Package ‘glmnet’

April 2, 2018

Type Package

Title Lasso and Elastic-Net Regularized Generalized Linear Models

Version 2.0-16

Date 2018-03-12

Author Jerome Friedman [aut, cre],
Trevor Hastie [aut, cre],
Rob Tibshirani [aut, cre],
Noah Simon [aut, ctb],
Balasubramanian Narasimhan [ctb],
Junyang Qian [ctb]

Maintainer Trevor Hastie <hastie@stanford.edu>

Depends Matrix (>= 1.0-6), utils, foreach

Imports methods

Suggests survival, knitr, lars

Description Extremely efficient procedures for fitting the entire lasso or elastic-net regularization path for linear regression, logistic and multinomial regression models, Poisson regression and the Cox model. Two recent additions are the multiple-response Gaussian, and the grouped multinomial regression. The algorithm uses cyclical coordinate descent in a path-wise fashion, as described in the paper linked to via the URL below.

License GPL-2

VignetteBuilder knitr

URL http://www.jstatsoft.org/v33/i01/.

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-02 12:06:40 UTC

R topics documented:

`glmnet-package` *Elastic net model paths for some generalized linear models*

Description

This package fits lasso and elastic-net model paths for regression, logistic and multinomial regression using coordinate descent. The algorithm is extremely fast, and exploits sparsity in the input x matrix where it exists. A variety of predictions can be made from the fitted models.

Details

Package:	glmnet
Type:	Package
Version:	1.0
Date:	2008-05-14
License:	What license is it under?

Very simple to use. Accepts `x,y` data for regression models, and produces the regularization path over a grid of values for the tuning parameter `lambda`. Only 5 functions: `glmnet`
`predict.glmnet`
`plot.glmnet`
`print.glmnet`
`coef.glmnet`

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, https://web.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010

http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/
Tibshirani, Robert., Bien, J., Friedman, J.,Hastie, T.,Simon, N.,Taylor, J. and Tibshirani, Ryan. (2012) *Strong Rules for Discarding Predictors in Lasso-type Problems, JRSSB, vol 74*,
http://statweb.stanford.edu/~tibs/ftp/strong.pdf
Stanford Statistics Technical Report
Glmnet Vignette https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
g2=sample(1:2,100,replace=TRUE)
g4=sample(1:4,100,replace=TRUE)
fit1=glmnet(x,y)
predict(fit1,newx=x[1:5,],s=c(0.01,0.005))
predict(fit1,type="coef")
plot(fit1,xvar="lambda")
fit2=glmnet(x,g2,family="binomial")
predict(fit2,type="response",newx=x[2:5,])
predict(fit2,type="nonzero")
fit3=glmnet(x,g4,family="multinomial")
predict(fit3,newx=x[1:3,],type="response",s=0.01)
```

---

beta_CVX *Simulated data for the glmnet vignette*

Description

Simple simulated data, used to demonstrate the features of glmnet

Usage

```
data(BinomialExample)
data(CVXResults)
data(CoxExample)
data(MultiGaussianExample)
data(MultinomialExample)
data(PoissonExample)
data(QuickStartExample)
data(SparseExample)
```

Format

Data objects used to demonstrate features in the glmnet vignette

Details

Theses datasets are artificial, and ere used to test out some of the features of glmnet.

Examples

```
data(QuickStartExample)
glmnet(x,y)
```

`cv.glmnet` *Cross-validation for glmnet*

Description

Does k-fold cross-validation for glmnet, produces a plot, and returns a value for `lambda`

Usage

