study coded as not differentially expressed, medium significantly expressed, and highly significantly expressed.

**Usage**

data(dataColon)

**Format**

The formats are:

- **mirCol** matrix [1:88, 1:2114]
- **unpenCol** data.frame: 88 obs. of 4 variables
- **respCol** Factor w/ 2 levels "Progr","TherBenefit"
- **mirExpr** Factor w/ 3 levels "nonExpr", "medExpr", "highExpr"

**Details**

This data is used for illustration in the vignette of the grent package and the corresponding paper (see references).

**Source**


**References**

Examples

```r
data(dataColon)
str(mirCol)
str(unpenCol)
str(respCol)
str(mirexpr)
```

---

**denet**

*The elastic net prior distribution*

---

**Description**

Density function and random number generator for the elastic net prior distribution.

**Usage**

```r
denet(x, lambda1=1, lambda2=1, log=FALSE)
renet(n, lambda1=1, lambda2=1)
```

**Arguments**

- `x` : vector of quantiles
- `n` : number of samples
- `lambda1` : lambda1 parameter value
- `lambda2` : lambda2 parameter value
- `log` : should the logarithm of the density be returned

**Details**

The elastic net prior density has density:

\[
f(x) = g(\lambda_1, \lambda_2)e^{\left[-0.5 \ast (\lambda_1|x| + \lambda_2x^2)\right]}
\]

**Value**

- `denet` gives the density of the input `x`
- `renet` gives a vector of length `n` of random values.

**Author(s)**

Magnus M. Münch <m.munch@vumc.nl>

**References**

Examples

```r
## Create data
n <- 100
x <- renet(n)
hist(x)

## Calculate density
dens <- denet(x)
plot(sort(x), dens[order(x)])
```

---

**gren**

*Group-regularized logistic elastic net regression*

---

**Description**

Function that estimates a group-regularized elastic net model.

**Usage**

```r
gren(x, y, m=rep(1, nrow(x)), unpenalized=NULL, partitions=NULL, alpha=0.5, lambda=NULL, intercept=TRUE, monotone=NULL, psel=TRUE, compare=TRUE, posterior=FALSE, nfolds=nrow(x), foldid=NULL, trace=TRUE, init=list(lambdag=NULL, mu=NULL, sigma=NULL, chi=NULL, ci=NULL), control=list(epsilon=0.001, maxit=500, maxit.opt=1000, maxit.vb=100))
```

**Arguments**

- **x**: feature data as either numeric matrix or data.frame of numeric variables.
- **y**: response as either a numeric with binomial/binary successes of length nrow(x) or a matrix of nrow(x) rows and two columns, where the first column contains the binomial/binary failures and the second column the binomial/binary successes.
- **m**: numeric of length nrow(x) that contains the number of Bernoulli trials.
- **unpenalized**: Optional numeric matrix or data.frame of numeric unpenalized covariates of nrow(x) rows.
- **partitions**: list that contains the (possibly multiple) partitions of the data. Every list object corresponds to one partition, where every partition is a numeric of length ncol(x) containing the group ids of the features.
- **alpha**: proportion of L1 penalty as a numeric of length 1.
- **lambda**: global penalty parameter. The default NULL will result in estimation by cross-validation.
- **intercept**: logical to indicate whether an intercept should be included.
- **monotone**: list of two logical vectors of length length(partitions). The first one indicates whether the corresponding partition’s penalty parameters should be monotonically estimates, the second vector decreasing indicates whether the monotone penalty parameters are decreasing with group number.
psel either a numeric vector that indicates the number of features to select or a logical. If TRUE feature selection is done by letting glmnet determine the penalty parameter sequence.

compare logical, if TRUE, a regular non-group-regularized model is estimated.

posterior if TRUE, the full variational Bayes posterior is returned.

nfolds numeric of length 1 with the number of folds used in the cross-validation of the global lambda. The default is nrow(x).

foldid optional numeric vector of length nrow(x) with the fold assignments of the observations.

trace if TRUE, progress of the algorithm is printed.

init optional list containing the starting values of the iterative algorithm. See Details for more information.

control a list of algorithm control parameters. See Details for more information.

Details

This is the main function of the package that estimates a group-regularized elastic net regression. The elastic net penalty's proportion of L1-norm penalisation is determined by alpha. Alpha close to 0 implies more ridge-like penalty, while alpha close to 1 implies lasso-like penalty. The algorithm is a two-step procedure: first, a global lambda penalty is estimates by cross-validation. Next, the groupwise lambda multipliers are estimates by an EM algorithm. The EM algorithm consists of: i) an expectation step in which the expected marginal likelihood of the penalty multipliers is iteratively approximated by a variational Bayes EM algorithm and ii) a maximisation step in which the approximate expected marginal likelihood is maximised with respect to the penalty multipliers. After convergence of the algorithm an (optional) frequentist elastic net model is fit using the estimated penalty multipliers by setting psel=TRUE or by setting psel to a numeric vector.

