Package 'dglars'

October 9, 2023

Type Package

Title Differential Geometric Least Angle Regression

Version 2.1.7

Date 2023-10-08

Author Luigi Augugliaro [aut, cre],
Angelo Mineo [aut],
Ernst Wit [aut],
Hassan Pazira [aut],
Michael Wichura [ctb, cph],
John Burkardt [ctb, cph]

Maintainer Luigi Augugliaro <luigi.augugliaro@unipa.it>

Imports methods

Description Differential geometric least angle regression method for fitting sparse generalized linear models. In this version of the package, the user can fit models specifying Gaussian, Poisson, Binomial, Gamma and Inverse Gaussian family. Furthermore, several link functions can be used to model the relationship between the conditional expected value of the response variable and the linear predictor. The solution curve can be computed using an efficient predictor-corrector or a cyclic coordinate descent algorithm, as described in the paper linked to via the URL below.

License GPL (>= 2)

URL https://www.jstatsoft.org/v59/i08/.

LazyLoad yes

Depends Matrix, R (>= 3.2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-10-09 17:30:02 UTC

R topics documented:

`dglars-package` *Differential Geometric Least Angle Regression*

Description

Differential geometric least angle regression method for fitting sparse generalized linear models. In this version of the package, the user can fit models specifying Gaussian, Poisson, Binomial, Gamma and Inverse Gaussian family. Furthermore, several link functions can be used to model the relationship between the conditional expected value of the response variable and the linear predictor. The solution curve can be computed using an efficient predictor-corrector or a cyclic coordinate descent algorithm, as described in the paper linked to via the URL below.

Details

Package:	dglars
Type:	Package
Version:	2.1.7
Date:	2023-10-08
License:	GPL (>=2)

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References

Augugliaro L., Mineo A.M. and Wit E.C. (2016) <doi:10.1093/biomet/asw023> *A differential-geometric approach to generalized linear models with grouped predictors*, Vol 103(3), 563-577.

Augugliaro L., Mineo A.M. and Wit E.C. (2014) <doi:10.18637/jss.v059.i08> *dglars: An R Package to Estimate Sparse Generalized Linear Models*, *Journal of Statistical Software*, Vol 59(8), 1-40. https://www.jstatsoft.org/v59/i08/.

Augugliaro L., Mineo A.M. and Wit E.C. (2013) <doi:10.1111/rssb.12000> *dgLARS: a differential geometric approach to sparse generalized linear models*, *Journal of the Royal Statistical Society. Series B.*, Vol 75(3), 471-498.

Efron B., Hastie T., Johnstone I. and Tibshirani R. (2004) <doi:10.1214/009053604000000067> *Least Angle Regression*, *The Annals of Statistics*, Vol. 32(2), 407-499.

Pazira H., Augugliaro L. and Wit E.C. (2018) <doi:10.1007/s11222-017-9761-7> *Extended differential-geometric LARS for high-dimensional GLMs with general dispersion parameter*, *Statistics and Computing*, Vol 28(4), 753-774.

`AIC.dglars` *Akaike's An Information Criterion*

Description

`AIC.dglars` is used to compute the Akaike's 'An Information Criterion' for the sequence of models estimated by "`dglars`".

Usage

```
## S3 method for class 'dglars'
AIC(object, phi = c("pearson", "deviance", "mle", "grcv"),
k = 2, complexity = c("df", "gdf"), g = NULL, ...)

## S3 method for class 'dglars'
BIC(object, ...)
```

Arguments

`object`	a fitted `dglars` object.
`phi`	a description of the estimator of the dispersion parameter (see below for more details).
`k`	non negative value used to weight the complexity of the fitted dglars model (see below for more details).
`complexity`	argument used to specify the method to measure the complexity of a fitted dglars model, i.e. the number of non-zero estimates (`complexity = "df"`) of the generalized degrees-of-freedom (`complexity = "gdf"`); see below for more details.
`g`	vector of values of the tuning parameter.
`...`	further arguments passed to the function `link{grcv}`.

Details

The values returned by `AIC.dglars` are computed according to the following formula of a generic measure of Goodness-of-Fit (GoF):

$$-2\text{log-likelihood} + k\text{comp},$$

where "comp" represents the term used to measure the complexity of the fitted model, and k is the 'weight' of the complexity in the previous formula.

For binomial and Poisson family, the log-likelihood function is evaluated assuming that the dispersione parameter is known and equal to one while for the remaining families the dispersion parameter is estimated by the method specified by `phi` (see `phihat` for more details).

According to the results given in Augugliaro et. al. (2013), the complexity of a model fitted by dglars method can be measured by the classical notion of 'Degrees-of-Freedom' (`complexity = "df"`), i.e., the number of non-zero estimated, or by the notion of 'Generalized Degrees-of-Freedom' (`complexity = "gdf"`).

By the previous formula, it is easy to see that the standard AIC-values are obtained setting `k = 2` and `complexity = "df"` (default values for the function `AIC.dglars`) while the so-called BIC-values (Schwarz's Bayesian criterion) are obtained setting `k = log(n)`, where n denotes the sample size, and `complexity = "df"` (default values for the function `BIC.dglars`).

The optional argument `g` is used to specify the values of the tuning parameter; if not specified (default), the values of the measure of goodness-of-fit are computed for the sequence of models storage in `object` otherwise `predict.dglars` is used to compute the estimate of the parameters needed to evaluate the log-likelihood function (see the example below).

Value

`AIC.dglars` and `BIC.dglars` return a named list with class "`gof_dglars`" and components:

`val`	the sequence of AIC/BIC-values;
`g`	the sequence of γ-values;
`loglik`	the sequence of log-likelihood values used to compute the AIC or BIC;
`k`	the non negative value used to weight the complexity of the fitted dglars model;
`comp`	the measures of model complexity used to compute the measure of goodness-of-fit. It is equal to `npar` when codecomplexity = "df";
`npar`	the seqeunce of the number of non-zero estimates
`phi`	a description of the estimator used to estimate the dispersion pamater;
`phih`	the vector of penalized estimate of the dispersion parameter used to evaluate the log-likelihood function;
`complexity`	character specifying the method to measure the complexity of a fitted dglars model;
`object`	the fitted `dglars` object;
`type`	character specifying the type of used measure-of-goodness of fit, i.e., AIC, BIC or GoF.

In order to summarize the information about the AIC-valuse, a `print` method is available for an object with class "`gof_dglars`".

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References

Augugliaro L., Mineo A.M. and Wit E.C. (2013) <doi:10.1111/rssb.12000> *dgLARS: a differential geometric approach to sparse generalized linear models*, *Journal of the Royal Statistical Society. Series B.*, Vol 75(3), 471-498.

Sakamoto, Y., Ishiguro, M., and Kitagawa G. (1986, ISBN:978-90-277-2253-9) *Akaike Information Criterion Statistics*. KTK Scientific Publishers, 1986.

See Also

logLik.dglars, predict.dglars, dglars and summary.dglars.