```
cv.glmnet(x, y, weights, offset, lambda, type.measure, nfolds, foldid, grouped, keep,
      parallel, ...)
```

Arguments

`x`	`x` matrix as in `glmnet`.
`y`	response `y` as in `glmnet`.
`weights`	Observation weights; defaults to 1 per observation
`offset`	Offset vector (matrix) as in `glmnet`
`lambda`	Optional user-supplied lambda sequence; default is `NULL`, and `glmnet` chooses its own sequence
`nfolds`	number of folds - default is 10. Although `nfolds` can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is `nfolds=3`
`foldid`	an optional vector of values between 1 and `nfold` identifying what fold each observation is in. If supplied, `nfold` can be missing.
`type.measure`	loss to use for cross-validation. Currently five options, not all available for all models. The default is `type.measure="deviance"`, which uses squared-error for gaussian models (a.k.a `type.measure="mse"` there), deviance for logistic and poisson regression, and partial-likelihood for the Cox model. `type.measure="class"` applies to binomial and multinomial logistic regression only, and gives misclassification error. `type.measure="auc"` is for two-class logistic regression only, and gives area under the ROC curve. `type.measure="mse"` or `type.measure="mae"` (mean absolute error) can be used by all models except the `"cox"`; they measure the deviation from the fitted mean to the response.

grouped	This is an experimental argument, with default TRUE, and can be ignored by most users. For all models except the "cox", this refers to computing nfolds separate statistics, and then using their mean and estimated standard error to describe the CV curve. If grouped=FALSE, an error matrix is built up at the observation level from the predictions from the nfold fits, and then summarized (does not apply to type.measure="auc"). For the "cox" family, grouped=TRUE obtains the CV partial likelihood for the Kth fold by *subtraction*; by subtracting the log partial likelihood evaluated on the full dataset from that evaluated on the on the (K-1)/K dataset. This makes more efficient use of risk sets. With grouped=FALSE the log partial likelihood is computed only on the Kth fold
keep	If keep=TRUE, a *prevalidated* array is returned containing fitted values for each observation and each value of lambda. This means these fits are computed with this observation and the rest of its fold omitted. The folid vector is also returned. Default is keep=FALSE
parallel	If TRUE, use parallel foreach to fit each fold. Must register parallel before hand, such as doMC or others. See the example below.
...	Other arguments that can be passed to glmnet

Details

The function runs glmnet nfolds+1 times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The error is accumulated, and the average error and standard deviation over the folds is computed. Note that cv.glmnet does NOT search for values for alpha. A specific value should be supplied, else alpha=1 is assumed by default. If users would like to cross-validate alpha as well, they should call cv.glmnet with a pre-computed vector foldid, and then use this same fold vector in separate calls to cv.glmnet with different values of alpha. Note also that the results of cv.glmnet are random, since the folds are selected at random. Users can reduce this randomness by running cv.glmnet many times, and averaging the error curves.

Value

an object of class "cv.glmnet" is returned, which is a list with the ingredients of the cross-validation fit.

lambda	the values of lambda used in the fits.
cvm	The mean cross-validated error - a vector of length length(lambda).
cvsd	estimate of standard error of cvm.
cvup	upper curve = cvm+cvsd.
cvlo	lower curve = cvm-cvsd.
nzero	number of non-zero coefficients at each lambda.
name	a text string indicating type of measure (for plotting purposes).
glmnet.fit	a fitted glmnet object for the full data.
lambda.min	value of lambda that gives minimum cvm.
lambda.1se	largest value of lambda such that error is within 1 standard error of the minimum.

`fit.preval` if `keep=TRUE`, this is the array of prevalidated fits. Some entries can be `NA`, if that and subsequent values of `lambda` are not reached for that fold

`foldid` if `keep=TRUE`, the fold assignments used

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Noah Simon helped develop the 'coxnet' function.
Jeffrey Wong and B. Narasimhan helped with the parallel option
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, https://web.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/

See Also

`glmnet` and `plot`, `predict`, and `coef` methods for `"cv.glmnet"` object.

Examples

```
set.seed(1010)
n=1000;p=100
nzc=trunc(p/10)
x=matrix(rnorm(n*p),n,p)
beta=rnorm(nzc)
fx= x[,seq(nzc)] %*% beta
eps=rnorm(n)*5
y=drop(fx+eps)
px=exp(fx)
px=px/(1+px)
ly=rbinom(n=length(px),prob=px,size=1)
set.seed(1011)
cvob1=cv.glmnet(x,y)
plot(cvob1)
coef(cvob1)
predict(cvob1,newx=x[1:5,], s="lambda.min")
title("Gaussian Family",line=2.5)
set.seed(1011)
cvob1a=cv.glmnet(x,y,type.measure="mae")
plot(cvob1a)
title("Gaussian Family",line=2.5)
set.seed(1011)
par(mfrow=c(2,2),mar=c(4.5,4.5,4,1))
cvob2=cv.glmnet(x,ly,family="binomial")
```