The user may speed up the procedure by specifying initial values for the EM algorithm in init. init is a list that contains:

lambdag initial values for $\lambda_g$ in a list of length length(partitions).

mu initial values for the $\mu_j$ in a numeric vector of length ncol(x) + ncol(unpenalized) + intercept.

chi initial values for the $\chi_j$ in a numeric vector of length ncol(x).

ci initial values for the $c_i$ in a numeric vector of length nrow(x).

sigma The initial values for the $\Sigma_{ij}$ in a matrix of numerics with ncol(x) rows and columns.

control is a list with parameters to control the estimation procedure. It consists of the following components:

epsilon numeric with the relative convergence tolerance. Default is epsilon=0.001.

maxit numeric with whole number that gives the maximum number of iterations to update the lambdag. Default is maxit=500.

maxit.opt numeric with whole number that gives the maximum number of iterations to numerically maximise the Lambdag. Maximisation occurs at every iteration. Default is maxit.opt=1000.

maxit.vb numeric with whole number that gives the maximum number of iterations to update the variational parameters mu, sigma, chi, and ci. One full update sequence per iteration. Default is maxit=100.
Value

Function returns an S3 list object of class `gren` containing output with the following components:

- **call**: The function call that produced the output.
- **alpha**: Proportion of L1 penalty as a numeric of length 1.
- **lambda**: Global penalty parameter as numeric. Estimated by cross-validation if `lambda=NULL`.
- **lambdag.seq**: List with full sequence of penalty multipliers over iterations.
- **lambdag**: List with final estimates of penalty multipliers.
- **vb.post**: List with variational posterior parameters $\mu_j$, $\sigma_{ij}$, $c_i$, and $\chi_j$.
- **freq.model**: Frequentist elastic net model as output of `glmnet` call. `NULL` if `p sel=FALSE`.
- **iter**: List with number of iterations of `lambdag` estimation, with number of optimisation iterations of `lambdag`, and number of variational Bayes iterations.
- **conv**: List of logicals with convergence of `lambdag` sequence, optimisation steps, and variational Bayes iterations.
- **args**: List with input arguments of `gren` call.

Author(s)

Magnus M. Münch <m.munch@vumc.nl>

References


See Also

`predict.gren`, `coef.gren`, `cv.gren`

Examples

```r
## Create data
p <- 1000
n <- 100
set.seed(2018)
x <- matrix(rnorm(n*p), ncol=p, nrow=n)
beta <- c(rnorm(p/2, 0, 0.1), rnorm(p/2, 0, 1))
m <- rep(1, n)
y <- rbinom(n, m, as.numeric(1/(1 + exp(-x %*% as.matrix(beta)))))
partitions <- list(groups=rep(c(1, 2), each=p/2))

## estimate model
fit.gren <- gren(x, y, m, partitions=partitions)
```
predict.gren

Predictions and coefficients from gren model.

Description

Create predictions from new data using a fitted gren model/retrieve coefficients from fitted model. Both are S3 methods.

Usage

```r
## S3 method for class 'gren'
predict(object, newx, unpenalized=NULL, s=NULL,
        type=c("groupreg", "regular"), ...)

## S3 method for class 'gren'
coef(object, s=NULL, type=c("groupreg", "regular"), ...)
```

Arguments

- `object`: A fitted gren model.
- `newx`: New data for which to do predictions.
- `unpenalized`: New unpenalized data for which to do predictions.
- `s`: Value of lambda for which to create predictions/coefficients, may be a vector.
- `type`: Either groupreg, which creates predictions/coefficients of group-regularized model, or regular, for predictions/coefficients from regular model.
- `...`: Further arguments to be passed.

Details

This are the predict/coefficient functions of the gren package.

Value

- `predict`: returns a numeric matrix with predicted probabilities.
- `coef`: returns a matrix with coefficients.

Author(s)

Magnus M. Münch <m.munch@vumc.nl>

References

See Also

gren, coef.gren

Examples

```r
## Create data
p <- 1000
n <- 100
set.seed(2018)
x <- matrix(rnorm(n*p), ncol=p, nrow=n)
beta <- c(rnorm(p/2, 0, 0.1), rnorm(p/2, 0, 1))
m <- rep(1, n)
y <- rbinom(n, m, as.numeric(1/(1 + exp(-x %*% as.matrix(beta)))))
partitions <- list(groups=rep(c(1, 2), each=p/2))

## Estimate model
fit.gren <- gren(x, y, m, partitions=partitions)

## Create new data
xnew <- matrix(rnorm(n*p), ncol=p, nrow=n)

## Create predictions/coefficients
preds <- predict(fit.gren, xnew, type="groupreg")
coefs <- coef(fit.gren, type="groupreg")
```
Index

*Topic **datasets**
  dataCervical, 5
  dataCol, 6

*Topic **package**
  gren-package, 2

coef.gren, 2, 4, 10, 12
coef.gren (predict.gren), 11
cv.gren, 2, 3, 10

dataCervical, 5
dataCol, 6
denet, 2, 7

glmnet, 9
gren, 2–6, 8, 12
gren-package, 2

mirCerv (dataCervical), 5
mirCol (dataCol), 6
mirCons (dataCervical), 5
mirExpr (dataCol), 6

predict.gren, 2, 4, 10, 11

renet, 2
renet (denet), 7
respCerv (dataCervical), 5
respCol (dataCol), 6

unpenCol (dataCol), 6