Package ‘c060’

Version 0.2-4
Date 2014-12-09
Author Martin Sill, Thomas Hielscher, Manuela Zucknick, Natalia Becker.
Maintainer Martin Sill <m.sill@dkfz.de>
Title Extended Inference for Lasso and Elastic-Net Regularized Cox and
Generalized Linear Models
Depends
Imports glmnet, survival, parallel, mlegp, tgp, peperr, penalizedSVM,
lattice
Suggests
Description c060 provides additional functions to perform stability selection, model valida-
tion and parameter tuning for glmnet models
License GPL-2
LazyLoad yes
Repository CRAN
Repository/R-Forge/Project c060
Repository/R-Forge/Revision 185
Repository/R-Forge/DateTimeStamp 2014-12-11 13:09:54
Date/Publication 2014-12-11 17:04:36
NeedsCompilation no

R topics documented:

aggregation.auc ................................................. 2
balancedFolds .................................................. 4
coeq.sum.intsearch ........................................... 5
complexity.glmnet ............................................. 5
epsgo ............................................................ 7
fit.glmnet ....................................................... 11
PLL.coxnet ...................................................... 12
Determine the area under the ROC curve for a fitted model

Description
Evaluate the area under the ROC curve for a fitted model on new data. To be used as argument aggregation.fun in peperr call.

Usage
aggregation.auc(full.data=NULL, response, x, model, cplx=NULL, type=c("apparent", "noinf"), fullsample.attr = NULL, ...)

Arguments
full.data passed from peperr, but not used for calculation.
response vector of binary response.
x n*p matrix of covariates.
model model fitted as returned by a fit.fun, as used in a call to peperr.
cplx passed from peperr, but not necessary for calculation.
type character.
fullsample.attr passed from peperr, but not necessary for calculation.
... additional arguments, passed to predict function.

Details
Area under the ROC curve is calculated based on internal glmnet:::auc function from package glmnet.

Value
Scalar, indicating the area under the ROC curve.
Author(s)

Thomas Hielscher <t.hielscher@dkfz.de>

See Also

peperr

Examples

```r

## Not run:
# binomial model - classification

library(c060)
library(gridExtra)
library(ggplot2)

set.seed(0815)
x <- matrix(rnorm(100*20),100,20)
y <- sample(0:1,100,replace=TRUE)

peperr_obj <- peperr(response=y, x=x, fit.fun=fit.glmnet, args.fit=list(family="binomial"),
  complexity=complexity.glmnet, args.complexity=list(nfolds=10, family="binomial"),
  trace=F, RNG="fixed", seed=0815,
  # aggregation.fun=c060:::aggregation.misclass,
  # aggregation.fun=c060:::aggregation.brier,
  aggregation.fun=c060:::aggregation.auc,
  indices=resample.indices(n=nrow(x), sample.n = 100, method = "sub632")

tmp <- data.frame(grp=, error=unlist(peperr_obj$sample.error))
errs <- data.frame(error=c(perr(peperr_obj,"resample"),
  perr(peperr_obj,"632p"),perr(peperr_obj,"app"),
  perr(peperr_obj,"nullmodel")), col = c("red","blue","green","brown"),
  row.names=c("mean\out-of-bag","632plus","apparent","null model"))

p <- ggplot(tmp, aes(grp, error))
p <- p + geom_boxplot(outlier.colour = rgb(0,0,0,0), outlier.size=0) +
  geom_jitter(position=position_jitter(width=.1)) +
  theme_bw() + scale_y_continuous("AUC") + scale_x_discrete("") +
  geom_hline(aes(yintercept=error, colour=col), data=errs, show_guide=T) +
  scale_colour_identity("error type", guide = "legend", breaks=errs$col, labels=rownames(errs)) +
  ggtitle("AUC \nin bootstrap samples \n")

p2 <- ggplot(data.frame(complx=peperr_obj$sample.complexity), aes(x=complx))
p2 <- p2 + geom_histogram(binwidth = 0.02, fill = "white", colour="black") +
  theme_bw()+ xlab(expression(lambda)) +
  ylab("frequency") +
  geom_vline(xintercept=peperr_obj$selected.complexity, colour="red") +
  ggtitle("Selected complexity \nin bootstrap samples") +
  ggplot2::annotate("text", x = 0.12, y = -0.5,
  label = "full data", colour="red", size=4)
```
balancedFolds

grid.arrange(pg2, pg, ncol=2)

## End(Not run)

### balancedFolds

Function producing stratified/balanced folds for cross validation

**Description**

Get balanced folds for cross validation, which are used for tuning penalization parameters.