Examples

```
#################################
# y ~ Pois

library("dglars")
set.seed(123)
n <- 100
p <- 5
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + X[, 1]  + X[, 2]
mu <- poisson()$linkinv(eta)
y <- rpois(n, mu)
out <- dglars(y ~ X, poisson)
out
AIC(out)
AIC(out, g = seq(2, 1, by = -0.1))
AIC(out, complexity = "gdf")
AIC(out, k = log(n)) #BIC-values
BIC(out)

#################################
# y ~ Gamma

n <- 100
p <- 50
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + 2 * X[, 1L]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
out <- dglars(y ~ X, Gamma("log"))

AIC(out, phi = "pearson")
AIC(out, phi = "deviance")
AIC(out, phi = "mle")
AIC(out, phi = "grcv")
```

alon — *Data from the microarray experiment done by Alon et al. (1999)*

Description

The data set contains the gene expression data origianlly analized in Alon et al. (1999). 62 samples (40 tumor samples, 22 normal samples) from colon-cancer patients were analyzed with an Affymetrix oligonucleotide Hum6000 array. The binary variable `y` is used to indicate a normal sample (`y = 0`) or a tumor sample (`y = 1`).

Two thousand out of around 6500 genes were selected based on the confidence in the measured expression levels (for details refer to publication).

Usage

```
data(alon)
```

References

Alon U., Barkai N., Notterman D.A., Gish K., Ybarra S., Mack D. and Levine A.J. (1999) <doi:10.1073/pnas.96.12.6745> *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissue probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. USA* **96**, 6745-6750

breast — *Breast Cancer microarray experiment*

Description

This data set details microarray experiment for 52 breast cancer patients. The binary variable `status` is used to indicate whether or not the patient has died of breast cancer (`status = 0` = did not die of breast cancer, `status = 1` = died of breast cancer). The other variables contain the amplification or deletion of the considered genes.

Rather than measuring gene expression, this experiment aims to measure gene amplification or deletion, which refers to the number of copies of a particular DNA sequence within the genome. The aim of the experiment is to find out the key genomic factors involved in agressive and non-agressive forms of breast cancer.

The experiment was conducted by the Dr.\ John Bartlett and Dr.\ Caroline Witton in the Division of Cancer Sciences and Molecular Pathology of the University of Glasgow at the city's Royal Infirmary.

Usage

```
data(breast)
```

Source

Dr. John Bartlett and Dr. Caroline Witton, Division of Cancer Sciences and Molecular Pathology, University of Glasgow, Glasgow Royal Infirmary.

References

Augugliaro L., Mineo A.M. and Wit E.C. (2013) <doi:10.1111/rssb.12000> *dgLARS: a differential geometric approach to sparse generalized linear models*, *Journal of the Royal Statistical Society. Series B.*, Vol 75(3), 471-498.

Wit E.C. and McClure J. (2004, ISBN:978-0-470-84993-4) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

`coef.cvdglars` *Extract the Coefficients Estimated by* `cvdglars`

Description

`coef.cvdglars` is used to extract the coefficients estimated by k-fold cross-validation deviance.

Usage

```
## S3 method for class 'cvdglars'
coef(object, ...)
```

Arguments

`object`	fitted `cvdglars` object
`...`	additional argument used to ensure the compatibility with the generic method function “`coef`”.

Value

`coef.cvdglars` returns a named list with components `beta`, i.e., the estimate of the coefficient vector, and `phi` the estimate of the dispersion parameter.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <`luigi.augugliaro@unipa.it`>

See Also

`cvdglars` function.

Examples

```
############################
# Logistic regression model
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), n, p)
b <- 1:2
eta <- b[1] + X[,1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- cvdglars.fit(X, y, family = binomial)
coef(fit)
```

coef.dglars *Extract the dgLARS Coefficient Path*

Description

coef.dglars is used to extract the coefficient path computed by dgLARS method.

Usage

```
## S3 method for class 'dglars'
coef(object, type = c("pearson", "deviance", "mle", "grcv"),
g = NULL, ...)
```

Arguments

object	fitted dglars object.
type	a description of the estimator used for the dispersion parameter.
g	vector of values of the tuning parameter.
...	further arguments passed to the function link{grcv}.

Details

coef.dglars is a wrapper function calling “predict.dglars” and “phihat”. By default, this function returns the sequence of the penalized coefficients and the sequence of the penalized estimate of the dispersion parameter ϕ. The user can specify the argumnets of the function grcv by the argument ...).

Value

coef.dglars returns a named list with component:

beta	the sequence of the penalized estimates of the regression coefficients;
phi	the penalized estimates of the dispersion parameter;
g	the vector of the values of the tuning parameter.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

See Also

predict.dglars, phihat and grcv.

Examples

```
###########################
# Logistic regression model
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), n, p)
b <- 1:2
eta <- b[1] + X[, 1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- dglars(y ~ X, family = binomial)
coef(fit)
coef(fit, g = seq(4, 0.5, length = 10))

###########################
# Gamma family
n <- 100
p <- 10
X <- matrix(abs(rnorm(n * p)), n, p)
b <- 1:2
eta <- b[1] + X[, 1] * b[2]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
fit <- dglars(y ~ X, Gamma("log"))
coef(fit, type = "pearson")
coef(fit, type = "deviance")
coef(fit, type = "mle")
```

cvdglars *Cross-Validation Method for dgLARS*

Description

Uses the k-fold cross-validation deviance to estimate the solution point of the dgLARS solution curve.

Usage

```
cvdglars(formula, family = gaussian, g, unpenalized,
b_wght, data, subset, contrasts = NULL, control = list())

cvdglars.fit(X, y, family = gaussian, g, unpenalized,
b_wght, control = list())
```

Arguments

`formula` an object of class “`formula`”: a symbolic description of the model to be fitted. When the `binomial` family is used, the responce can be a vector with entries 0/1 (failure/success) or, alternatively, a matrix where the first column is the number of “successes” and the second column is the number of “failures”.

`family` a description of the error distribution and link function used to specify the model. This can be a character string naming a family function or the result of a call to a family function (see `family` for details). By default the gaussian family with identity link function is used.

`g` argument available only for `ccd` algorithm. When the `ccd` algorithm is used to fit the dgLARS model, this argument can be used to specify the values of the tuning parameter.

`unpenalized` a vector used to specify the unpenalized estimators; `unpenalized` can be a vector of integers or characters specifying the names of the predictors with unpenalized estimators.

`b_wght` a vector, with length equal to the number of columns of the matrix `X`, used to compute the weights used in the adaptive dgLARS method. `b_wght` is used to specify the initial estimates of the parameter vector.

`data` an optional data frame, list or environment (or object coercible by ‘as.data.frame’ to a data frame) containing the variables in the model. If not found in ‘data’, the variables are taken from ‘environment(formula)’.

`subset` an optional vector specifying a subset of observations to be used in the fitting process.

