Package: ncpen (via r-universe)

November 6, 2024

Type Package

Title Unified Algorithm for Non-convex Penalized Estimation for Generalized Linear Models

Version 1.0.0

Date 2018-11-16

Description An efficient unified nonconvex penalized estimation algorithm for Gaussian (linear), binomial Logit (logistic), Poisson, multinomial Logit, and Cox proportional hazard regression models. The unified algorithm is implemented based on the convex concave procedure and the algorithm can be applied to most of the existing nonconvex penalties. The algorithm also supports convex penalty: least absolute shrinkage and selection operator (LASSO). Supported nonconvex penalties include smoothly clipped absolute deviation (SCAD), minimax concave penalty (MCP), truncated LASSO penalty (TLP), clipped LASSO (CLASSO), sparse ridge (SRIDGE), modified bridge (MBRIDGE) and modified log (MLOG). For high-dimensional data (data set with many variables), the algorithm selects relevant variables producing a parsimonious regression model. Kim, D., Lee, S. and Kwon, S. (2018) <arXiv:1811.05061>, Lee, S., Kwon, S. and Kim, Y. (2016) <doi:10.1016/j.csda.2015.08.019>, Kwon, S., Lee, S. and Kim, Y. (2015) <doi:10.1016/j.csda.2015.07.001>. (This research is funded by Julian Virtue Professorship from Center for Applied Research at Pepperdine Graziadio Business School and the National Research Foundation of Korea.)

License GPL (>= 3)

URL `https://github.com/zeemkr/ncpen`

BugReports `https://github.com/zeemkr/ncpen/issues`

LazyData TRUE

Imports Rcpp (>= 0.11.2)

LinkingTo Rcpp, RcppArmadillo

Depends R(>= 3.4)

RoxygenNote 6.1.1

Language en-US

Repository https://zeemkr.r-universe.dev

RemoteUrl https://github.com/zeemkr/ncpen

RemoteRef HEAD

RemoteSha e17a0f5f2869d3993a3323e7a269dbb1819201ac

Contents

`ncpen-package` *ncpen: A package for non-convex penalized estimation for generalized linear models*

Description

This package fits the generalized linear models with various non-convex penalties. Supported regression models are Gaussian (linear), binomial Logit (logistic), multinomial Logit, Poisson and Cox proportional hazard. A unified algorithm is implemented in **ncpen** based on the convex concave procedure or difference convex algorithm that can be applied to most of existing non-convex penalties. The available penalties in the package are the least absolute shrinkage and selection operator(LASSO), smoothly clipped absolute deviation (SCAD), minimax concave penalty (MCP), truncated ℓ_1-penalty (TLP), clipped LASSO (CLASSO), sparse bridge (SRIDGE), modified bridge (MBRIDGE), and modified log (MLOG) penalties.

Details

The package accepts a design matrix X and vector of responses y, and produces the regularization path over a grid of values for the tuning parameter `lambda`. Also provides user-friendly processes for plotting, selecting tuning parameters using cross-validation or generalized information criterion (GIC), ℓ_2-regularization, penalty weights, standardization and intercept.

Note

This research is funded by Julian Virtue Professorship from Center for Applied Research at Pepperdine Graziadio Business School and the National Research Foundation of Korea.

Author(s)

Dongshin Kim, Sunghoon Kwon and Sangin Lee

References

Kim, D., Lee, S. and Kwon, S. (2018). A unified algorithm for the non-convex penalized estimation: The `ncpen` package. *http://arxiv.org/abs/1811.05061*.

Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, **92C**, 53-67.

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, **94**, 275-286.

Choi, H., Kim, Y. and Kwon, S. (2013). Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, **6**, 231-242.

`coef.cv.ncpen` *coef.cv.ncpen: extracts the optimal coefficients from* `cv.ncpen`*.*

Description

The function returns the optimal vector of coefficients.

Usage

```
## S3 method for class 'cv.ncpen'
coef(object, type = c("rmse", "like"), ...)
```

Arguments

`object`	(cv.ncpen object) fitted `cv.ncpen` object.
`type`	(character) a cross-validated error type which is either `rmse` or `like`.
`...`	other S3 parameters. Not used. Each error type is defined in `cv.ncpen`.

Value

the optimal coefficients vector selected by cross-validation.

`type`	error type.
`lambda`	the optimal lambda selected by CV.
`beta`	the optimal coefficients selected by CV.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`cv.ncpen`, `plot.cv.ncpen` , `gic.ncpen`

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10)
coef(fit)
### logistic regression with classo penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="binomial")
```

```
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,family="binomial",penalty="classo")
coef(fit)
### multinomial regression with sridge penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,family="multinomial",penalty="sridge")
coef(fit)
```

---

coef.ncpen   *coef.ncpen: extract the coefficients from an* ncpen *object*

Description

The function returns the coefficients matrix for all lambda values.

Usage

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

Arguments

object	(ncpen object) fitted ncpen object.
...	other S3 parameters. Not used.

Value

beta	The coefficients matrix or list for multinomial.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

ncpen

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat)
coef(fit)
### multinomial regression with classo penalty
sam = sam.gen.ncpen(n=200,p=20,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat,family="multinomial",penalty="classo")
coef(fit)
```

`control.ncpen` *control.ncpen: do preliminary works for* `ncpen`.

Description

The function returns controlled samples and tuning parameters for `ncpen` by eliminating unnecessary errors.