**Usage**

`balancedFolds(class.column.factor, cross.outer)`

**Arguments**

- `class.column.factor`:
  class labels of length n.
- `cross.outer`:
  number of folds.

**Value**

- `permutated.cut`:
  vector of length n, indicating the fold belongs to.
- `model`:
  model list.
  - `alpha` - optimal alpha.
  - `lambda` - optimal lambda.
  - `nfolds` - cross-validation's folds.
  - `cvreg` - cv.glmnet object for optimal alpha.
  - `fit` - glmnet object for optimal alpha and optimal lambda.

**Author(s)**

Natalia Becker natalia.becker at dkfz.de

**References**


**See Also**

EPSG0
**coef.sum.intsearch**

Get coefficients for a model

**Description**

Get coefficients for a model after applying interval search for tuning parameters.

**Usage**

```r
## S3 method for class 'sum.intsearch'
coef(object, ...)
```

**Arguments**

- `object`: an object as returned by the function `summary.intsearch`.
- `...`: additional argument(s)

**Value**

named vector of non-zero coefficients for the optimal lambda

**Author(s)**

Natalia Becker <natalia.becker@dkfz.de>

**References**


**See Also**

`EPSGO`, `summary.intsearch`, `plot.sum.intsearch`

---

**complexity.glmnet**

Interface for determination of penalty lambda in penalized regression model via cross-validation

**Description**

Determines the amount of shrinkage for a penalized regression model fitted by glmnet via cross-validation, conforming to the calling convention required by argument `complexity` in `peperr` call.

**Usage**

```r
complexity.glmnet(response, x, full.data, ...)
```
Arguments

response  a survival object (with Surv(time, status), or a binary vector with entries 0 and 1).

x  n*p matrix of covariates.

full.data  data frame containing response and covariates of the full data set.

...  additional arguments passed to cv.glmnet call such as family.

Details

Function is basically a wrapper for cv.glmnet of package glmnet. A n-fold cross-validation (default n=10) is performed to determine the optimal penalty lambda. For Cox PH regression models the deviance based on penalized partial log-likelihood is used as loss function. For binary endpoints other loss functions are available as well (see type.measure). Deviance is default. Calling peperr, the default arguments of cv.glmnet can be changed by passing a named list containing these as argument args.complexity. Note that only penalized Cox PH (family="cox") and logistic regression models (family="binomial") are sensible for prediction error evaluation with package peperr.

Value

Scalar value giving the optimal lambda.

Author(s)

Thomas Hielscher \texttt{t.hielscher@dkfz.de}

References


See Also

peperr, cv.glmnet
Description

Finds an optimal solution for the Q.func function.

Usage

epsgo(Q.func, bounds, round.n=5, parms.coding="none",
      fminlower=0, flag.find.one.min =FALSE,
      show=c("none", "final", "all"), N= NULL, maxevals = 500,
      pdf.name=NULL, pdf.width=12, pdf.height=12, my.mfrow=c(1,1),
      verbose=TRUE, seed=123, ... )

Arguments

- Q.func: name of the function to be minimized.
- bounds: bounds for parameters
- round.n: number of digits after comma, default: 5
- parms.coding: parmers coding: none or log2, default: none.
- fminlower: minimal value for the function Q.func, default is 0.
- flag.find.one.min: do you want to find one min value and stop? Default: FALSE
- show: show plots of DIRECT algorithm: none, final iteration, all iterations. Default: none
- N: define the number of start points, see details.
- maxevals: the maximum number of DIRECT function evaluations, default: 500.
- pdf.name: pdf name
- pdf.width: default: 12
- pdf.height: default: 12
- my.mfrow: default: c(1,1)
- verbose: verbose? default: TRUE.
- seed: seed
- ...: additional argument(s)