`contrasts` an optional list. See the ‘contrasts.arg’ of ‘model.matrix.default’.

`control` a list of control parameters. See ‘Details’.

`X` design matrix of dimension $n \times p$.

`y` response vector. When the `binomial` family is used, this argument can be a vector with entries 0 (failure) or 1 (success). Alternatively, the response can be a matrix where the first column is the number of “successes” and the second column is the number of “failures”.

Details

`cvdglars` function runs `dglars` `nfold+1` times. The deviance is stored, and the average and its standard deviation over the folds are computed.

`cvdglars.fit` is the workhorse function: it is more efficient when the design matrix have already been calculated. For this reason we suggest to use this function when the dgLARS method is applied in a high-dimensional setting, i.e. when `p>n`.

The control argument is a list that can supply any of the following components:

algorithm**:** a string specifying the algorithm used to compute the solution curve. The predictor-corrector algorithm is used when algorithm = ''pc'' (default), while the cyclic coordinate d escent method is used setting algorithm = ''ccd'';

method**:** a string by means of to specify the kind of solution curve. If method = ''dgLASSO'' (default) the algorithm computes the solution curve defined by the differential geometric generalization of the LASSO estimator; otherwise, if method = ''dgLARS'', the differential geometric generalization of the least angle regression method is used;

nfold**:** a non negative integer used to specify the number of folds. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Default is nfold = 10;

foldid a n-dimensional vector of integers, between 1 and n, used to define the folds for the cross-validation. By default foldid is randomly generated;

ng**:** number of values of the tuning parameter used to compute the cross-validation deviance. Default is ng = 100;

nv**:** control parameter for the pc algorithm. An integer value belonging to the interval $[1; min(n, p)]$ (default is nv = min(n-1,p)) used to specify the maximum number of variables included in the final model;

np**:** control parameter for the pc/ccd algorithm. A non negative integer used to define the maximum number of points of the solution curve. For the predictor-corrector algorithm np is set to $50 \cdot min(n - 1, p)$ (default), while for the cyclic coordinate descent method is set to 100 (default), i.e. the number of values of the tuning parameter γ;

g0**:** control parameter for the pc/ccd algorithm. Set the smallest value for the tuning parameter γ. Default is g0 = ifelse(p<n, 1.0e-06, 0.05);

dg_max**:** control parameter for the pc algorithm. A non negative value used to specify the maximum length of the step size. Setting dg_max = 0 (default) the predictor-corrector algorithm uses the optimal step size (see Augugliaro et al. (2013) for more details) to approximate the value of the tuning parameter corresponding to the inclusion/exclusion of a variable from the model;

nNR**:** control parameter for the pc algorithm. A non negative integer used to specify the maximum number of iterations of the Newton-Raphson algorithm used in the corrector step. Default is nNR = 200;

NReps**:** control parameter for the pc algorithm. A non negative value used to define the convergence criterion of the Newton-Raphson algorithm. Default is NReps = 1.0e-06;

ncrct**:** control parameter for the pc algorithm. When the Newton-Raphson algorithm does not converge, the step size ($d\gamma$) is reduced by $d\gamma = cf \cdot d\gamma$ and the corrector step is repeated. ncrct is a non negative integer used to specify the maximum number of trials for the corrector step. Default is ncrct = 50;

cf**:** control parameter for the pc algorithm. The contractor factor is a real value belonging to the interval $[0, 1]$ used to reduce the step size as previously described. Default is cf = 0.5;

nccd**:** control parameter for the ccd algorithm. A non negative integer used to specify the maximum number for steps of the cyclic coordinate descent algorithm. Default is 1.0e+05.

eps control parameter for the pc/ccd algorithm. The meaning of this parameter is related to the algorithm used to estimate the solution curve:

i. if `algorithm = ''pc''` then `eps` is used

   a. to identify a variable that will be included in the active set (absolute value of the corresponding Rao's score test statistic belongs to $[\gamma - \texttt{eps}, \gamma + \texttt{eps}]$);

   b. to establish if the corrector step must be repeated;

   c. to define the convergence of the algorithm, i.e., the actual value of the tuning parameter belongs to the interval [`g0 - eps`, `g0 + eps`];

ii. if `algorithm = ''ccd''` then `eps` is used to define the convergence for a single solution point, i.e., each inner coordinate-descent loop continues until the maximum change in the Rao's score test statistic, after any coefficient update, is less than `eps`.

Default is `eps = 1.0e-05`.

Value

`cvdglars` returns an object with S3 class "`cvdglars`", i.e. a list containing the following components:

`call` the call that produced this object;

`formula_cv` if the model is fitted by `cvdglars`, the used formula is returned;

`family` a description of the error distribution used in the model;

`var_cv` a character vector with the name of variables selected by cross-validation;

`beta` the vector of the coefficients estimated by cross-validation;

`phi` the cross-validation estimate of the disperion parameter;

`dev_m` a vector of length `ng` used to store the mean cross-validation deviance;

`dev_v` a vector of length `ng` used to store the variance of the mean cross-validation deviance;

`g` the value of the tuning parameter corresponding to the minimum of the cross-validation deviance;

`g0` the smallest value for the tuning parameter;

`g_max` the value of the tuning parameter corresponding to the starting point of the dgLARS solution curve;

`X` the used design matrix;

`y` the used response vector;

`w` the vector of weights used to compute the adaptive dglars method;

`conv` an integer value used to encode the warnings and the errors related to the algorithm used to fit the dgLARS solution curve. The values returned are:

`0` convergence of the algorithm has been achieved,

`1` problems related with the predictor-corrector method: error in predictor step,

`2` problems related with the predictor-corrector method: error in corrector step,

`3` maximum number of iterations has been reached,

`4` error in dynamic allocation memory;

`control` the list of control parameters used to compute the cross-validation deviance.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References

Augugliaro L., Mineo A.M. and Wit E.C. (2014) <doi:10.18637/jss.v059.i08> *dglars: An R Package to Estimate Sparse Generalized Linear Models*, *Journal of Statistical Software*, Vol 59(8), 1-40. https://www.jstatsoft.org/v59/i08/.

Augugliaro L., Mineo A.M. and Wit E.C. (2013) <doi:10.1111/rssb.12000> *dgLARS: a differential geometric approach to sparse generalized linear models*, *Journal of the Royal Statistical Society. Series B.*, Vol 75(3), 471-498.

See Also

`coef.cvdglars`, `print.cvdglars`, `plot.cvdglars` methods

Examples

```
###########################
# Logistic regression model
# y ~ Binomial
set.seed(123)
n <- 100
p <- 100
X <- matrix(rnorm(n * p), n, p)
b <- 1:2
eta <- b[1] + X[, 1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit_cv <- cvdglars.fit(X, y, family = binomial)
fit_cv
```

`dglars` *dgLARS Solution Curve for GLM*

Description

`dglars` function is used to estimate the solution curve defined by dgLARS method.