```
plot(cvob2)
title("Binomial Family",line=2.5)
frame()
set.seed(1011)
cvob3=cv.glmnet(x,ly,family="binomial",type.measure="class")
plot(cvob3)
title("Binomial Family",line=2.5)
## Not run:
set.seed(1011)
cvob3a=cv.glmnet(x,ly,family="binomial",type.measure="auc")
plot(cvob3a)
title("Binomial Family",line=2.5)
set.seed(1011)
mu=exp(fx/10)
y=rpois(n,mu)
cvob4=cv.glmnet(x,y,family="poisson")
plot(cvob4)
title("Poisson Family",line=2.5)

# Multinomial
n=500;p=30
nzc=trunc(p/10)
x=matrix(rnorm(n*p),n,p)
beta3=matrix(rnorm(30),10,3)
beta3=rbind(beta3,matrix(0,p-10,3))
f3=x%*% beta3
p3=exp(f3)
p3=p3/apply(p3,1,sum)
g3=rmult(p3)
set.seed(10101)
cvfit=cv.glmnet(x,g3,family="multinomial")
plot(cvfit)
title("Multinomial Family",line=2.5)
# Cox
beta=rnorm(nzc)
fx=x[,seq(nzc)]%*%beta/3
hx=exp(fx)
ty=rexp(n,hx)
tcens=rbinom(n=n,prob=.3,size=1)# censoring indicator
y=cbind(time=ty,status=1-tcens) # y=Surv(ty,1-tcens) with library(survival)
foldid=sample(rep(seq(10),length=n))
fit1_cv=cv.glmnet(x,y,family="cox",foldid=foldid)
plot(fit1_cv)
title("Cox Family",line=2.5)
# Parallel
require(doMC)
registerDoMC(cores=4)
x = matrix(rnorm(1e5 * 100), 1e5, 100)
y = rnorm(1e5)
system.time(cv.glmnet(x,y))
system.time(cv.glmnet(x,y,parallel=TRUE))

## End(Not run)
```

deviance.glmnet *Extract the deviance from a glmnet object*

Description

Compute the deviance sequence from the glmnet object

Usage

```
## S3 method for class 'glmnet'
deviance(object,...)
```

Arguments

`object`	fitted glmnet object
`...`	additional print arguments

Details

A glmnet object has components `dev.ratio` and `nulldev`. The former is the fraction of (null) deviance explained. The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Null deviance is defined to be 2*(loglike_sat -loglike(Null)); The NULL model refers to the intercept model, except for the Cox, where it is the 0 model. Hence dev.ratio=1-deviance/nulldev, and this `deviance` method returns (1-dev.ratio)*nulldev.

Value

(1-dev.ratio)*nulldev

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*

See Also

`glmnet`, `predict`, `print`, and `coef` methods.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=glmnet(x,y)
deviance(fit1)
```

---

glmnet *fit a GLM with lasso or elasticnet regularization*

Description

Fit a generalized linear model via penalized maximum likelihood. The regularization path is computed for the lasso or elasticnet penalty at a grid of values for the regularization parameter lambda. Can deal with all shapes of data, including very large sparse data matrices. Fits linear, logistic and multinomial, poisson, and Cox regression models.

Usage

```
glmnet(x, y, family=c("gaussian","binomial","poisson","multinomial","cox","mgaussian"),
    weights, offset=NULL, alpha = 1, nlambda = 100,
    lambda.min.ratio = ifelse(nobs<nvars,0.01,0.0001), lambda=NULL,
    standardize = TRUE, intercept=TRUE, thresh = 1e-07,  dfmax = nvars + 1,
    pmax = min(dfmax * 2+20, nvars), exclude, penalty.factor = rep(1, nvars),
    lower.limits=-Inf, upper.limits=Inf, maxit=100000,
    type.gaussian=ifelse(nvars<500,"covariance","naive"),
    type.logistic=c("Newton","modified.Newton"),
    standardize.response=FALSE, type.multinomial=c("ungrouped","grouped"))
```

Arguments

`x`	input matrix, of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class `"sparseMatrix"` as in package `Matrix`; not yet available for `family="cox"`)
`y`	response variable. Quantitative for `family="gaussian"`, or `family="poisson"` (non-negative counts). For `family="binomial"` should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class). For `family="multinomial"`, can be a `nc>=2` level factor, or a matrix with `nc` columns of counts or proportions. For either `"binomial"` or `"multinomial"`, if `y` is presented as a vector, it will be coerced into a factor. For `family="cox"`, `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. The function `Surv()` in package **survival** produces such a matrix. For `family="mgaussian"`, `y` is a matrix of quantitative responses.
`family`	Response type (see above)