Details

if the number of start points (N) is not defined by the user, it will be defined dependent on the dimensionality of the parameter space. N=10D+1, where D is the number of parameters, but for high dimensional parameter space with more than 6 dimensions, the initial set is restricted to 65. However for one-dimensional parameter space the N is set to 21 due to stability reasons.
The idea of EPSGO (Efficient Parameter Selection via Global Optimization): Beginning from an initial Latin hypercube sampling containing N starting points we train an Online GP, look for the point with the maximal expected improvement, sample there and update the Gaussian Process(GP). Thereby it is not so important that GP really correctly models the error surface of the SVM in parameter space, but that it can give us information about potentially interesting points in parameter space where we should sample next. We continue with sampling points until some convergence criterion is met.

DIRECT is a sampling algorithm which requires no knowledge of the objective function gradient. Instead, the algorithm samples points in the domain, and uses the information it has obtained to decide where to search next. The DIRECT algorithm will globally converge to the maximal value of the objective function. The name DIRECT comes from the shortening of the phrase 'DIviding RECTangles', which describes the way the algorithm moves towards the optimum.

The code source was adopted from MATLAB originals, special thanks to Holger Froehlich.

Value

- fmin: minimal value of Q.func on the interval defined by bounds.
- xmin: corresponding parameters for the minimum
- iter: number of iterations
- neval: number of visited points
- maxevals: the maximum number of DIRECT function evaluations
- seed: seed
- bounds: bounds for parameters
- Q.func: name of the function to be minimized.
- points.fmin: the set of points with the same fmin
- Xtrain: visited points
- Ytrain: the output of Q.func at visited points Xtrain
- gp.seed: seed for Gaussian Process
- model.list: detailed information of the search process

Author(s)

Natalia Becker natalia.becker at dkfz.de

References


### Examples

```r
## Not run:
set.seed(1010)
n = 1000; p = 100
nc = trunc(p/10)
x = matrix(rnorm(n*p), n, p)
beta = rnorm(nc)
fx = x[, seq(nc)] * 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)

nfolds = 10
set.seed(1234)
foldid <- balancedFolds(class.column.factor = y.classes, cross.outer = nfolds)

# y - binomial
y.classes <- ifelse(y >= median(y), 1, 0)
bound <- t(data.frame(alpha = c(0, 1)))
colnames(bounds) <- c("lower", "upper")

fit <- epsgo(Q.func = "tune.glmnet.interval",
            bounds = bounds,
            parms.coding = "none",
            seed = 1234,
            show = "none",
            fminlower = -100,
            x = x, y = y.classes, family = "binomial",
            foldid = foldid,
            type.min = "lambda.1se",
            type.measure = "mse")
summary(fit)

# y - multinomial: low - low 25%, middle - (25,75)-quantiles, high - larger 75%.
y.classes <- ifelse(y <= quantile(y, 0.25), 1, ifelse(y >= quantile(y, 0.75), 3, 2))
bound <- t(data.frame(alpha = c(0, 1)))
colnames(bounds) <- c("lower", "upper")

fit <- epsgo(Q.func = "tune.glmnet.interval",
            bounds = bounds,
            parms.coding = "none",
            seed = 1234,
            show = "none",
            fminlower = -100,
            x = x, y = y.classes, family = "multinomial",
            foldid = foldid,
            type.min = "lambda.1se",
            type.measure = "mse")
```

summary(fit)

#poisson
N=500; p=20
nz=5
x=matrix(rnorm(N*p),N,p)
beta=rnorm(nz)
f = x[,seq(nz)]
mu=exp(f)
y.classes=rpois(N,mu)

nfolds = 10
set.seed(1234)
foldid <- balancedFolds(class.column.factor=y.classes, cross.outer=nfolds)

fit <- epsgo(Q.func="tune.glmnet.interval",
  bounds=bounds,
  parms.coding="none",
  seed = 1234,
  show="none",
  fminlower = -100,
  x = x, y = y.classes, family = "poisson",
  foldid = foldid,
  type.min = "lambda.1se",
  type.measure = "mse")
summary(fit)