Usage

```
dglars(formula, family = gaussian, g, unpenalized,
b_wght, data, subset, contrasts = NULL, control = list())

dglars.fit(X, y, family = gaussian, g, unpenalized,
b_wght, control = list())
```

Arguments

`formula` an object of class "`formula`": a symbolic description of the model to be fitted. When the `binomial` family is used, the responce can be a vector with entries 0/1 (failure/success) or, alternatively, a matrix where the first column is the number of "successes" and the second column is the number of "failures".

`family` a description of the error distribution and link function used to specify the model. This can be a character string naming a family function or the result of a call to a family function (see `family` for details). By default the gaussian family with identity link function is used.

`g` argument available only for `ccd` algorithm. When the model is fitted by using the `ccd` algorithm, this argument can be used to specify the values of the tuning parameter.

`unpenalized` a vector used to specify the unpenalized estimators; `unpenalized` can be a vector of integers or characters specifying the names of the predictors with unpenalized estimators (see example below for more details).

`b_wght` a $p+1$-dimensional vector used to compute the weights in the adaptive dgLARS method. `b_wght` is used to specify the initial estimates of the parameter vector.

`data` an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)'.

`subset` an optional vector specifying a subset of observations to be used in the fitting process.

`contrasts` an optional list. See the 'contrasts.arg' of 'model.matrix.default'.

`control` a list of control parameters. See 'Details'.

`X` design matrix of dimension $n \times p$.

`y` response vector. When the `binomial` family is used, this argument can be a vector with entries 0 (failure) or 1 (success). Alternatively, the response can be a matrix where the first column is the number of "successes" and the second column is the number of "failures".

Details

`dglars` function implements the differential geometric generalization of the least angle regression method (Efron et al., 2004) proposed in Augugliaro et al. (2013) and Pazira et al. (2017).

As in "`glm`", the user can specify family and link function using the argument `family`. When the `binomial` family is used, the responce can be a vector with entries 0/1 (failure/success) or, alternatively, a matrix where the first column is the number of "successes" and the second column is the number of "failures". Starting with the version 2.0.0, the model can be specified combining family and link functions as describted in the following table:

Family	**Link**
`gaussian`	'`identity`', '`log`' and '`inverse`'
`binomial`	'`logit`', '`probit`', '`cauchit`', '`log`' and '`cloglog`'
`poisson`	'`log`', '`identity`', and '`sqrt`'
`Gamma`	'`inverse`', '`identity`' and '`log`'
`inverse.gaussian`	'`1/mu^2`', '`inverse`', '`identity`', and '`log`'

The R code for binomial, Gamma and inverse gaussian families is due to Hassan Pazira while the fortran version is due to Luigi Augugliaro.

dglars.fit is a workhorse function: it is more efficient when the design matrix does not require manipulations. For this reason we suggest to use this function when the dgLARS method is applied in a high-dimensional setting, i.e., when p>n.

When gaussian, gamma or inverse.gaussian is used to model the error distribution, dglars returns the vector of the estimates of the dispersion parameter ϕ; by default, the generalized Pearson statistic is used as estimator but the user can use the function phihat to specify other estimators (see phihat for more details).

The dgLARS solution curve can be estimated using two different algorithms, i.e. the predictor-corrector method and the cyclic coordinate descent method (see below for more details about the argument algorithm). The first algorithm is based on two steps. In the first step, called predictor step, an approximation of the point that lies on the solution curve is computed. If the control parameter dg_max is equal to zero, in this step it is also computed an approximation of the optimal step size using a generalization of the method proposed in Efron et al. (2004). The optimal step size is defined as the reduction of the tuning parameter, denoted by $d\gamma$, such that at $\gamma - d\gamma$ there is a change in the active set. In the second step, called corrector step, a Newton-Raphson algorithm is used to correct the approximation previously computed. The main problem of this algorithm is that the number of arithmetic operations required to compute the approximation scales as the cube of the variables, this means that such algorithm is cumbersome in a high dimensional setting. To overcome this problem, the second algorithm compute the dgLARS solution curve using an adaptive version of the cyclic coordinate descent method proposed in Friedman et al. (2010).

The argument control is a list that can supply any of the following components:

algorithm: a string specifying the algorithm used to compute the solution curve. The predictor-corrector algorithm is used when algorithm = ''pc'' (default), while the cyclic coordinate descent method is used setting algorithm = ''ccd'';

method: a string by means of to specify the kind of solution curve. If method = ''dgLASSO'' (default) the algorithm computes the solution curve defined by the differential geometric generalization of the LASSO estimator; otherwise (method = ''dgLARS'') the differential geometric generalization of the least angle regression method is used;

nv: control parameter for the pc algorithm. An integer value between 1 and $\min(n, p)$ used to specify the maximum number of variables in the final model. Default is nv = min(n - 1, p);

np: control parameter for the pc/ccd algorithm. A non negative integer used to define the maximum number of solution points. For the predictor-corrector algorithm np is set to $50 \times \min(n-1, p)$ (default); for the cyclic coordinate descent method, if g is not specified, this argument is set equal to 100 (default);

g0: control parameter for the pc/ccd algorithm. This parameter is used to set the smallest value for the tuning parameter γ. Default is g0 = ifelse(p < n, 1.0e-04, 0.05); this argument is not required when g is used with the cyclic coordinate descent algorithm;

dg_max: control parameter for the pc algorithm. A non negative value used to specify the largest value for the step size. Setting dg_max = 0 (default) the predictor-corrector algorithm computes an approximation of the optimal step size (see Augugliaro et al. (2013) for more details);

nNR: control criterion parameter for the pc algorithm. A non negative integer used to specify the maximum number of iterations of the Newton-Raphson algorithm. Default is nNR = 50;

NReps**:** control parameter for the pc algorithm. A non negative value used to define the convergence of the Newton-Raphson algorithm. Default is NReps = 1.0e-06;

ncrct**:** control parameter for the pc algorithm. When the Newton-Raphson algorithm does not converge, the step size ($d\gamma$) is reduced by $d\gamma = cf \cdot d\gamma$ and the corrector step is repeated. ncrct is a non negative integer used to specify the maximum number of trials for the corrector step. Default is ncrct = 50;

cf**:** control parameter for the pc algorithm. The contractor factor is a real value belonging to the interval $[0, 1]$ used to reduce the step size as previously described. Default is cf = 0.5;

nccd**:** control parameter for the ccd algorithm. A non negative integer used to specify the maximum number for steps of the cyclic coordinate descent algorithm. Default is 1.0e+05.

eps control parameter for the pc/ccd algorithm. The meaning of this parameter is related to the algorithm used to estimate the solution curve:

i. if algorithm = ''pc'' then eps is used

a. to identify a variable that will be included in the active set (absolute value of the corresponding Rao's score test statistic belongs to $[\gamma - \texttt{eps}, \gamma + \texttt{eps}]$);

b. to establish if the corrector step must be repeated;

c. to define the convergence of the algorithm, i.e., the actual value of the tuning parameter belongs to the interval [g0 - eps, g0 + eps];

ii. if algorithm = ''ccd'' then eps is used to define the convergence for a single solution point, i.e., each inner coordinate-descent loop continues until the maximum change in the Rao's score test statistic, after any coefficient update, is less than eps.