Usage

```
control.ncpen(y.vec, x.mat, family = c("gaussian", "binomial", "poisson",
  "multinomial", "cox"), penalty = c("scad", "mcp", "tlp", "lasso",
  "classo", "ridge", "sridge", "mbridge", "mlog"), x.standardize = TRUE,
  intercept = TRUE, lambda = NULL, n.lambda = NULL,
  r.lambda = NULL, w.lambda = NULL, gamma = NULL, tau = NULL,
  alpha = NULL, aiter.max = 100, b.eps = 1e-07)
```

Arguments

`y.vec`	(numeric vector) response vector. Must be 0,1 for `binomial` and 1,2,..., for `multinomial`.
`x.mat`	(numeric matrix) design matrix without intercept. The censoring indicator must be included at the last column of the design matrix for `cox`.
`family`	(character) regression model. Supported models are `gaussian`, `binomial`, `poisson`, `multinomial`, and `cox`. Default is `gaussian`.
`penalty`	(character) penalty function. Supported penalties are `scad` (smoothly clipped absolute deviation), `mcp` (minimax concave penalty), `tlp` (truncated LASSO penalty), `lasso` (least absolute shrinkage and selection operator), `classo` (clipped lasso = mcp + lasso), `ridge` (ridge), `sridge` (sparse ridge = mcp + ridge), `mbridge` (modified bridge) and `mlog` (modified log). Default is `scad`.
`x.standardize`	(logical) whether to standardize `x.mat` prior to fitting the model (see details). The estimated coefficients are always restored to the original scale.
`intercept`	(logical) whether to include an intercept in the model.

`lambda`	(numeric vector) user-specified sequence of `lambda` values. Default is supplied automatically from samples.
`n.lambda`	(numeric) the number of `lambda` values. Default is 100.
`r.lambda`	(numeric) ratio of the smallest `lambda` value to largest. Default is 0.001 when n>p, and 0.01 for other cases.
`w.lambda`	(numeric vector) penalty weights for each coefficient (see references). If a penalty weight is set to 0, the corresponding coefficient is always nonzero.
`gamma`	(numeric) additional tuning parameter for controlling shrinkage effect of `classo` and `sridge` (see references). Default is half of the smallest `lambda`.
`tau`	(numeric) concavity parameter of the penalties (see reference). Default is 3.7 for `scad`, 2.1 for `mcp`, `classo` and `sridge`, 0.001 for `tlp`, `mbridge` and `mlog`.
`alpha`	(numeric) ridge effect (weight between the penalty and ridge penalty) (see details). Default value is 1. If penalty is `ridge` and `sridge` then `alpha` is set to 0.
`aiter.max`	(numeric) maximum number of iterations in CD algorithm.
`b.eps`	(numeric) convergence threshold for coefficients vector.

Details

The function is used internal purpose but useful when users want to extract proper tuning parameters for `ncpen`. Do not supply the samples from `control.ncpen` into `ncpen` or `cv.ncpen` directly to avoid unexpected errors.

Value

An object with S3 class `ncpen`.

`y.vec`	response vector.
`x.mat`	design matrix adjusted to supplied options such as family and intercept.
`family`	regression model.
`penalty`	penalty.
`x.standardize`	whether to standardize `x.mat`.
`intercept`	whether to include the intercept.
`std`	scale factor for `x.standardize`.
`lambda`	lambda values for the analysis.
`n.lambda`	the number of `lambda` values.
`r.lambda`	ratio of the smallest `lambda` value to largest.
`w.lambda`	penalty weights for each coefficient.
`gamma`	additional tuning parameter for controlling shrinkage effect of `classo` and `sridge` (see references).
`tau`	concavity parameter of the penalties (see references).
`alpha`	ridge effect (amount of ridge penalty). see details.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96, 1348-60. Zhang, C.H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2), 894-942. Shen, X., Pan, W., Zhu, Y. and Zhou, H. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5), 807-832. Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, 92C, 53-67. Kwon, S. Kim, Y. and Choi, H.(2013). Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, 6, 231-242. Huang, J., Horowitz, J.L. and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2), 587-613. Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4), 1509. Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`ncpen`, `cv.ncpen`

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
tun = control.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,tau=1)
tun$tau
### multinomial regression with sridge penalty
sam = sam.gen.ncpen(n=200,p=10,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
tun = control.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,
                    family="multinomial",penalty="sridge",gamma=10)
### cox regression with mcp penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,r=0.2,cf.min=0.5,cf.max=1,corr=0.5,family="cox")
x.mat = sam$x.mat; y.vec = sam$y.vec
tun = control.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,family="cox",penalty="scad")
```

`cv.ncpen` *cv.ncpen: cross validation for* `ncpen`

Description

performs k-fold cross-validation (CV) for nonconvex penalized regression models over a sequence of the regularization parameter `lambda`.

Usage