#gaussian
set.seed(1234)
x=matrix(rnorm(100*1000,0,1),100,1000)
y <- x[1:100,1:1000]%*%c(rep(2.5),rep(-2.5),rep(.1990))
foldid <- rep(1:10,each=10)

fit <- epsgo(Q.func="tune.glmnet.interval",
  bounds=bounds,
  parms.coding="none",
  seed = 1234,
  show="none",
  fminlower = -100,
  x = x, y = y, family = "gaussian",
  foldid = foldid,
  type.min = "lambda.1se",
  type.measure = "mse")
summary(fit)

# y - cox in vinette

## End(Not run)
fit.glmnet

Interface function for fitting a penalized regression model with glmnet

Description

Interface for fitting penalized regression models for binary of survival endpoint using glmnet, conforming to the requirements for argument fit.fun in peperr call.

Usage

```r
fit.glmnet(response, x, cplx, ...)
```

Arguments

- `response`: a survival object (with `Surv(time, status)`, or a binary vector with entries 0 and 1).
- `x`: `n*p` matrix of covariates.
- `cplx`: lambda penalty value.
- `...`: additional arguments passed to glmnet call such as family.

Details

Function is basically a wrapper for glmnet of package glmnet. Note that only penalized Cox PH (family="cox") and logistic regression models (family="binomial") are sensible for prediction error evaluation with package peperr.

Value

glmnet object

Author(s)

Thomas Hielscher \<t.hielscher@dkfz.de>

References


See Also

peperr, glmnet

PLL.coxnet

Predictive partial log-likelihood for glmnet Cox PH model fit

Description

Extracts the predictive partial log-likelihood from a glmnet Cox PH model fit.

Usage

## S3 method for class 'coxnet'
PLL(object, newdata, newtime, newstatus, complexity, ...)

Arguments

- **object**: fitted model of class coxnet.
- **newdata**: n_new*p matrix of covariates.
- **newtime**: n_new-vector of censored survival times.
- **newstatus**: n_new-vector of survival status, coded with 0 and 1
- **complexity**: lambda penalty value.
- **...**: additional arguments, not used.

Details

Used by function peperr, if function fit glmnet and family="cox" is used for model fit, which gives a class coxnet object. This is basically a wrapper based on the coxnet.deviance function from package glmnet.

Value

Vector of length n_new

Author(s)

Thomas Hielscher <t.hielscher@dkfz.de>

References

Plot.coef.glmnet

function to highlight the path of a pre-specified set of variables within the coefficient path

Description

Creates several plots showing the coefficient path for the final model of a cv.glmnet fit and highlights the path of a pre-specified set of variables within the coefficient path.

Usage

Plot.coef.glmnet(cvfit, betas)

Arguments

cvfit
  an object of class "cv.glmnet" as returned by the function cv.glmnet.
betas
  a vector of names of variables; must be a subset of rownames(coef(cvfit)).

Value

a list of four objects

stable
  a vector giving the positions of the estimated stable variables
lambda
  the penalization parameter used for the stability selection
lpos
  the position of the penalization parameter in the regularization path
error
  the desired type I error level w.r.t. to the chosen type I error rate
type
  the type I error rate

Author(s)

Manuela Zucknick <m.zucknick@dkfz-heidelberg.de>

References


Examples

## Not run:
set.seed(1010)
n=1000;p=100
ncz=trunc(p/10)
x=matrix(rnorm(n*p),n,p)
beta=rnorm(ncz)
fx= x[,seq(ncz)] %*% beta
Plot.peperr.curves

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(101)
cvob= cv.glmnet(x,y)
Plot.coef.glmnet(cvob, c("V1","V100"))

## End(Not run)

Plot.peperr.curves  Plot method for prediction error curves of a peperr object

Description

Plots individual and aggregated prediction error estimates based on bootstrap samples.

Usage

Plot.peperr.curves(x, at.risk=TRUE, allErrors=FALSE, bootRuns=FALSE, bootQuants=TRUE, bootQuants.level=0.95, leg.cex=0.7,...)