Default is eps = 1.0e-05.

Value

dglars returns an object with S3 class "dglars", i.e., a list containing the following components:

call	the call that produced this object;
formula	if the model is fitted by dglars, the used formula is returned;
family	a description of the error distribution used in the model;
unpenalized	the vector used to specify the unpenalized estimators;
np	the number of points of the dgLARS solution curve;
beta	the $(p + 1) \times$ np matrix corresponding to the dgLARS solution curve;
phi	the np dimensional vector of the Pearson estimates of the disperion parameter;
ru	the matrix of the Rao's score test statistics of the variables included in the final model. This component is reported only if the predictor-corrector algorithm is used to fit the model;
dev	the np dimensional vector of the deviance corresponding to the values of the tuning parameter γ;
nnonzero	the sequence of number of nonzero coefficients for each value of the tuning parameter γ;
g	the sequence of γ values used to compute the solution curve;
X	the used design matrix;

`y` the used response vector;

`w` the vector of weights used to compute the adaptive dglars method;

`action` a `np` dimensional vector of characters used to show how is changed the active set for each value of the tuning parameter γ;

`conv` an integer value used to encode the warnings and the errors related to the algorithm used to fit the model. The values returned are:

- `0` convergence of the algorithm has been achieved;
- `1` problems related with the predictor-corrector method: error in predictor step;
- `2` problems related with the predictor-corrector method: error in corrector step;
- `3` maximum number of iterations has been reached;
- `4` error in dynamic allocation memory;
- `5` fitted expected value is out of range;
- `6` does not exist dgLARS estimator;
- `7` maximum number of solution points ('codenp') reached.

`control` the list of control parameters used to compute the dgLARS solution curve.

Author(s)

Luigi Augugliaro and Hassan Pazira
Maintainer: Luigi Augugliaro <`luigi.augugliaro@unipa.it`>

References

Augugliaro L., Mineo A.M. and Wit E.C. (2016) <doi:10.1093/biomet/asw023> *A differential-geometric approach to generalized linear models with grouped predictors*, *Biometrika*, Vol 103(3), 563-577.

Augugliaro L., Mineo A.M. and Wit E.C. (2014) <doi:10.18637/jss.v059.i08> *dglars: An R Package to Estimate Sparse Generalized Linear Models*, *Journal of Statistical Software*, Vol 59(8), 1-40. `https://www.jstatsoft.org/v59/i08/`.

Augugliaro L., Mineo A.M. and Wit E.C. (2013) <doi:10.1111/rssb.12000> *dgLARS: a differential geometric approach to sparse generalized linear models*, *Journal of the Royal Statistical Society. Series B.*, Vol 75(3), 471-498.

Efron B., Hastie T., Johnstone I. and Tibshirani R. (2004) <doi:10.1214/009053604000000067> *Least Angle Regression*, *The Annals of Statistics*, Vol. 32(2), 407-499.

Friedman J., Hastie T. and Tibshirani R. (2010) <doi:10.18637/jss.v033.i01> *Regularization Paths for Generalized Linear Models via Coordinate Descent*, *Journal of Statistical Software*, Vol. 33(1), 1-22.

Pazira H., Augugliaro L. and Wit E.C. (2018) <doi:10.1007/s11222-017-9761-7> *Extended differential geometric LARS for high-dimensional GLMs with general dispersion parameter*, *Statistics and Computing*, Vol. 28(4), 753-774.

See Also

`coef.dglars`, `phihat`, `plot.dglars`, `print.dglars` and `summary.dglars` methods.

Examples

```
set.seed(123)

#############################
# y ~ Binomial
n <- 100
p <- 100
X <- matrix(rnorm(n * p), n, p)
eta <- 1 + 2 * X[,1]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- dglars(y ~ X, family = binomial)
fit

# adaptive dglars method
b_wght <- coef(fit)$beta[, 20]
fit <- dglars(y ~ X, family = binomial, b_wght = b_wght)
fit

# the first three coefficients are not penalized
fit <- dglars(y ~ X, family = binomial, unpenalized = 1:3)
fit

# 'probit' link function
fit <- dglars(y ~ X, family = binomial(”probit”))
fit

############################
# y ~ Poisson
n <- 100
p <- 100
X <- matrix(rnorm(n * p), n, p)
eta <- 2 + 2 * X[,1]
mu <- poisson()$linkinv(eta)
y <- rpois(n, mu)
fit <- dglars(y ~ X, family = poisson)
fit

############################
# y ~ Gamma
n <- 100
p <- 100
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + 2 * X[,1]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
fit <- dglars(y ~ X, Gamma(”log”))
fit
```

duke *Duke breast cancer microarray experiment*

Description

This data set details microarray experiment for 44 breast cancer patients. The binary variable `Status` is used to classify the patients into estrogen receptor-positive (`Status = 0`) and estrogen receptor-negative (`Status = 1`). The other variables contain the expression level of the considered genes.

Usage

```
data(duke)
```

References

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J.A. Olson, Jr., J.R. Marks and Joseph R. Nevins (2001) <doi:10.1073/pnas.201162998> *Predicting the clinical status of human breast cancer by using gene expression profiles*, *Proceedings of the National Accademy of Sciences of the USA*, Vol 98(20), 11462-11467.

gdf *Estimate the Generalized Degrees-of-Freedom*

Description

`gdf` returns to estimate of the generalized degrees-of-freedom.

Usage

```
gdf(object)
```

Arguments

`object`	fitted `dglars` object.

Details

For a general nonlinear modelling procedure, a rigorous definition of degrees-of-freedom is obtained using the covariance penalty theory (Efron, 2004). This theory was used in Augugliaro et al. (2013) to define a measure of model complexity for the dgLARS method, called “generalized degrees-of-freedom”. The `gdf` function implements the estimator proposed in Augugliaro et al. (2013).

Value

`gdf` returns a vector of length `np` with the generalized degrees-of-freedom.

Author(s)

Luigi Augugliaro and Hassan Pazira
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References

Augugliaro L., Mineo A.M. and Wit E.C. (2014) <doi:10.18637/jss.v059.i08> *dglars: An R Package to Estimate Sparse Generalized Linear Models*, *Journal of Statistical Software*, Vol 59(8), 1-40. https://www.jstatsoft.org/v59/i08/.

Augugliaro L., Mineo A.M. and Wit E.C. (2013) <doi:10.1111/rssb.12000> *dgLARS: a differential geometric approach to sparse generalized linear models*, *Journal of the Royal Statistical Society. Series B.*, Vol 75(3), 471-498.

Efron B. (2004) <doi:10.1198/016214504000000692> *The estimation of prediction error: covariance penalties and cross-validation*, *Journal of the American Statistical Association*, Vol. 99(467), 619-632.