```
cv.ncpen(y.vec, x.mat, family = c("gaussian", "linear", "binomial",
  "logit", "poisson", "multinomial", "cox"), penalty = c("scad", "mcp",
  "tlp", "lasso", "classo", "ridge", "sridge", "mbridge", "mlog"),
  x.standardize = TRUE, intercept = TRUE, lambda = NULL,
  n.lambda = NULL, r.lambda = NULL, w.lambda = NULL, gamma = NULL,
  tau = NULL, alpha = NULL, df.max = 50, cf.max = 100,
  proj.min = 10, add.max = 10, niter.max = 30, qiter.max = 10,
  aiter.max = 100, b.eps = 1e-06, k.eps = 1e-04, c.eps = 1e-06,
  cut = TRUE, local = FALSE, local.initial = NULL, n.fold = 10,
  fold.id = NULL)
```

Arguments

`y.vec`	(numeric vector) response vector. Must be 0,1 for `binomial` and 1,2,..., for `multinomial`.
`x.mat`	(numeric matrix) design matrix without intercept. The censoring indicator must be included at the last column of the design matrix for `cox`.
`family`	(character) regression model. Supported models are `gaussian` (or `linear`), `binomial` (or `logit`), `poisson`, `multinomial`, and `cox`. Default is `gaussian`.
`penalty`	(character) penalty function. Supported penalties are `scad` (smoothly clipped absolute deviation), `mcp` (minimax concave penalty), `tlp` (truncated LASSO penalty), `lasso` (least absolute shrinkage and selection operator), `classo` (clipped lasso = mcp + lasso), `ridge` (ridge), `sridge` (sparse ridge = mcp + ridge), `mbridge` (modified bridge) and `mlog` (modified log). Default is `scad`.
`x.standardize`	(logical) whether to standardize `x.mat` prior to fitting the model (see details). The estimated coefficients are always restored to the original scale.
`intercept`	(logical) whether to include an intercept in the model.
`lambda`	(numeric vector) user-specified sequence of `lambda` values. Default is supplied automatically from samples.
`n.lambda`	(numeric) the number of `lambda` values. Default is 100.
`r.lambda`	(numeric) ratio of the smallest `lambda` value to largest. Default is 0.001 when n>p, and 0.01 for other cases.
`w.lambda`	(numeric vector) penalty weights for each coefficient (see references). If a penalty weight is set to 0, the corresponding coefficient is always nonzero.
`gamma`	(numeric) additional tuning parameter for controlling shrinkage effect of `classo` and `sridge` (see references). Default is half of the smallest `lambda`.
`tau`	(numeric) concavity parameter of the penalties (see reference). Default is 3.7 for `scad`, 2.1 for `mcp`, `classo` and `sridge`, 0.001 for `tlp`, `mbridge` and `mlog`.
`alpha`	(numeric) ridge effect (weight between the penalty and ridge penalty) (see details). Default value is 1. If penalty is `ridge` and `sridge` then `alpha` is set to 0.
`df.max`	(numeric) the maximum number of nonzero coefficients.
`cf.max`	(numeric) the maximum of absolute value of nonzero coefficients.

`proj.min`	(numeric) the projection cycle inside CD algorithm (largely internal use. See details).
`add.max`	(numeric) the maximum number of variables added in CCCP iterations (largely internal use. See references).
`niter.max`	(numeric) maximum number of iterations in CCCP.
`qiter.max`	(numeric) maximum number of quadratic approximations in each CCCP iteration.
`aiter.max`	(numeric) maximum number of iterations in CD algorithm.
`b.eps`	(numeric) convergence threshold for coefficients vector.
`k.eps`	(numeric) convergence threshold for KKT conditions.
`c.eps`	(numeric) convergence threshold for KKT conditions (largely internal use).
`cut`	(logical) convergence threshold for KKT conditions (largely internal use).
`local`	(logical) whether to use local initial estimator for path construction. It may take a long time.
`local.initial`	(numeric vector) initial estimator for `local=TRUE`.
`n.fold`	(numeric) number of folds for CV.
`fold.id`	(numeric vector) fold ids from 1 to k that indicate fold configuration.

Details

Two kinds of CV errors are returned: root mean squared error and negative log likelihood. The results depends on the random partition made internally. To choose an optimal coefficients form the cv results, use `coef.cv.ncpen`. `ncpen` does not search values of `gamma`, `tau` and `alpha`.

Value

An object with S3 class `cv.ncpen`.

`ncpen.fit`	ncpen object fitted from the whole samples.
`fold.index`	fold ids of the samples.
`rmse`	rood mean squared errors from CV.
`like`	negative log-likelihoods from CV.
`lambda`	sequence of `lambda` used for CV.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96, 1348-60. Zhang, C.H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2), 894-942. Shen, X., Pan, W., Zhu, Y. and Zhou, H. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5), 807-832. Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, 92C, 53-67. Kwon, S. Kim, Y. and Choi, H.(2013). Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, 6, 231-242. Huang, J., Horowitz, J.L. and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2), 587-613. Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4), 1509. Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`plot.cv.ncpen`, `coef.cv.ncpen`, `ncpen`, `predict.ncpen`

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="gaussian")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,family="gaussian", penalty="scad")
coef(fit)
```

`cv.ncpen.reg` *cv.ncpen: cross validation for* `ncpen`

Description

performs k-fold cross-validation (CV) for nonconvex penalized regression models over a sequence of the regularization parameter `lambda`.

Usage

```
cv.ncpen.reg(formula, data, family = c("gaussian", "linear", "binomial",
  "logit", "multinomial", "cox", "poisson"), penalty = c("scad", "mcp",
  "tlp", "lasso", "classo", "ridge", "sridge", "mbridge", "mlog"),
  x.standardize = TRUE, intercept = TRUE, lambda = NULL,
  n.lambda = NULL, r.lambda = NULL, w.lambda = NULL, gamma = NULL,
  tau = NULL, alpha = NULL, df.max = 50, cf.max = 100,
  proj.min = 10, add.max = 10, niter.max = 30, qiter.max = 10,
  aiter.max = 100, b.eps = 1e-06, k.eps = 1e-04, c.eps = 1e-06,
  cut = TRUE, local = FALSE, local.initial = NULL, n.fold = 10,
  fold.id = NULL)
```

Arguments

`formula`	(formula) regression formula. To include/exclude intercept, use `intercept` option instead of using the "0 +" option in the formula. The y value must be 0,1 for `binomial` and 1,2,..., for `multinomial`.

data	(numeric matrix or data.frame) contains both y and X. Each row is an observation vector. The censoring indicator must be included at the last column of the data for cox.
family	(character) regression model. Supported models are gaussian (or linear), binomial (or logit), poisson, multinomial, and cox. Default is gaussian.
penalty	(character) penalty function. Supported penalties are scad (smoothly clipped absolute deviation), mcp (minimax concave penalty), tlp (truncated LASSO penalty), lasso (least absolute shrinkage and selection operator), classo (clipped lasso = mcp + lasso), ridge (ridge), sridge (sparse ridge = mcp + ridge), mbridge (modified bridge) and mlog (modified log). Default is scad.
x.standardize	(logical) whether to standardize x.mat prior to fitting the model (see details). The estimated coefficients are always restored to the original scale.
intercept	(logical) whether to include an intercept in the model.
lambda	(numeric vector) user-specified sequence of lambda values. Default is supplied automatically from samples.
n.lambda	(numeric) the number of lambda values. Default is 100.
r.lambda	(numeric) ratio of the smallest lambda value to largest. Default is 0.001 when n>p, and 0.01 for other cases.
w.lambda	(numeric vector) penalty weights for each coefficient (see references). If a penalty weight is set to 0, the corresponding coefficient is always nonzero.
gamma	(numeric) additional tuning parameter for controlling shrinkage effect of classo and sridge (see references). Default is half of the smallest lambda.
tau	(numeric) concavity parameter of the penalties (see reference). Default is 3.7 for scad, 2.1 for mcp, classo and sridge, 0.001 for tlp, mbridge and mlog.
alpha	(numeric) ridge effect (weight between the penalty and ridge penalty) (see details). Default value is 1. If penalty is ridge and sridge then alpha is set to 0.
df.max	(numeric) the maximum number of nonzero coefficients.
cf.max	(numeric) the maximum of absolute value of nonzero coefficients.
proj.min	(numeric) the projection cycle inside CD algorithm (largely internal use. See details).
add.max	(numeric) the maximum number of variables added in CCCP iterations (largely internal use. See references).
niter.max	(numeric) maximum number of iterations in CCCP.
qiter.max	(numeric) maximum number of quadratic approximations in each CCCP iteration.
aiter.max	(numeric) maximum number of iterations in CD algorithm.
b.eps	(numeric) convergence threshold for coefficients vector.
k.eps	(numeric) convergence threshold for KKT conditions.
c.eps	(numeric) convergence threshold for KKT conditions (largely internal use).
cut	(logical) convergence threshold for KKT conditions (largely internal use).