Arguments

x  peperr object.
at.risk  number at risk to be display. default is TRUE.
allErrors  Display .632, no information and average out-of-bag error in addition. default is FALSE.
bootRuns  Display individual out-of-bag bootstrap samples. default is FALSE.
bootQuants  Display pointwise out-of-bag bootstrap quantiles as shaded area. default is TRUE.
bootQuants.level  Quantile probabilities for pointwise out-of-bag bootstrap quantiles. default is 0.95, i.e. 2.5% and 97.5% quantiles.
leg.cex  size of legend text
...  additional arguments, not used.

Details

This function is literally taken from plot.peperr in the peperr package. The display of prediction error curves is adapted to allow for numbers at risk and pointwise bootstrap quantiles.

Author(s)

Thomas Hielscher <t.hielscher@dkfz.de>
plot.stabpath

References


See Also

peperr

Examples

## Not run:

# example from glmnet package
set.seed(10101)
library(glmnet)
library(survival)
library(peperr)

N=1000;p=30
ncp=p/3
x=matrix(rnorm(N*p),N,p)
beta=rnorm(ncp)
fx=x[,seq(ncp)]
hx=exp(fx)
ty=rexp(N,hx)
tcens=rbinom(n=N,prob=.3,size=1)# censoring indicator
y=Surv(ty,1-tcens)

peperr.object <- peperr(response=y, x=x,
                         fit.fun=fit.glmnet, args.fit=list(family="cox"),
                         complexity=complexity.glmnet, 
                         args.complexity=list(family="cox",nfolds=10), 
                         indices=resample.indices(n=N, method="sub632", sample.n=10))

# pointwise bootstrap quantiles and all error types
Plot.peperr.curves(peperr.object, allErrors=TRUE)

# individual bootstrap runs and selected error types
Plot.peperr.curves(peperr.object, allErrors=FALSE, bootRuns=TRUE)

## End(Not run)

plot.stabpath  function to plot a stability path
Description

Given a desired family-wise error rate (FWER) and a stability path calculated with `stabpath`, the function selects an stable set of features and plots the stability path and the corresponding regularization path.

Usage

```r
## S3 method for class 'stabpath'
plot(x, error=0.05, type=c("pfer","pcer"), pi_thr=0.6, xvar=c("lambda","norm","dev"),
     col.all="black", col.sel="red", ...)
```

Arguments

- `x`: an object of class "stabpath" as returned by the function `stabpath`.
- `error`: the desired type I error level w.r.t. to the chosen type I error rate.
- `type`: The type I error rate used for controlling the number falsely selected variables. If type="pfer" the per-family error rate is controlled and `error` corresponds to the expected number of type I errors. Selecting type="pfer" and `error` in the range of 0 > `error` < 1 will control the family-wise error rate, i.e. the probability that at least one variable in the estimated stable set has been falsely selected. If type="pcer" the per-comparison error rate is controlled and `error` corresponds to the expected number of type I errors divided by the number variables.
- `pi_thr`: the threshold used for the stability selection, should be in the range of 0.5 > `pi_thr` < 1.
- `xvar`: the variable used for the xaxis, e.g. for "lambda" the selection probabilities are plotted along the log of the penalization parameters, for "norm" along the L1-norm and for "dev" along the fraction of explained deviance.
- `col.all`: the color used for the variables that are not in the estimated stable set
- `col.sel`: the color used for the variables in the estimated stable set
- `...`: further arguments that are passed to `matplot`

Value

a list of four objects

- `stable`: a vector giving the positions of the estimated stable variables
- `lambda`: the penalization parameter used for the stability selection
- `lpos`: the position of the penalization parameter in the regularization path
- `error`: the desired type I error level w.r.t. to the chosen type I error rate
- `type`: the type I error rate

Author(s)

Martin Sill \(<m.sill@dkfz.de>\)


References


See Also

stabsel, stabpath

Examples

```r
## Not run:
gaussian
set.seed(1234)
x = matrix(rnorm(100*1000,0,1),100,1000)
y <- x[,1:100,1:1000]*x[,2:1000]*x[,rep(2,5),rep(-2,5),rep(.1,990)]
res <- stabpath(y,x,weakness=1,mc.cores=2)
plot(res,error=.5,type='pfer')
```

## End(Not run)

plot.sum.intsearch  

Plot Summary object for interval search models

Description

Produces a plot for summary object of a fitted interval search model. Plot 'visited' points against iteration steps. start.N points are initial points selected before interval search starts.