`weights` observation weights. Can be total counts if responses are proportion matrices. Default is 1 for each observation

`offset` A vector of length `nobs` that is included in the linear predictor (a `nobs x nc` matrix for the `"multinomial"` family). Useful for the `"poisson"` family (e.g. log of exposure time), or for refining a model by starting at a current fit. Default is `NULL`. If supplied, then values must also be supplied to the `predict` function.

`alpha` The elasticnet mixing parameter, with $0 \leq \alpha \leq 1$. The penalty is defined as

$$(1-\alpha)/2||\beta||_2^2 + \alpha||\beta||_1.$$

`alpha=1` is the lasso penalty, and `alpha=0` the ridge penalty.

`nlambda` The number of `lambda` values - default is 100.

`lambda.min.ratio`
Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs > nvars`, the default is `0.0001`, close to zero. If `nobs < nvars`, the default is `0.01`. A very small value of `lambda.min.ratio` will lead to a saturated fit in the `nobs < nvars` case. This is undefined for `"binomial"` and `"multinomial"` models, and `glmnet` will exit gracefully when the percentage deviance explained is almost 1.

`lambda` A user supplied `lambda` sequence. Typical usage is to have the program compute its own `lambda` sequence based on `nlambda` and `lambda.min.ratio`. Supplying a value of `lambda` overrides this. WARNING: use with care. Avoid supplying a single value for `lambda` (for predictions after CV use `predict()` instead). Supply instead a decreasing sequence of `lambda` values. `glmnet` relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.

`standardize` Logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is `standardize=TRUE`. If variables are in the same units already, you might not wish to standardize. See details below for y standardization with `family="gaussian"`.

`intercept` Should intercept(s) be fitted (default=TRUE) or set to zero (FALSE)

`thresh` Convergence threshold for coordinate descent. Each inner coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than `thresh` times the null deviance. Defaults value is `1E-7`.

`dfmax` Limit the maximum number of variables in the model. Useful for very large `nvars`, if a partial path is desired.

`pmax` Limit the maximum number of variables ever to be nonzero

`exclude` Indices of variables to be excluded from the model. Default is none. Equivalent to an infinite penalty factor (next item).

`penalty.factor` Separate penalty factors can be applied to each coefficient. This is a number that multiplies `lambda` to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables (and implicitly infinity for variables listed in `exclude`). Note: the penalty factors are internally rescaled to sum to nvars, and the lambda sequence will reflect this change.

`lower.limits` Vector of lower limits for each coefficient; default `-Inf`. Each of these must be non-positive. Can be presented as a single value (which will then be replicated), else a vector of length `nvars`

`upper.limits` Vector of upper limits for each coefficient; default `Inf`. See `lower.limits`

`maxit` Maximum number of passes over the data for all lambda values; default is 10^5.

`type.gaussian` Two algorithm types are supported for (only) `family="gaussian"`. The default when `nvar<500` is `type.gaussian="covariance"`, and saves all inner-products ever computed. This can be much faster than `type.gaussian="naive"`, which loops through `nobs` every time an inner-product is computed. The latter can be far more efficient for `nvar >> nobs` situations, or when `nvar > 500`.

`type.logistic` If `"Newton"` then the exact hessian is used (default), while `"modified.Newton"` uses an upper-bound on the hessian, and can be faster.

`standardize.response`
This is for the `family="mgaussian"` family, and allows the user to standardize the response variables

`type.multinomial`
If `"grouped"` then a grouped lasso penalty is used on the multinomial coefficients for a variable. This ensures they are all in our out together. The default is `"ungrouped"`