local	(logical) whether to use local initial estimator for path construction. It may take a long time.
local.initial	(numeric vector) initial estimator for local=TRUE.
n.fold	(numeric) number of folds for CV.
fold.id	(numeric vector) fold ids from 1 to k that indicate fold configuration.

Details

Two kinds of CV errors are returned: root mean squared error and negative log likelihood. The results depends on the random partition made internally. To choose an optimal coefficients form the cv results, use coef.cv.ncpen. ncpen does not search values of gamma, tau and alpha.

Value

An object with S3 class cv.ncpen.

ncpen.fit	ncpen object fitted from the whole samples.
fold.index	fold ids of the samples.
rmse	rood mean squared errors from CV.
like	negative log-likelihoods from CV.
lambda	sequence of lambda used for CV.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96, 1348-60. Zhang, C.H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2), 894-942. Shen, X., Pan, W., Zhu, Y. and Zhou, H. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5), 807-832. Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, 92C, 53-67. Kwon, S. Kim, Y. and Choi, H.(2013). Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, 6, 231-242. Huang, J., Horowitz, J.L. and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2), 587-613. Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4), 1509. Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

plot.cv.ncpen, coef.cv.ncpen, ncpen, predict.ncpen

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=5,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="gaussian")
x.mat = sam$x.mat; y.vec = sam$y.vec
data = cbind(y.vec, x.mat)
colnames(data) = c("y", paste("xv", 1:ncol(x.mat), sep = ""))
fit1 = cv.ncpen.reg(formula = y ~ xv1 + xv2 + xv3 + xv4 + xv5, data = data, n.lambda=10,
                    family="gaussian", penalty="scad")
fit2 = cv.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=10,family="gaussian", penalty="scad")
coef(fit1)
```

`excluded` *Check whether a pair should be excluded from interactions.*

Description

This is internal use only function.

Usage

```
excluded(excluded.pair, a, b)
```

Arguments

`excluded.pair`	a pair.
`a`	first column to be compared.
`b`	second column to be compared.

Value

TRUE if excluded, FALSE otherwise.

`fold.cv.ncpen` *fold.cv.ncpen: extracts fold ids for* `cv.ncpen`.

Description

The function returns fold configuration of the samples for CV.

Usage

```
fold.cv.ncpen(c.vec, n.fold = 10, family = c("gaussian", "binomial",
  "multinomial", "cox", "poisson"))
```

Arguments

c.vec	(numeric vector) vector for construction of CV ids: censoring indicator for cox and response vector for the others.
n.fold	(numeric) number of folds for CV.
family	(character) regression model. Supported models are gaussian, binomial, poisson, multinomial, and cox. Default is gaussian.

Value

fold ids of the samples.

idx	fold ids.
n.fold	the number of folds.
family	the model.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

cv.ncpen, plot.cv.ncpen , gic.ncpen

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
fold.id = fold.cv.ncpen(c.vec=y.vec,n.fold=10)
### logistic regression with classo penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="binomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fold.id = fold.cv.ncpen(c.vec=y.vec,n.fold=10,family="binomial")
### poison regression with mlog penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="poisson")
x.mat = sam$x.mat; y.vec = sam$y.vec
fold.id = fold.cv.ncpen(c.vec=y.vec,n.fold=10,family="poisson")
### multinomial regression with sridge penalty
sam = sam.gen.ncpen(n=200,p=20,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fold.id = fold.cv.ncpen(c.vec=y.vec,n.fold=10,family="multinomial")
### cox regression with mcp penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,r=0.2,cf.min=0.5,cf.max=1,corr=0.5,family="cox")
x.mat = sam$x.mat; y.vec = sam$y.vec
fold.id = fold.cv.ncpen(c.vec=x.mat[,21],n.fold=10,family="cox")
```

`gic.ncpen` *gic.ncpen: compute the generalized information criterion (GIC) for the selection of lambda*

Description

The function provides the selection of the regularization parameter lambda based on the GIC including AIC and BIC.

Usage

```
gic.ncpen(fit, weight = NULL, verbose = TRUE, ...)
```

Arguments

`fit`	(ncpen object) fitted `ncpen` object.
`weight`	(numeric) the weight factor for various information criteria. Default is BIC if `n>p` and GIC if `n<p` (see details).
`verbose`	(logical) whether to plot the GIC curve.
`...`	other graphical parameters to `plot`.

Details

User can supply various `weight` values (see references). For example, `weight=2`, `weight=log(n)`, `weight=log(log(p))log(n)`, `weight=log(log(n))log(p)`, corresponds to AIC, BIC (fixed dimensional model), modified BIC (diverging dimensional model) and GIC (high dimensional model).

Value

The coefficients `matrix`.