Usage

## S3 method for class 'sum.intsearch'
plot(x,type="summary",startN=21,... )

Arguments

x  an object of class sum.intsearch as returned by the function summary.intsearch.
type  type of plot to be drawn. type="summary" will plot the partial log likelihood deviance as a function of both tuning parameters \( \alpha \) and log \( \lambda \). The final solution will be highlighted by solid red line. Alternatively, type="points" will draw the distribution of initial and visited points of the interval search plotted in chronological order.
startN  number of initial points. Needed if type="points"
...  additional argument(s)
predictProb.coxnet

Author(s)
Natalia Becker
<natalia.becker@dkfz.de>

References

See Also
EPSGO, summary.intsearch

predictProb.coxnet Extract predicted survival probabilities from a glmnet fit

Description
Extracts predicted survival probabilities from survival model fitted by glmnet, providing an interface as required by pmpec.

Usage
## S3 method for class 'coxnet'
predictProb(object, response, x, times, complexity, ...)

Arguments
object a fitted model of class glmnet
response 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
x n*p matrix of covariates.
times vector of evaluation time points.
complexity lambda penalty value.
... additional arguments, currently not used.

Value
Matrix with probabilities for each evaluation time point in times (columns) and each new observation (rows).

Author(s)
Thomas Hielscher
<t.hielscher@dkfz.de>
predictProb.glmnet

References

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

http://www.jstatsoft.org/v39/i05/


See Also

predictProb.glmnet, peperr, glmnet

predictProb.glmnet Extract predicted survival probabilities from a glmnet fit

Description

Extracts predicted survival probabilities from survival model fitted by glmnet, providing an interface as required by pmpec.

Usage

## S3 method for class 'glmnet'
predictProb(object, response, x, times, complexity, ...)

Arguments

object a fitted model of class glmnet.
response 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. For family="multinomial", can be a nc>=2 level factor, or a matrix with nc columns of counts or proportions.
x n*p matrix of covariates.
times vector of evaluation time points.
complexity lambda penalty value.
... additional arguments, currently not used.
Value

Matrix with probabilities for each evaluation time point in times (columns) and each new observation (rows).

Author(s)

Thomas Hielscher <t.hielscher@dkfz.de>

References

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

See Also

predictprob.coxnet, peperr, glmnet

Stability path for glmnet models

Description

The function calculates the stability path for glmnet models, e.g. the selection probabilities of the features along the range of regularization parameters.

Usage

stabpath(y,x,size=0.632,steps=100,weakness=1,mc.cores=getOption("mc.cores", 2L),...)

Arguments

y response variable. Like for the glmnet function: 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. For family="multinomial", can be a nc>=2 level factor, or a matrix with nc columns of counts or proportions. 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

x

input matrix. Like for the glmnet function: of dimension nob 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")

size

proportion of samples drawn in every subsample used for the stability selection.

steps

number of subsamples used for the stability selection.

weakness

weakness parameter used for the randomised lasso as described in Meinshausen and Bühlmann (2010). For each subsample the features are reweighted by a random weight uniformly sampled in [weakness,1]. This additional randomisation leads to a more consistent estimation of the stable set of features.

mc.cores

number of cores used for the parallelization. For unix like system the parallelization is done by forking using the function `mclapply`. For windows systems socket cluster are used.

... further arguments that are passed to the glmnet function.

Value

an object of class "stabpath", which is a list of three objects

fit

the fit object of class "glmnet" as returned from the glmnet function when applied to the complete data set.

stabpath

a matrix which represents the stability path.

qs

a vector holding the values of the average number of non-zero coefficients w.r.t to the lambdas in the regularization path.