Details

The sequence of models implied by `lambda` is fit by coordinate descent. For `family="gaussian"` this is the lasso sequence if `alpha=1`, else it is the elasticnet sequence. For the other families, this is a lasso or elasticnet regularization path for fitting the generalized linear regression paths, by maximizing the appropriate penalized log-likelihood (partial likelihood for the "cox" model). Sometimes the sequence is truncated before `nlambda` values of `lambda` have been used, because of instabilities in the inverse link functions near a saturated fit. `glmnet(...,family="binomial")` fits a traditional logistic regression model for the log-odds. `glmnet(...,family="multinomial")` fits a symmetric multinomial model, where each class is represented by a linear model (on the log-scale). The penalties take care of redundancies. A two-class `"multinomial"` model will produce the same fit as the corresponding `"binomial"` model, except the pair of coefficient matrices will be equal in magnitude and opposite in sign, and half the `"binomial"` values. Note that the objective function for `"gaussian"` is

$$1/2RSS/nobs + \lambda * penalty,$$

and for the other models it is

$$-loglik/nobs + \lambda * penalty.$$

Note also that for `"gaussian"`, `glmnet` standardizes y to have unit variance (using 1/n rather than 1/(n-1) formula) before computing its lambda sequence (and then unstandardizes the resulting coefficients); if you wish to reproduce/compare results with other software, best to supply a standardized y. The coefficients for any predictor variables with zero variance are set to zero for all values of lambda. The latest two features in glmnet are the `family="mgaussian"` family and the `type.multinomial="grouped"` option for multinomial fitting. The former allows a multi-response gaussian model to be fit, using a "group -lasso" penalty on the coefficients for each variable. Tying the responses together like this is called "multi-task" learning in some domains. The grouped multinomial allows the same penalty for the `family="multinomial"` model, which is also multi-responsed. For both of these the penalty on the coefficient vector for variable j is

$$(1-\alpha)/2||\beta_j||_2^2+\alpha||\beta_j||_1.$$

When `alpha=1` this is a group-lasso penalty, and otherwise it mixes with quadratic just like elasticnet. A small detail in the Cox model: if death times are tied with censored times, we assume the censored times occurred just *before* the death times in computing the Breslow approximation; if users prefer the usual convention of *after*, they can add a small number to all censoring times to achieve this effect.

Value

An object with S3 class `"glmnet","*"` , where `"*"` is `"elnet"`, `"lognet"`, `"multnet"`, `"fishnet"` (poisson), `"coxnet"` or `"mrelnet"` for the various types of models.

`call`	the call that produced this object
`a0`	Intercept sequence of length `length(lambda)`
`beta`	For `"elnet"`, `"lognet"`, `"fishnet"` and `"coxnet"` models, a `nvars x length(lambda)` matrix of coefficients, stored in sparse column format (`"CsparseMatrix"`). For `"multnet"` and `"mgaussian"`, a list of `nc` such matrices, one for each class.
`lambda`	The actual sequence of `lambda` values used. When `alpha=0`, the largest lambda reported does not quite give the zero coefficients reported (`lambda=inf` would in principle). Instead, the largest `lambda` for `alpha=0.001` is used, and the sequence of `lambda` values is derived from this.
`dev.ratio`	The fraction of (null) deviance explained (for `"elnet"`, this is the R-square). The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Hence dev.ratio=1-dev/nulldev.
`nulldev`	Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)); The NULL model refers to the intercept model, except for the Cox, where it is the 0 model.
`df`	The number of nonzero coefficients for each value of `lambda`. For `"multnet"`, this is the number of variables with a nonzero coefficient for *any* class.
`dfmat`	For `"multnet"` and `"mrelnet"` only. A matrix consisting of the number of nonzero coefficients per class
`dim`	dimension of coefficient matrix (ices)
`nobs`	number of observations
`npasses`	total passes over the data summed over all lambda values
`offset`	a logical variable indicating whether an offset was included in the model
`jerr`	error flag, for warnings and errors (largely for internal debugging).

Author(s)

Jerome Friedman, Trevor Hastie, Noah Simon and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, https://web.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/
Tibshirani, Robert., Bien, J., Friedman, J.,Hastie, T.,Simon, N.,Taylor, J. and Tibshirani, Ryan. (2012) *Strong Rules for Discarding Predictors in Lasso-type Problems, JRSSB vol 74*,
http://statweb.stanford.edu/~tibs/ftp/strong.pdf
Stanford Statistics Technical Report
Glmnet Vignette https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

See Also

`print`, `predict`, `coef` and `plot` methods, and the `cv.glmnet` function.