`gic`	the GIC values.
`lambda`	the sequence of lambda values used to calculate GIC.
`opt.beta`	the optimal coefficients selected by GIC.
`opt.lambda`	the optimal lambda value.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Wang, H., Li, R. and Tsai, C.L. (2007). Tuning parameter selectors for the smoothly clipped absolute deviation method. *Biometrika*, 94(3), 553-568. Wang, H., Li, B. and Leng, C. (2009). Shrinkage tuning parameter selection with a diverging number of parameters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3), 671-683. Kim, Y., Kwon, S. and Choi, H. (2012). Consistent Model Selection Criteria on High Dimensions. *Journal of Machine Learning Research*, 13, 1037-1057. Fan, Y. and Tang, C.Y. (2013). Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3), 531-552. Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`ncpen`

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat)
gic.ncpen(fit,pch="*",type="b")
### multinomial regression with classo penalty
sam = sam.gen.ncpen(n=200,p=20,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat,family="multinomial",penalty="classo")
gic.ncpen(fit,pch="*",type="b")
```

`interact.data` *Construct Interaction Matrix*

Description

`interact.data` interacts all the data in a `data.frame` or `matrix`.

Usage

```
interact.data(data, base.cols = NULL, exclude.pair = NULL)
```

Arguments

`data`	a `data.frame` or `matrix` to interact.
`base.cols`	indicates columns from one category. Interactions among variables from a same base.col will be avoided. For example, if three indicator columns, "ChannelR", "ChannelC" and "ChannelB", are created from a categorical column "Channel", then the interaction among them can be excluded by assigning `base.cols=c("Channel")`. Multiple `base.cols` are possible.

`exclude.pair`	the pairs will be excluded from interactions. This should be a `list` object of pairs. For example, `list(c("a1", "a2"), c("d1", "d2"))`.

Value

This returns an object of `matrix` which contains interactions.

Examples

```
df = data.frame(1:3, 4:6, 7:9, 10:12, 13:15);
colnames(df) = c("aa", "bb", "cc", "dd", "aa2");
df

interact.data(df);
interact.data(df, base.cols = "aa");
interact.data(df, base.cols = "aa", exclude.pair = list(c("bb", "cc")));
```

---

make.ncpen.data *Create ncpen Data Structure Using a Formula*

Description

This function creates ncpen y `vector` and x `matrix` from data using formula.

Usage

```
make.ncpen.data(formula, data)
```

Arguments

`formula`	(formula) regression formula. Intercept will not be created.
`data`	(numeric matrix or data.frame) contains both y and X.

Value

List of y vector and x matrix.

`y.vec`	y `vector`
`x.mat`	x `matrix`

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Examples

```
data = data.frame(y = 1:5, x1 = 6:10, x2 = 11:15);
formula = log(y) ~ log(x1) + x2;
make.ncpen.data(formula, data);
```

---

native_cpp_ncpen_fun_ *Native ncpen function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_ncpen_fun_(y_vec, x_mat0, w_vec0, lam_vec0, gam, tau, alp,
  d_max, iter_max, qiter_max, qiiter_max, b_eps, k_eps, p_eff, cut, c_eps,
  add, family, penalty, loc, ob_vec, div)
```

Arguments

y_vec	.
x_mat0	.
w_vec0	.
lam_vec0	.
gam	.
tau	.
alp	.
d_max	.
iter_max	.
qiter_max	.
qiiter_max	.
b_eps	.
k_eps	.
p_eff	.
cut	.
c_eps	.
add	.
family	.
penalty	.
loc	.
ob_vec	.
div	.

Value

.

`native_cpp_nr_fun_` *N/A.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_nr_fun_(fam, y_vec, x_mat, iter_max, b_eps)
```

Arguments

fam	.
y_vec	.
x_mat	.
iter_max	.
b_eps	.

Value

.

`native_cpp_obj_fun_` *Native object function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_obj_fun_(name, y_vec, x_mat, b_vec)
```

Arguments

name	.
y_vec	.
x_mat	.
b_vec	.

Value

.

native_cpp_obj_grad_fun_ *Native object gradient function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_obj_grad_fun_(name, y_vec, x_mat, b_vec)
```

Arguments

name	.
y_vec	.
x_mat	.
b_vec	.

Value

.

native_cpp_obj_hess_fun_ *Native object Hessian function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_obj_hess_fun_(name, y_vec, x_mat, b_vec)
```

Arguments

name	.
y_vec	.
x_mat	.
b_vec	.

Value

.

`native_cpp_pen_fun_` *Native Penalty function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_pen_fun_(name, b_vec, lam, gam, tau)
```

Arguments

`name`	.
`b_vec`	.
`lam`	.
`gam`	.
`tau`	.

Value

.

`native_cpp_pen_grad_fun_` *Native Penalty Gradient function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_pen_grad_fun_(name, b_vec, lam, gam, tau)
```

Arguments

`name`	.
`b_vec`	.
`lam`	.
`gam`	.
`tau`	.

Value

.

`native_cpp_p_ncpen_fun_`
Native point ncpen function.

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_p_ncpen_fun_(y_vec, x_mat, b_vec, w_vec, lam, gam, tau, alp,
  iter_max, qiter_max, qiiter_max, b_eps, k_eps, p_eff, cut, c_eps, family,
  penalty)
```

Arguments

`y_vec`	.
`x_mat`	.
`b_vec`	.
`w_vec`	.
`lam`	.
`gam`	.
`tau`	.
`alp`	.
`iter_max`	.
`qiter_max`	.
`qiiter_max`	.
`b_eps`	.
`k_eps`	.
`p_eff`	.
`cut`	.
`c_eps`	.
`family`	.
`penalty`	.

Value

.

native_cpp_qlasso_fun_   *Native QLASSO function.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_qlasso_fun_(q_mat, l_vec, b_vec0, w_vec, lam, iter_max,
  iiter_max, b_eps, k_eps, p_eff, q_rank, cut, c_eps)
```

Arguments

q_mat	.
l_vec	.
b_vec0	.
w_vec	.
lam	.
iter_max	.
iiter_max	.
b_eps	.
k_eps	.
p_eff	.
q_rank	.
cut	.
c_eps	.

Value

.

native_cpp_set_dev_mode_ *N/A.*

Description

This is internal use only function. Manual left blank on purpose.

Usage

```
native_cpp_set_dev_mode_(dev_mode)
```

Arguments

dev_mode	.

Value

.

ncpen *ncpen: nonconvex penalized estimation*

Description

Fits generalized linear models by penalized maximum likelihood estimation. The coefficients path is computed for the regression model over a grid of the regularization parameter lambda. Fits Gaussian (linear), binomial Logit (logistic), Poisson, multinomial Logit regression models, and Cox proportional hazard model with various non-convex penalties.