See Also

dglars, AIC.dglars, BIC.dglars and summary.dglars.

Examples

```
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n*p), n, p)
b <- 1:2
eta <- b[1] + X[,1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- dglars(y ~ X, binomial)
gdf(fit)
```

grcv *General Refitted Cross-Validation Estimator*

Description

grcv computes the estimate of the dispersion parameter using the general refitted cross-validation method.

Usage

```
grcv(object, type = c("BIC", "AIC"), nit = 10L, trace = FALSE,
     control = list(), ...)
```

Arguments

`object`	fitted `dglars` object.
`type`	the measure of goodness-of-fit used in Step 2 to select the two set of variables (see section **Description** for more details). Default is `type = BIC`.
`control`	a list of control parameters passed to the function `dglars`.
`nit`	integer specifying the number of times that the general refitted cross-validation method is repeated (see section **Description** for more details). Default is `nit = 10L`.
`trace`	flag used to print out information about the algorithm. Default is `trace = FALSE`.
`...`	further arguments passed to the functions `AIC.dglars` or `BIC.dglars`.

Details

The general refitted cross-validation (grcv) estimator (Pazira et al., 2018) is an estimator of the dispersion parameter of the exponential family based on the following four stage procedure:

Step	Description
1.	randomly split the data set $D = (y, X)$ into two even datasets, denoted by D_1 and D_2.
2.	fit dglars model to the dataset D_1 to select a set of variables A_1. fit dglars model to the dataset D_2 to select a set of variables A_2.
3.	fit the glm model to the dataset D_1 using the variables that are in A_2; then estimate the disporsion parameter using the Pearson method. Denote by $\hat{\phi}_1(A_2)$ the resulting estimate. fit the glm model to the dataset D_2 using the variables that are in A_1; then estimate the disporsion parameter using the Pearson method. Denote by $\hat{\phi}_2(A_1)$ the resulting estimate.
4.	estimate ϕ using the following estimator: $\hat{\phi}_{grcv} = (\hat{\phi}_1(A_2) + \hat{\phi}_2(A_1))/2$.

In order to reduce the random variabilty due to the splitting of the dataset (Step 1), the previous procedure is repeated '`nit`'-times; the median of the resulting estimates is used as final estimate of the dispersion parameter. In Step 3, the two sets of variables are selected using the `AIC.dglars` and `BIC.dglars`; in this step, the Pearson method is used to obtain a first estimate of the dispersion parameter. Furthermore, if the function `glm` does not converge the function `dglars` is used to compute the maximum likelihood estimates.

Value

`grcv` returns the estimate of the dispersion parameter.

Author(s)

Luigi Augugliaro and Hassan Pazira
Maintainer: Luigi Augugliaro <`luigi.augugliaro@unipa.it`>

References

Pazira H., Augugliaro L. and Wit E.C. (2018) <doi:10.1007/s11222-017-9761-7> *Extended differential-geometric LARS for high-dimensional GLMs with general dispersion parameter*, *Statistics and Computing*, Vol 28(4), 753-774.

See Also

phihat, AIC.dglars and BIC.dglars.

Examples

```
############################
# y ~ Gamma
set.seed(321)
n <- 100
p <- 50
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + 2 * X[,1]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
fit <- dglars(y ~ X, Gamma("log"))

phi
grcv(fit, type = "AIC")
grcv(fit, type = "BIC")
```

`logLik.dglars` *Extract Log-Likelihood*

Description

logLik method for an object with class ‘dglars’.

Usage

```
## S3 method for class 'dglars'
logLik(object, phi = c("pearson", "deviance", "mle", "grcv"),
g = NULL, ...)
```

Arguments

`object` any fitted `dglars` object from which the log-likelihood values can be extracted.

`phi` a description of the estimator used to estimate the dispersion pamater (see below for more details).

`g` vector of values of the tuning parameter.

`...` further arguments passed to the function `grcv`.

Details

logLik.dglars returns the sequence of the log-likelihood values of the models fitted by “dglars”. For the binomial and Poisson family, the dispersion parameter is assumed known and equal to one while for the other families the dispersion parameter is estimated using the method specified by the argument “phi” (see phihat for more details). The optional argument g is used to specified the values of the tuning parameter; if not specified (default), the log-likelihood values are computed for the sequence of models storage in object otherwise predict.dglars is used to compute the estimate of the parameters needed to evaluate the log-likelihood function (see the example below).

Value

logLik.dglars returns an object of class “loglik_dglars”. This is a named list containing the following components:

`loglik` the log-likelihood of the sequence of models fitted by dglars method.

`df` the numbers of non-zero estimates corresponding to the used γ-values.

`object` the fitted dglars object.

`g` the sequence of γ-values.

`phi` a description of the estimator used to estimate the dispersion pamater.

`phih` the sequence of estimated dispersion parameter.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

See Also

dglars, phihat, predict.dglars, AIC.dglars and BIC.dglars.

Examples

```
#################################
# y ~ Poisson

library(dglars)
set.seed(123)
n <- 100
p <- 5
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + X[, 1]  + X[, 2]
mu <- poisson()$linkinv(eta)
y <- rpois(n, mu)
out <- dglars(y ~ X, poisson)
logLik(out)
logLik(out, g = seq(2, 0.5, by = -0.1))

############################
# y ~ Gamma
```

```
n <- 100
p <- 5
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + 2 * X[, 1L]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
out <- dglars(y ~ X, Gamma("log"))

# generalized Pearson statistic
logLik(out, phi = "pearson")

# deviance estimator
logLik(out, phi = "deviance")

# mle estimator
logLik(out, phi = "mle")

# grcv estimator
logLik(out, phi = "grcv")
```

phihat *Estimate the Dispersion Parameter*

Description

phihat returns the estimates of the dispersion parameter.

Usage

```
phihat(object, type = c("pearson", "deviance", "mle", "grcv"), g = NULL, ...)
```

Arguments

object	fitted dglars object.
type	a description of the used estimator.
g	vector of values of the tuning parameter.
...	further arguments passed to the function grcv.

Details

phihat implements four different estimators of the dispersion parameter, i.e, the generalized Pearson statistic (type = "pearson"), the deviance estimator (type = "deviance"), the maximum likelihood estimator (type = "mle") and general refitted cross-Validation estimator (type = "grcv") proposed in Pazira et al. (2018). For regression models with Gamma family, the maximum likelihood estimator of the dispersion parameter is computed using the approximation proposed in Cordeiro et al. (1997).

Value

phihat returns a vector with the estimates of the dispersion parameter.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

References

Cordeiro G. M. and McCullagh P. (1991) <doi:10.2307/2345592> Bias Correction in Generalized Linear Models, *Journal of the Royal Statistical Society. Series B.*, Vol 53(3), 629–643.

Jorgensen B. (1997, ISBN:0412997188) *The Theory of Dispersion Models*, Chapman and Hall, Great Britain.