Examples

```
# Gaussian
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=glmnet(x,y)
print(fit1)
coef(fit1,s=0.01) # extract coefficients at a single value of lambda
predict(fit1,newx=x[1:10,],s=c(0.01,0.005)) # make predictions

#multivariate gaussian
y=matrix(rnorm(100*3),100,3)
fit1m=glmnet(x,y,family="mgaussian")
plot(fit1m,type.coef="2norm")

#binomial
g2=sample(1:2,100,replace=TRUE)
fit2=glmnet(x,g2,family="binomial")

#multinomial
g4=sample(1:4,100,replace=TRUE)
fit3=glmnet(x,g4,family="multinomial")
fit3a=glmnet(x,g4,family="multinomial",type.multinomial="grouped")
#poisson
N=500; p=20
nzc=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(nzc)
f = x[,seq(nzc)]%*%beta
mu=exp(f)
y=rpois(N,mu)
fit=glmnet(x,y,family="poisson")
plot(fit)
```

```
pfit = predict(fit,x,s=0.001,type="response")
plot(pfit,y)

#Cox
set.seed(10101)
N=1000;p=30
nzc=p/3
x=matrix(rnorm(N*p),N,p)
beta=rnorm(nzc)
fx=x[,seq(nzc)]%*%beta/3
hx=exp(fx)
ty=rexp(N,hx)
tcens=rbinom(n=N,prob=.3,size=1)# censoring indicator
y=cbind(time=ty,status=1-tcens) # y=Surv(ty,1-tcens) with library(survival)
fit=glmnet(x,y,family="cox")
plot(fit)

# Sparse
n=10000;p=200
nzc=trunc(p/10)
x=matrix(rnorm(n*p),n,p)
iz=sample(1:(n*p),size=n*p*.85,replace=FALSE)
x[iz]=0
sx=Matrix(x,sparse=TRUE)
inherits(sx,"sparseMatrix")#confirm that it is sparse
beta=rnorm(nzc)
fx=x[,seq(nzc)]%*%beta
eps=rnorm(n)
y=fx+eps
px=exp(fx)
px=px/(1+px)
ly=rbinom(n=length(px),prob=px,size=1)
system.time(fit1<-glmnet(sx,y))
system.time(fit2n<-glmnet(x,y))
```

`glmnet.control` *internal glmnet parameters*

Description

View and/or change the factory default parameters in glmnet

Usage

```
glmnet.control(fdev=1.0e-5, devmax=0.999, eps=1.0e-6, big=9.9e35, mnlam=5, pmin=1.0e-9,
     exmx=250.0,prec=1e-10,mxit=100,factory=FALSE)
```

Arguments

`fdev`	minimum fractional change in deviance for stopping path; factory default = 1.0e-5
`devmax`	maximum fraction of explained deviance for stopping path; factory default = 0.999
`eps`	minimum value of lambda.min.ratio (see glmnet); factory default= 1.0e-6
`big`	large floating point number; factory default = 9.9e35. Inf in definition of upper.limit is set to big
`mnlam`	minimum number of path points (lambda values) allowed; factory default = 5
`pmin`	minimum probability for any class. factory default = 1.0e-9. Note that this implies a pmax of 1-pmin.
`exmx`	maximum allowed exponent. factory default = 250.0
`prec`	convergence threshold for multi response bounds adjustment solution. factory default = 1.0e-10
`mxit`	maximum iterations for multiresponse bounds adjustment solution. factory default = 100
`factory`	If `TRUE`, reset all the parameters to the factory default; default is `FALSE`

Details

If called with no arguments, `glmnet.control()` returns a list with the current settings of these parameters. Any arguments included in the call sets those parameters to the new values, and then silently returns. The values set are persistent for the duration of the R session.

Value

A list with named elements as in the argument list

Author(s)

Jerome Friedman, Trevor Hastie
Maintainer: Trevor Hastie `<hastie@stanford.edu>`

See Also

`glmnet`

Examples

```
glmnet.control(fdev=0)#continue along path even though not much changes
glmnet.control() # view current settings
glmnet.control(factory=TRUE) # reset all the parameters to their default
```

`plot.cv.glmnet` *plot the cross-validation curve produced by cv.glmnet*

Description

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

Usage

```
## S3 method for class 'cv.glmnet'
plot(x, sign.lambda, ...)
```

Arguments

`x`	fitted `"cv.glmnet"` object
`sign.lambda`	Either plot against `log(lambda)` (default) or its negative if `sign.lambda=-1`.
`...`	Other graphical parameters to plot

Details

A plot is produced, and nothing is returned.