Author(s)

Martin Sill <m.sill@dkfz.de>

References


See Also

glmnet, stabsel, plot.stabpath
Examples

```r
## Not run:
# gaussian
set.seed(1234)
x <- matrix(rnorm(100*1000,0,1),100,1000)
y <- x[1:100,1:1000] c(rep(2.5),rep(-2.5),rep(.1,990))
res <- stabpath(y,x,weakness=1,mc.cores=2)
plot(res)

# binomial
y = sample(1:2,100,replace=TRUE)
res <- stabpath(y,x,weakness=1,mc.cores=2,family="binomial")
plot(res)

# multinomial
y = sample(1:4,100,replace=TRUE)
res <- stabpath(y,x,weakness=1,mc.cores=2,family="multinomial")
plot(res)

# poisson
N=100; p=1000
nzc=5
x = matrix(rnorm(N*p),N,p)
beta = rnorm(nzc)
f = x[,seq(nzc)] %*% beta
mu = exp(f)
y = rpois(N,mu)
res <- stabpath(y,x,weakness=1,mc.cores=2,family="poisson")
plot(res)

# Cox
library(survival)
set.seed(10101)
N=100; p=1000
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)
y = cbind(time=ty,status=1-tcens)
res <- stabpath(y,x,weakness=1,mc.cores=2,family="cox")
plot(res)

## End(Not run)
```

stabsel

*function to estimate a stable set of variables*
**Description**

Given a desired type I error rate and a stability path calculated with `stability.path` the function selects a stable set of variables.

**Usage**

```r
stabsel(x, error=0.05, type=c("pfer","pcer"), pi_thr=0.6)
```

**Arguments**

- `x`: an object of class "stabpath" as returned by the function `stabpath`.
- `error`: the desired type I error level w.r.t. to the chosen type I error rate.
- `type`: The type I error rate used for controlling the number falsely selected variables. If `type="pfer"` the per-family error rate is controlled and `error` corresponds to the expected number of type I errors. Selecting `type="pfer"` and `error` in the range of $0 > error < 1$ will control the family-wise error rate, i.e. the probability that at least one variable in the estimated stable set has been falsely selected. If `type="pcer"` the per-comparison error rate is controlled and `error` corresponds to the expected number of type I errors divided by the number variables.
- `pi_thr`: the threshold used for the stability selection, should be in the range of $0.5 > pi_thr < 1$.

**Value**

A list of four objects

- `stable`: a vector giving the positions of the estimated stable variables
- `lambda`: the penalization parameter used for the stability selection
- `lpos`: the position of the penalization parameter in the regularization path
- `error`: the desired type I error level w.r.t. to the chosen type I error rate
- `type`: the type I error rate

**Author(s)**

Martin Sill <m.sill@dkfz.de>

**References**


See Also

plot.stabpath, stabpath

Examples

## Not run:
#gaussian
set.seed(1234)
x = matrix(rnorm(100*1000, 0, 1), 100, 1000)
y <- x[1:100, 1:1000] %*% Sc(rep(2, 5), rep(-2, 5), rep(.1, 990))
res <- stabpath(y, x, weakness = 1, mc.cores = 2)
stabsel(res, error = 0.05, type = "pfer")

## End(Not run)

summary.intsearch

Summary method for interval search models

### Description

Produces a summary of a fitted interval search model

### Usage

```
## S3 method for class 'intsearch'
summary(object, digits = max(3, getOption("digits") - 3), verbose = TRUE, first.n = 5, ...)
```

### Arguments

- `object`: an object of class `intsearch` as returned by the function `EPSSG0`.
- `digits`: digits after the comma
- `verbose`: default set to `TRUE`.
- `first.n`: show first.n entries, default 5.
- `...`: additional argument(s)

### Value

A list of following elements

- `info`: a data frame of four objects for optimal models
  - `alpha`: a vector of alphas
  - `lambda`: a vector of penalization parameter lambda
  - `deviances`: a vector of deviances
  - `n.features`: a vector of number of features selected in each optimal model

- `opt.alpha`: an optimal value for alpha
tune.glmnet.interval

- **opt.lambda**: an optimal value for lambda
- **opt.error**: an optimal value for error, hier minimal diviance
- **opt.models**: a list of optimal models with the same optimal error

**Author(s)**

Natalia Becker <natalia.becker@dkfz.de>

**References**


**See Also**

[EPSGO.plot.sum.intsearch](#)

---

**tune.glmnet.interval**

Wrapper function for glmnet objects.