Usage

```
ncpen(y.vec, x.mat, family = c("gaussian", "linear", "binomial", "logit",
  "poisson", "multinomial", "cox"), penalty = c("scad", "mcp", "tlp",
  "lasso", "classo", "ridge", "sridge", "mbridge", "mlog"),
  x.standardize = TRUE, intercept = TRUE, lambda = NULL,
  n.lambda = NULL, r.lambda = NULL, w.lambda = NULL, gamma = NULL,
  tau = NULL, alpha = NULL, df.max = 50, cf.max = 100,
  proj.min = 10, add.max = 10, niter.max = 30, qiter.max = 10,
  aiter.max = 100, b.eps = 1e-07, k.eps = 1e-04, c.eps = 1e-06,
  cut = TRUE, local = FALSE, local.initial = NULL)
```

Arguments

`y.vec`	(numeric vector) response vector. Must be 0,1 for `binomial` and 1,2,..., for `multinomial`.
`x.mat`	(numeric matrix) design matrix without intercept. The censoring indicator must be included at the last column of the design matrix for `cox`.
`family`	(character) regression model. Supported models are `gaussian` (or `linear`), `binomial` (or `logit`), `poisson`, `multinomial`, and `cox`, Default is `gaussian`.
`penalty`	(character) penalty function. Supported penalties are `scad` (smoothly clipped absolute deviation), `mcp` (minimax concave penalty), `tlp` (truncated lasso penalty), `lasso` (least absolute shrinkage and selection operator), `classo` (clipped lasso = mcp + lasso), `ridge` (ridge), `sridge` (sparse ridge = mcp + ridge), `mbridge` (modified bridge) and `mlog` (modified log). Default is `scad`.
`x.standardize`	(logical) whether to standardize `x.mat` prior to fitting the model (see details). The estimated coefficients are always restored to the original scale.
`intercept`	(logical) whether to include an intercept in the model.
`lambda`	(numeric vector) user-specified sequence of `lambda` values. Default is supplied automatically from samples.
`n.lambda`	(numeric) the number of `lambda` values. Default is 100.
`r.lambda`	(numeric) ratio of the smallest `lambda` value to largest. Default is 0.001 when n>p, and 0.01 for other cases.
`w.lambda`	(numeric vector) penalty weights for each coefficient (see references). If a penalty weight is set to 0, the corresponding coefficient is always nonzero.
`gamma`	(numeric) additional tuning parameter for controlling shrinkage effect of `classo` and `sridge` (see references). Default is half of the smallest `lambda`.
`tau`	(numeric) concavity parameter of the penalties (see reference). Default is 3.7 for `scad`, 2.1 for `mcp`, `classo` and `sridge`, 0.001 for `tlp`, `mbridge` and `mlog`.
`alpha`	(numeric) ridge effect (weight between the penalty and ridge penalty) (see details). Default value is 1. If penalty is `ridge` and `sridge` then `alpha` is set to 0.
`df.max`	(numeric) the maximum number of nonzero coefficients.
`cf.max`	(numeric) the maximum of absolute value of nonzero coefficients.
`proj.min`	(numeric) the projection cycle inside CD algorithm (largely internal use. See details).
`add.max`	(numeric) the maximum number of variables added in CCCP iterations (largely internal use. See references).
`niter.max`	(numeric) maximum number of iterations in CCCP.
`qiter.max`	(numeric) maximum number of quadratic approximations in each CCCP iteration.
`aiter.max`	(numeric) maximum number of iterations in CD algorithm.
`b.eps`	(numeric) convergence threshold for coefficients vector in CD algorithm
`k.eps`	(numeric) convergence threshold for KKT conditions.

c.eps	(numeric) convergence threshold for KKT conditions (largely internal use).
cut	(logical) convergence threshold for KKT conditions (largely internal use).
local	(logical) whether to use local initial estimator for path construction. It may take a long time.
local.initial	(numeric vector) initial estimator for local=TRUE.

Details

The sequence of models indexed by lambda is fit by using concave convex procedure (CCCP) and coordinate descent (CD) algorithm (see references). The objective function is

$$(sum of squared residuals)/2n + [alpha * penalty + (1 - alpha) * ridge]$$

for gaussian and

$$(log - likelihood)/n - [alpha * penalty + (1 - alpha) * ridge]$$

for the others, assuming the canonical link. The algorithm applies the warm start strategy (see references) and tries projections after proj.min iterations in CD algorithm, which makes the algorithm fast and stable. x.standardize makes each column of x.mat to have the same Euclidean length but the coefficients will be re-scaled into the original. In multinomial case, the coefficients are expressed in vector form. Use coef.ncpen.

Value

An object with S3 class ncpen.

y.vec	response vector.
x.mat	design matrix.
family	regression model.
penalty	penalty.
x.standardize	whether to standardize x.mat=TRUE.
intercept	whether to include the intercept.
std	scale factor for x.standardize.
lambda	sequence of lambda values.
w.lambda	penalty weights.
gamma	extra shrinkage parameter for classo and sridge only.
alpha	ridge effect.
local	whether to use local initial estimator.
local.initial	local initial estimator for local=TRUE.
beta	fitted coefficients. Use coef.ncpen for multinomial since the coefficients are represented as vectors.
df	the number of non-zero coefficients.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Kim, D., Lee, S. and Kwon, S. (2018). A unified algorithm for the non-convex penalized estimation: The ncpen package. *http://arxiv.org/abs/1811.05061*.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96, 1348-60.

Zhang, C.H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2), 894-942.

Shen, X., Pan, W., Zhu, Y. and Zhou, H. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5), 807-832.

Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, 92C, 53-67.

Kwon, S. Kim, Y. and Choi, H.(2013). Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, 6, 231-242.

Huang, J., Horowitz, J.L. and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2), 587-613.

Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4), 1509.