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*

See Also

`glmnet` and `cv.glmnet`.

Examples

```
set.seed(1010)
n=1000;p=100
nzc=trunc(p/10)
x=matrix(rnorm(n*p),n,p)
beta=rnorm(nzc)
fx= (x[,seq(nzc)] %*% beta)
eps=rnorm(n)*5
y=drop(fx+eps)
px=exp(fx)
```

```
px=px/(1+px)
ly=rbinom(n=length(px),prob=px,size=1)
cvob1=cv.glmnet(x,y)
plot(cvob1)
title("Gaussian Family",line=2.5)
frame()
set.seed(1011)
par(mfrow=c(2,2),mar=c(4.5,4.5,4,1))
cvob2=cv.glmnet(x,ly,family="binomial")
plot(cvob2)
title("Binomial Family",line=2.5)
set.seed(1011)
cvob3=cv.glmnet(x,ly,family="binomial",type="class")
plot(cvob3)
title("Binomial Family",line=2.5)
```

`plot.glmnet` *plot coefficients from a "glmnet" object*

Description

Produces a coefficient profile plot of the coefficient paths for a fitted `"glmnet"` object.

Usage

```
## S3 method for class 'glmnet'
plot(x, xvar = c("norm", "lambda", "dev"), label = FALSE, ...)
## S3 method for class 'multnet'
plot(x, xvar = c("norm", "lambda", "dev"), label = FALSE,type.coef=c("coef","2norm"), ...)
## S3 method for class 'mrelnet'
plot(x, xvar = c("norm", "lambda", "dev"), label = FALSE,type.coef=c("coef","2norm"), ...)
```

Arguments

`x`	fitted `"glmnet"` model
`xvar`	What is on the X-axis. `"norm"` plots against the L1-norm of the coefficients, `"lambda"` against the log-lambda sequence, and `"dev"` against the percent deviance explained.
`label`	If `TRUE`, label the curves with variable sequence numbers.
`type.coef`	If `type.coef="2norm"` then a single curve per variable, else if `type.coef="coef"`, a coefficient plot per response
`...`	Other graphical parameters to plot

Details

A coefficient profile plot is produced. If `x` is a multinomial model, a coefficient plot is produced for each class.

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*

See Also

glmnet, and print, predict and coef methods.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
g2=sample(1:2,100,replace=TRUE)
g4=sample(1:4,100,replace=TRUE)
fit1=glmnet(x,y)
plot(fit1)
plot(fit1,xvar="lambda",label=TRUE)
fit3=glmnet(x,g4,family="multinomial")
plot(fit3,pch=19)
```

---

predict.cv.glmnet *make predictions from a "cv.glmnet" object.*

Description

This function makes predictions from a cross-validated glmnet model, using the stored "glmnet.fit" object, and the optimal value chosen for lambda.

Usage

```
## S3 method for class 'cv.glmnet'
predict(object, newx, s=c("lambda.1se","lambda.min"),...)
## S3 method for class 'cv.glmnet'
coef(object,s=c("lambda.1se","lambda.min"),...)
```

Arguments

object	Fitted "cv.glmnet" object.
newx	Matrix of new values for x at which predictions are to be made. Must be a matrix; can be sparse as in Matrix package. See documentation for predict.glmnet.

`s`	Value(s) of the penalty parameter `lambda` at which predictions are required. Default is the value `s="lambda.1se"` stored on the CV `object`. Alternatively `s="lambda.min"` can be used. If `s` is numeric, it is taken as the value(s) of `lambda` to be used.
`...`	Not used. Other arguments to predict.

Details

This function makes it easier to use the results of cross-validation to make a prediction.

Value

The object returned depends the ... argument which is passed on to the `predict` method for `glmnet` objects.

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent, Journal of Statistical Software, Vol. 33, Issue 1, Feb 2010*
http://www.jstatsoft.org/v33/i01/

See Also

`glmnet`, and `print`, and `coef` methods, and `cv.glmnet`.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
cv.fit=cv.glmnet(x,y)
predict(cv.fit,newx=x[1:5,])
coef(cv.fit)
coef(cv.fit,s="lambda.min")
predict(cv.fit,newx=x[1:5,],s=c(0.001,0.002))
```

`predict.glmnet` *make predictions from a "glmnet" object.*

Description

Similar to other predict methods, this functions predicts fitted values, logits, coefficients and more from a fitted `"glmnet"` object.