Pazira H., Augugliaro L. and Wit E.C. (2018) <doi:10.1007/s11222-017-9761-7> Extended differential-geometric LARS for high-dimensional GLMs with general dispersion parameter, *Statistics and Computing*, Vol 28(4), 753-774.

See Also

grcv, coef.dglars, logLik.dglars, AIC.dglars and BIC.dglars.

Examples

```
############################
# y ~ Gamma

library("dglars")
set.seed(321)
n <- 100
p <- 50
X <- matrix(abs(rnorm(n*p)),n,p)
eta <- 1 + 2 * X[, 1L]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
fit <- dglars(y ~ X, Gamma("log"))
g <- seq(range(fit$g)[1L], range(fit$g)[2L], length = 10)

# generalized Pearson statistic
phihat(fit, type = "pearson")
phihat(fit, type = "pearson", g = g)

# deviance estimator
phihat(fit, type = "deviance")
phihat(fit, type = "deviance", g = g)

# mle
phihat(fit, type = "mle")
phihat(fit, type = ”mle”, g = g)

# grcv
phihat(fit, type = ”grcv”)
phihat(fit, type = ”grcv”, g = g)
```

plot.cvdglars *Plot from a cvdglars Object*

Description

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the used γ values.

Usage

```
## S3 method for class 'cvdglars'
plot(x, ...)
```

Arguments

`x`	fitted `cvdglars` object.
`...`	additional graphical parameters to plot.

Details

A plot for a `cvdglars` object is produced.

The plot shows the curve of the cross-validation deviance and the upper and lower standard deviation curves. A vertical dashed red line is used the indetify the value of the γ parameter corresponding to the minimum of the cross-validation deviance.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

See Also

`cvdglars` function.

Examples

```
###########################
# Logistic regression model
# y ~ Binomial
set.seed(123)
n <- 100
p <- 100
```

```
X <- matrix(rnorm(n*p), n, p)
b <- 1:2
eta <- b[1] + X[,1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit_cv <- cvdglars.fit(X, y, family = binomial)
plot(fit_cv)
```

plot.dglars *Plot from a dglars Object*

Description

Produces plots to study the sequence of models identified by dgLARS method.

Usage

```
## S3 method for class 'dglars'
plot(x, type = c("both", "AIC", "BIC"), ...)
```

Arguments

x	fitted dglars object.
type	a description of the measure of goodness-of-fit used to compare the sequence of models fitted by dglars or dglars.fit. See below for a more details.
...	further arguments passed to the functions AIC.dglars or BIC.dglars.

Details

plot.dglars method produces different plots to study the sequence of models fitted by dgLARS method.

First plot gives information about the goodness-of-fit of the sequence of models fitted by dgLARS method. The user can plot the sequence of AIC (type = "AIC") or BIC values (type = "BIC"). By default, AIC and BIC values are shown on the same plot (type = "both"). More general measures of goodness-of-fit can be specified by using the argument "..." to pass futher arguments to function AIC.dglars (see the examples below). The value of the tuning parameter corresponding to the minimum of the used measure of goodness-of-fit is indentified by a vertical dashed red line, while the γ values at which corresponds a change in the active set are labeled by vertical dashed gray lines. Second plot shows the coefficient profile plot; if the predictor-corrector algorithm is used to fit the model, the third plot shows the Rao's score test statistics as function of γ.

Author(s)

Luigi Augugliaro and Hassan Pazira
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

See Also

dglars, summary.dglars and AIC.dglars.

Examples

```
###########################
# Logistic regression model
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), n, p)
b <- 1:2
eta <- b[1] + X[, 1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- dglars.fit(X, y, family = binomial)
plot(fit)
plot(fit, type = "AIC")
plot(fit, type = "BIC")
plot(fit, type = "AIC", k = 5)
plot(fit, type = "AIC", complexity = "gdf")
```

predict.dglars *Predict Method for dgLARS Fits.*

Description

predict.dglars is used to obtain general predictions from a dglars object.

Usage

```
## S3 method for class 'dglars'
predict(object, xnew, ynew, g = NULL,
type = c("coefficients", "nnonzero", "predictors", "eta",
"mu", "probability", "class", "deviance"), ...)
```

Arguments

object fitted dglars object.

xnew matrix of new values of the predictors at which predictions are to be made. This argument is not used for type "coefficients", "nnonzero" and "predictors".

ynew vector of new values of the responce variable. This argument is used only when type ie equal to "deviance".

g value(s) of the tuning parameter γ at which the predictions are required. By default, the predictions are made using the sequence of γ values storaged in dglars.

type type of prediction required; see below for more details.

... additional argument used to ensure the compatibility with the generic method function "predict".

Value

The object returned by predict.dglars depends on type argument:

coefficients: a named list with components “beta”, i.e., the matrix corresponding to the dgLARS solution curve, and “phi”, i.e., the sequence of Pearson estimates of the dispersion parameter;

nnonzero: the number of nonzero estimates;

predictors: a named list; each component is a vector containing the indices of the variables that are in the active set;

eta: a matrix with the linear preditors. If xnew is not specified then the linear predictors are computed using object$X;

mu: a matrix with the fitted expeted values, obtained by transforming the linear predictor by the inverse of the link function. For models with ‘binomial’ family, canonical link function (‘logit’) and response a vector with elements 0 (failure) or 1 (success), type = "mu" and type = "probability" give the same result. If xnew is not spefified then these values are computed using object$X;

probability: available only for ‘binomial’ family. In this case predict.dglars returns a matrix with the fitted probabilities; furthermore, if the model is specified by using the canonical link function (‘logit’) and response a vector with entries 0/1, type = "mu" and type = "probability" give the same result. If xnew is not specified then these values are computed using object$X;

class: available only for ‘binomial’ family. In this case predict.dglars returns a matrix with the fitted class. If xnew is not specified then these values are computed using object$X otherwise xnew is used to compute the fitted probabilities;

deviance: a vector with the scaled residual deviances.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

See Also

dglars and coef.dglars.

Examples

```
######################
# Logistic regression model
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), n, p)
Xnew <- matrix(rnorm(n * p), n, p)
b <- 1:2
eta <- b[1] + X[, 1] * b[2]
mu <- binomial()$linkinv(eta)
```

```
y <- rbinom(n, 1, mu)
fit <- dglars.fit(X, y, binomial)
coef(fit)
predict(fit, type = "coefficients")
g <- seq(3, 1, by = -0.1)
coef(fit, g = g)
predict(fit, type = "coefficients", g = g)
predict(fit, type = "nnonzero")
predict(fit, type = "nnonzero", g = g)
predict(fit, type = "predictors")
predict(fit, type = "predictors", g = g)
predict(fit, type = "eta", g = g)
predict(fit, type = "eta", g = g, xnew = Xnew)
predict(fit, type = "mu", g = g)
predict(fit, type = "mu", g = g, xnew = Xnew)
predict(fit, type = "probability", g = g)
predict(fit, type = "probability", g = g, xnew = Xnew)
predict(fit, type = "class", g = g)
predict(fit, type = "class", g = g, xnew = Xnew)
```

print.cvdglars *Print a cvdglars Object*

Description

Print information about the dgLARS models selected by k-fold cross-validation deviance.