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`coef.ncpen`, `plot.ncpen`, `gic.ncpen`, `predict.ncpen`, `cv.ncpen`

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=10,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="gaussian")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat,family="gaussian", penalty="scad")
```

`ncpen.reg` *ncpen.reg: nonconvex penalized estimation*

Description

Fits generalized linear models by penalized maximum likelihood estimation. The coefficients path is computed for the regression model over a grid of the regularization parameter `lambda`. Fits Gaussian (linear), binomial Logit (logistic), Poisson, multinomial Logit regression models, and Cox proportional hazard model with various non-convex penalties.

Usage

```
ncpen.reg(formula, data, family = c("gaussian", "linear", "binomial",
  "logit", "multinomial", "cox", "poisson"), penalty = c("scad", "mcp",
  "tlp", "lasso", "classo", "ridge", "sridge", "mbridge", "mlog"),
  x.standardize = TRUE, intercept = TRUE, lambda = NULL,
  n.lambda = NULL, r.lambda = NULL, w.lambda = NULL, gamma = NULL,
  tau = NULL, alpha = NULL, df.max = 50, cf.max = 100,
  proj.min = 10, add.max = 10, niter.max = 30, qiter.max = 10,
  aiter.max = 100, b.eps = 1e-07, k.eps = 1e-04, c.eps = 1e-06,
  cut = TRUE, local = FALSE, local.initial = NULL)
```

Arguments

`formula`	(formula) regression formula. To include/exclude intercept, use `intercept` option instead of using the "0 +" option in the formula. The y value must be 0,1 for `binomial` and 1,2,..., for `multinomial`.
`data`	(numeric matrix or data.frame) contains both y and X. Each row is an observation vector. The censoring indicator must be included at the last column of the data for `cox`.
`family`	(character) regression model. Supported models are `gaussian` (or `linear`), `binomial` (or `logit`), `poisson`, `multinomial`, and `cox`. Default is `gaussian`.
`penalty`	(character) penalty function. Supported penalties are `scad` (smoothly clipped absolute deviation), `mcp` (minimax concave penalty), `tlp` (truncated LASSO penalty), `lasso` (least absolute shrinkage and selection operator), `classo` (clipped lasso = mcp + lasso), `ridge` (ridge), `sridge` (sparse ridge = mcp + ridge), `mbridge` (modified bridge) and `mlog` (modified log). Default is `scad`.
`x.standardize`	(logical) whether to standardize `x.mat` prior to fitting the model (see details). The estimated coefficients are always restored to the original scale.
`intercept`	(logical) whether to include an intercept in the model.
`lambda`	(numeric vector) user-specified sequence of `lambda` values. Default is supplied automatically from samples.
`n.lambda`	(numeric) the number of `lambda` values. Default is 100.
`r.lambda`	(numeric) ratio of the smallest `lambda` value to largest. Default is 0.001 when n>p, and 0.01 for other cases.
`w.lambda`	(numeric vector) penalty weights for each coefficient (see references). If a penalty weight is set to 0, the corresponding coefficient is always nonzero.
`gamma`	(numeric) additional tuning parameter for controlling shrinkage effect of `classo` and `sridge` (see references). Default is half of the smallest `lambda`.
`tau`	(numeric) concavity parameter of the penalties (see reference). Default is 3.7 for `scad`, 2.1 for `mcp`, `classo` and `sridge`, 0.001 for `tlp`, `mbridge` and `mlog`.
`alpha`	(numeric) ridge effect (weight between the penalty and ridge penalty) (see details). Default value is 1. If penalty is `ridge` and `sridge` then `alpha` is set to 0.
`df.max`	(numeric) the maximum number of nonzero coefficients.

cf.max	(numeric) the maximum of absolute value of nonzero coefficients.
proj.min	(numeric) the projection cycle inside CD algorithm (largely internal use. See details).
add.max	(numeric) the maximum number of variables added in CCCP iterations (largely internal use. See references).
niter.max	(numeric) maximum number of iterations in CCCP.
qiter.max	(numeric) maximum number of quadratic approximations in each CCCP iteration.
aiter.max	(numeric) maximum number of iterations in CD algorithm.
b.eps	(numeric) convergence threshold for coefficients vector.
k.eps	(numeric) convergence threshold for KKT conditions.
c.eps	(numeric) convergence threshold for KKT conditions (largely internal use).
cut	(logical) convergence threshold for KKT conditions (largely internal use).
local	(logical) whether to use local initial estimator for path construction. It may take a long time.
local.initial	(numeric vector) initial estimator for local=TRUE.

Details

The sequence of models indexed by lambda is fit by using concave convex procedure (CCCP) and coordinate descent (CD) algorithm (see references). The objective function is

$$(sum of squared residuals)/2n + [alpha * penalty + (1 - alpha) * ridge]$$

for gaussian and

$$(log - likelihood)/n - [alpha * penalty + (1 - alpha) * ridge]$$

for the others, assuming the canonical link. The algorithm applies the warm start strategy (see references) and tries projections after proj.min iterations in CD algorithm, which makes the algorithm fast and stable. x.standardize makes each column of x.mat to have the same Euclidean length but the coefficients will be re-scaled into the original. In multinomial case, the coefficients are expressed in vector form. Use coef.ncpen.

Value

An object with S3 class ncpen.

y.vec	response vector.
x.mat	design matrix.
family	regression model.
penalty	penalty.
x.standardize	whether to standardize x.mat=TRUE.
intercept	whether to include the intercept.
std	scale factor for x.standardize.

`lambda`	sequence of `lambda` values.
`w.lambda`	penalty weights.
`gamma`	extra shrinkage parameter for `classo` and sridge only.
`alpha`	ridge effect.
`local`	whether to use local initial estimator.
`local.initial`	local initial estimator for `local=TRUE`.
`beta`	fitted coefficients. Use `coef.ncpen` for `multinomial` since the coefficients are represented as vectors.
`df`	the number of non-zero coefficients.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96, 1348-60. Zhang, C.H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2), 894-942. Shen, X., Pan, W., Zhu, Y. and Zhou, H. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5), 807-832. Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, 92C, 53-67. Kwon, S. Kim, Y. and Choi, H.(2013). Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, 6, 231-242. Huang, J., Horowitz, J.L. and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2), 587-613. Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4), 1509. Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`coef.ncpen`, `plot.ncpen`, `gic.ncpen`, `predict.ncpen`, `cv.ncpen`

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=5,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="gaussian")
x.mat = sam$x.mat; y.vec = sam$y.vec
data = cbind(y.vec, x.mat)
colnames(data) = c("y", paste("xv", 1:ncol(x.mat), sep = ""))
fit1 = ncpen.reg(formula = y ~ xv1 + xv2 + xv3 + xv4 + xv5, data = data,
                 family="gaussian", penalty="scad")
fit2 = ncpen(y.vec=y.vec,x.mat=x.mat);
```

plot.cv.ncpen *plot.cv.ncpen: plot cross-validation error curve.*

Description

The function Produces a plot of the cross-validated errors from `cv.ncpen` object.