Usage

```
## S3 method for class 'glmnet'
predict(object, newx, s = NULL,
type=c("link","response","coefficients","nonzero","class"), exact = FALSE, newoffset, ...)
## S3 method for class 'glmnet'
coef(object,s=NULL, exact=FALSE, ...)
```

Arguments

- `object` — Fitted "glmnet" model object.
- `newx` — Matrix of new values for `x` at which predictions are to be made. Must be a matrix; can be sparse as in `Matrix` package. This argument is not used for `type=c("coefficients","nonzero")`
- `s` — Value(s) of the penalty parameter `lambda` at which predictions are required. Default is the entire sequence used to create the model.
- `type` — Type of prediction required. Type "link" gives the linear predictors for "binomial", "multinomial", "poisson" or "cox" models; for "gaussian" models it gives the fitted values. Type "response" gives the fitted probabilities for "binomial" or "multinomial", fitted mean for "poisson" and the fitted relative-risk for "cox"; for "gaussian" type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for `s`. Note that for "binomial" models, results are returned only for the class corresponding to the second level of the factor response. Type "class" applies only to "binomial" or "multinomial" models, and produces the class label corresponding to the maximum probability. Type "nonzero" returns a list of the indices of the nonzero coefficients for each value of `s`.
- `exact` — This argument is relevant only when predictions are made at values of `s` (lambda) *different* from those used in the fitting of the original model. If `exact=FALSE` (default), then the predict function uses linear interpolation to make predictions for values of `s` (lambda) that do not coincide with those used in the fitting algorithm. While this is often a good approximation, it can sometimes be a bit coarse. With `exact=TRUE`, these different values of `s` are merged (and sorted) with `object$lambda`, and the model is refit before predictions are made. In this case, it is required to supply the original data `x=` and `y=` as additional named arguments to `predict()` or `coef()`. The workhorse `predict.glmnet()` needs to `update` the model, and so needs the data used to create it. The same is true of `weights`, `offset`, `penalty.factor`, `lower.limits`, `upper.limits` if these were used in the original call. Failure to do so will result in an error.
- `newoffset` — If an offset is used in the fit, then one must be supplied for making predictions (except for `type="coefficients"` or `type="nonzero"`)
- `...` — This is the mechanism for passing arguments like `x=` when `exact=TRUE`; see`exact` argument.

Details

The shape of the objects returned are different for "multinomial" objects. This function actually calls `NextMethod()`, and the appropriate predict method is invoked for each of the three model types. `coef(...)` is equivalent to `predict(type="coefficients",...)`

Value

The object returned depends on type.

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*, https://web.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010
http://www.jstatsoft.org/v33/i01/
Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011) *Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5) 1-13*
http://www.jstatsoft.org/v39/i05/

See Also

`glmnet`, and `print`, and `coef` methods, and `cv.glmnet`.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
g2=sample(1:2,100,replace=TRUE)
g4=sample(1:4,100,replace=TRUE)
fit1=glmnet(x,y)
predict(fit1,newx=x[1:5,],s=c(0.01,0.005))
predict(fit1,type="coef")
fit2=glmnet(x,g2,family="binomial")
predict(fit2,type="response",newx=x[2:5,])
predict(fit2,type="nonzero")
fit3=glmnet(x,g4,family="multinomial")
predict(fit3,newx=x[1:3,],type="response",s=0.01)
```

`print.glmnet` *print a glmnet object*

Description

Print a summary of the glmnet path at each step along the path.

Usage

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

Arguments

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

Details

The call that produced the object `x` is printed, followed by a three-column matrix with columns `Df`, `%dev` and `Lambda`. The `Df` column is the number of nonzero coefficients (Df is a reasonable name only for lasso fits). `%dev` is the percent deviance explained (relative to the null deviance).

Value

The matrix above is silently returned

Author(s)

Jerome Friedman, Trevor Hastie and Rob Tibshirani
Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent*

See Also

`glmnet`, `predict` and `coef` methods.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=glmnet(x,y)
print(fit1)
```

Index