Usage

```
## S3 method for class 'cvdglars'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	fitted `dglars` object
`digits`	significant digits in printout
`...`	additional print arguments

Details

The call that produced the object `x` is printed, followed by the estimate of the coefficients of the variables included in the active set. Such estimates are obtained using the whole data set while the optimal value of the tuning parameter is estimated by k-fold cross-validation deviance. The last part of the print gives information about the k-fold cross-validation deviance, the algorithm and the method used to compute the solution curve.

Value

The vector of the estimates is silently returned.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <`luigi.augugliaro@unipa.it`>

See Also

`cvdglars` function.

Examples

```
######################
# Logistic regression model

set.seed(123)

n <- 100
p <- 10
X <- matrix(rnorm(n*p) ,n, p)
b <- 1:2
eta <- b[1] + X[,1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- cvdglars.fit(X, y, family = "binomial")
fit
```

`print.dglars` *Printing a dgLARS Object*

Description

Print information about the sequence of models estimated by dgLARS method.

Usage

```
## S3 method for class 'dglars'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	fitted `dglars` object
`digits`	significant digits in printout
`...`	additional print arguments

Details

The call that produced the object x is printed, followed by a five-column data.frame with columns "Sequence", "g", "Dev", "%Dev" and "n. non zero". The column named "Sequence" gives information on how is changed the active set along the path. The column "g" shows the sequence of γ values used to compute the solution curve, while the columns "Dev" and "%Dev" show the corresponding deviance and the fraction of explained deviance, respectively. Finally the "n. non zero" column shows the number of nonzero coefficients. The last part gives information about the algorithm and the method used to compute the solution curve. The code about the convergence of the used algorithm is also showed.

Value

The data.frame above is silently returned.

Author(s)

Luigi Augugliaro
Maintainer: Luigi Augugliaro <luigi.augugliaro@unipa.it>

See Also

dglars function.

Examples

```
#############################
# y ~ Binomial
set.seed(123)
n <- 100
p <- 100
X <- matrix(rnorm(n * p), n, p)
eta <- 1 + 2 * X[,1]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- dglars(y ~ X, family = binomial)
fit

# adaptive dglars method
b_wght <- coef(fit)$beta[, 20]
fit <- dglars(y ~ X, family = binomial, b_wght = b_wght)
fit

# the first three coefficients are not penalized
fit <- dglars(y ~ X, family = binomial, unpenalized = 1:3)
fit

# 'probit' link function
fit <- dglars(y ~ X, family = binomial("probit"))
fit
```

summary.dglars *Summaryzing dgLARS Fits*

Description

Summary method for an object with class 'dglars'.

Usage

```
## S3 method for class 'dglars'
summary(object, type = c("AIC", "BIC"),
digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	fitted `dglars` object.
`type`	a description of the used measure of goodness-of-fit, i.e., the Akaike Information Criterion (`type = "AIC"`) or the Bayesian Information Criterion (`type = "BIC"`). See below for the description of how to define more general measures of goodness-of-fit.
`digits`	significant digits in printout.
`...`	additional arguments passed to "`AIC.dglars`" or "`BIC.dglars`". See below for more details.

Details

`summary.dglars` gives information about the sequence of models estimtaed by dgLARS method.

To select the best fitted model, summary method uses a measure of goodness-of-fit (GoF) defined as follows:

$$-2\text{log-likelihood} + k\text{comp},$$

where "comp" represents the term used to measure the complexity of the fitted models, and k is the 'weight' of the complexity in the previous formula. This quantity is computed using the functions "`AIC.dglars`" or "`BIC.dglars`".

By default, `summary.dglars` function computes the AIC criterion, but the user can use "`dots`" to pass to the function `AIC.dglars` the additional arguments needed to compute a more general measure of goodness-of-fit, i.e, "`g`", "`phi`", "`k`" and "`complexity`" (see "`AIC.dglars`" for the description of these arguments). Below we give some examples on how to use these additional arguments.

The output of the summary method is divided in two sections.

The first section shows the call that produced `object` followed by a `data.frame`. The column named "`Sequence`" gives information on how is changed the active set along the path. The column "`g`" shows the sequence of the γ-values used to compute the solution curve, while the column "`%Dev`" shows the the fraction of explained deviance. The remaining columns show the complexity measure, the used measure of goodness-of-fit and the corresponding ranking of the fitted models.

The second section shows the details of the selected model, i.e. family and link function used to specify the generalized linear model, the penalized estimate of the coefficient vector, the value of the tuning parameter, the null and residual deviance, and finaly the value of the used measure of goodness-of-fit. Information about the method and the algorithm used to compute the solution curve is also provided.

Value

`summary.dglars` function silently returns a named list with components:

`table` a `data.frame` with the information about the sequence of model fitted by `dglars` function;

`formula.gof` if the model is specified by the formula in `dglars`, then the formula of the selected model is reported;

`b.gof` the estimates of the coefficients of the selected model;

`phi.gof` the estimate of the dispersion parameter of the selected model;

`nulldev` the null residual deviance;

`resdev.gof` the residual deviance of the selected model;

`type` a description of the measure of goodness-of-fit used to select the model;

`k` the ‘weight’ used to compute the measure of goodness-of-fit;

`complexity` a description of the method used to measure the complexity of the fitted models;

`phi` a description of the method used the estimate the dispersion parameter.

Author(s)

Luigi Augugliaro and Hassan Pazira
Maintainer: Luigi Augugliaro <`luigi.augugliaro@unipa.it`>

See Also

`dglars`, `AIC.dglars`, `BIC.dglars` and `gdf` functions.

Examples

```
###########################
# Logistic regression model
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n*p), n, p)
b <- 1:2
eta <- b[1] + X[, 1] * b[2]
mu <- binomial()$linkinv(eta)
y <- rbinom(n, 1, mu)
fit <- dglars(y ~ X, family = binomial)

summary(fit, type = "AIC")
summary(fit, type = "AIC", k = 0.1)
summary(fit, type = "AIC", complexity = "gdf")
summary(fit, type = "BIC", complexity = "df")
summary(fit, type = "BIC", complexity = "gdf")

############################
# y ~ Gamma
n <- 100
p <- 10
X <- matrix(abs(rnorm(n * p)), n, p)
eta <- 1 + 2 * X[, 1]
mu <- drop(Gamma()$linkinv(eta))
shape <- 0.5
phi <- 1 / shape
y <- rgamma(n, scale = mu / shape, shape = shape)
fit <- dglars(y ~ X, Gamma("log"))

summary(fit, phi = "pearson")
summary(fit, phi = "deviance")
summary(fit, phi = "mle")
```

Index