**Description**

Wrapper function for glmnet objects used by epsgo function. This function is mainly used within the function epsgo.

**Usage**

tune.glmnet.interval(parms, x, y,
weights,
offset = NULL,
lambda = NULL,
type.measure = c("mse", "deviance", "class", "auc", "mae"),
seed=12345,
nfolds = 10,
foldid=NULL,
grouped = TRUE,
type.min=c("lambda.min", "lambda.1se"),
family,
verbose=FALSE,
...)

**Arguments**

- **parms**: tuning parameter alpha for glmnet object
- **x, y**: x is a matrix where each row refers to a sample and each column refers to a gene; y is a factor which includes the class for each sample
- **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.
- **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. Do not supply 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.
- **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.
- **seed**: seed
- **nfolds**: number of cross-validation's folds, default 10.
- **foldid**: an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing.
- **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
- **type.min**: parameter for choosing optimal model: 'lambda.min'- value of lambda that gives minimum mean cross-validated error (cvm). 'lambda.1se' - largest value of lambda such that error is within one standard error of the minimum.
- **family**: family of the model, i.e. cox, glm,...
- **verbose**: verbose
- **...**: Further parameters
Value

q.val minimal value of Q.func on the interval defined by bounds. Here, q.val is minimum mean cross-validated error (cvm)

model model list
  • alpha - optimal alpha
  • lambda - optimal lambda
  • nfolds - cross-validation's folds
  • cvreg - cv.glmnet object for optimal alpha
  • fit - glmnet object for optimal alpha and optimal lambda

Author(s)

Natalia Becker natalia.becker at d kfz.de

References


See Also

epsgo
Index

*Topic classification
aggregation.auc, 2

*Topic coefficient path
Plot.coef.glmnet, 13

*Topic graphs
epsgo, 7

*Topic iteration
balancedFolds, 4
epsgo, 7
tune.glmnet.interval, 25

*Topic models
aggregation.auc, 2
balancedFolds, 4
complexity.glmnet, 5
epsgo, 7
fit.glmnet, 11
PLL.coxnet, 12
Plot.peperr.curves, 14
predictProb.coxnet, 18
predictProb.glmnet, 19
tune.glmnet.interval, 25

*Topic multivariate
balancedFolds, 4
epsgo, 7
tune.glmnet.interval, 25

*Topic optimize
balancedFolds, 4
epsgo, 7
tune.glmnet.interval, 25

*Topic penalized regression
complexity.glmnet, 5
fit.glmnet, 11
PLL.coxnet, 12
predictProb.coxnet, 18
predictProb.glmnet, 19

*Topic plot
plot.sum.intsearch, 17

*Topic regression
aggregation.auc, 2
Plot.peperr.curves, 14

*Topic stability selection
plot.stabpath, 15
stabpath, 20
stabsel, 22

*Topic summary
summary.intsearch, 24

*Topic survival
PLL.coxnet, 12
Plot.peperr.curves, 14
predictProb.coxnet, 18
predictProb.glmnet, 19

*Topic system
coeff.sum.intsearch, 5

aggregation.auc, 2
balancedFolds, 4
balancedFolds (balancedFolds), 4

coef.sum.intsearch, 5
complexity.glmnet, 5
cv.glmnet, 6

EPSGO, 4, 5, 18, 25
epsgo, 7, 25, 27

fit.glmnet, 11
glmnet, 12, 19–21

peperr, 3, 6, 12, 15, 19, 20
PLL.coxnet, 12
Plot.coef.glmnet, 13
Plot.peperr.curves, 14
plot.stabpath, 15, 21, 24
plot.sum.intsearch, 5, 17, 25
predictProb.coxnet, 18, 20
predictProb.glmnet, 19, 19

stabpath, 17, 20, 24
INDEX

stabsel, 17, 21, 22
summary.intsearch, 5, 18, 24

tune.glmnet.interval, 25