Usage

```
## S3 method for class 'cv.ncpen'
plot(x, type = c("rmse", "like"), log.scale = FALSE,
  ...)
```

Arguments

`x`	fitted `cv.ncpen` object.
`type`	(character) a cross-validated error type which is either `rmse` or `like`.
`log.scale`	(logical) whether to use log scale of lambda for horizontal axis.
`...`	other graphical parameters to `plot`

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

`cv.ncpen`

Examples

```
### linear regression with scad penalty
par(mfrow=c(1,2))
sam =  sam.gen.ncpen(n=500,p=10,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="gaussian")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,n.lambda=50,family="gaussian", penalty="scad")
plot(fit)
plot(fit,log.scale=F)
```

plot.ncpen — *plot.ncpen: plots coefficients from an* ncpen *object.*

Description

Produces a plot of the coefficients paths for a fitted ncpen object. Class-wise paths can be drawn for multinomial.

Usage

```
## S3 method for class 'ncpen'
plot(x, log.scale = FALSE, mult.type = c("mat", "vec"),
  ...)
```

Arguments

x	(ncpen object) Fitted ncpen object.
log.scale	(logical) whether to use log scale of lambda for horizontal axis.
mult.type	(character) additional option for multinomial whether to draw the coefficients class-wise or not. Default is mat that uses class-wise coefficients.
...	other graphical parameters to plot

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

ncpen

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat)
plot(fit)
### multinomial regression with classo penalty
sam = sam.gen.ncpen(n=200,p=20,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat,family="multinomial",penalty="classo")
plot(fit)
plot(fit,mult.type="vec",log.scale=TRUE)
```

power.data *Power Data*

Description

power.data power data and return a data.frame with column names with tail.

Usage

```
power.data(data, power, tail = "_pow")
```

Arguments

data	a data.frame or matrix object.
power	power.
tail	tail text for column names for powered data. For example, if a column "sales" is powered by 4 (=power) and tail is "_pow", then the output column name becomes "sales_pow4".

Value

This returns an object of matrix.

Examples

```
df = data.frame(a = 1:3, b= 4:6);
power.data(df, 2, ".pow");
```

predict.ncpen *predict.ncpen: make predictions from an ncpen object*

Description

The function provides various types of predictions from a fitted ncpen object: response, regression, probability, root mean squared error (RMSE), negative log-likelihood (LIKE).

Usage

```
## S3 method for class 'ncpen'
predict(object, type = c("y", "reg", "prob", "rmse",
  "like"), new.y.vec = NULL, new.x.mat = NULL, prob.cut = 0.5, ...)
```

Arguments

object	(ncpen object) fitted ncpen object.
type	(character) type of prediction. y returns new responses from new.x.mat. reg returns new linear predictors from new.x.mat. prob returns new class probabilities from new.x.mat for binomial and multinomial. rmse returns RMSE from new.y.vec and new.x.mat. prob returns LIKE from new.y.vec and new.x.mat.
new.y.vec	(numeric vector). vector of new response at which predictions are to be made.
new.x.mat	(numeric matrix). matrix of new design at which predictions are to be made.
prob.cut	(numeric) threshold value of probability for binomial.
...	other S3 parameters. Not used.

Value

prediction values depending on type for all lambda values.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics and Data Analysis*, 94, 275-286.

See Also

ncpen

Examples

```
### linear regression with scad penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec[1:190],x.mat=x.mat[1:190,])
predict(fit,"y",new.x.mat=x.mat[190:200,])
### logistic regression with classo penalty
sam =  sam.gen.ncpen(n=200,p=20,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="binomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec[1:190],x.mat=x.mat[1:190,],family="binomial",penalty="classo")
predict(fit,"y",new.x.mat=x.mat[190:200,])
predict(fit,"y",new.x.mat=x.mat[190:200,],prob.cut=0.3)
predict(fit,"reg",new.x.mat=x.mat[190:200,])
predict(fit,"prob",new.x.mat=x.mat[190:200,])
### multinomial regression with sridge penalty
sam = sam.gen.ncpen(n=200,p=20,q=5,k=3,cf.min=0.5,cf.max=1,corr=0.5,family="multinomial")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec[1:190],x.mat=x.mat[1:190,],family="multinomial",penalty="classo")
predict(fit,"y",new.x.mat=x.mat[190:200,])
```

```
predict(fit,"reg",new.x.mat=x.mat[190:200,])
predict(fit,"prob",new.x.mat=x.mat[190:200,])
```

---

sam.gen.ncpen *sam.gen.ncpen: generate a simulated dataset.*

Description

Generate a synthetic dataset based on the correlation structure from generalized linear models.

Usage

```
sam.gen.ncpen(n = 100, p = 50, q = 10, k = 3, r = 0.3,
  cf.min = 0.5, cf.max = 1, corr = 0.5, seed = NULL,
  family = c("gaussian", "binomial", "multinomial", "cox", "poisson"))
```

Arguments

`n`	(numeric) the number of samples.
`p`	(numeric) the number of variables.
`q`	(numeric) the number of nonzero coefficients.
`k`	(numeric) the number of classes for `multinomial`.
`r`	(numeric) the ratio of censoring for `cox`.
`cf.min`	(numeric) value of the minimum coefficient.
`cf.max`	(numeric) value of the maximum coefficient.
`corr`	(numeric) strength of correlations in the correlation structure.
`seed`	(numeric) seed number for random generation. Default does not use seed.
`family`	(character) model type.

Details

A design matrix for regression models is generated from the multivariate normal distribution with a correlation structure. Then the response variables are computed with a specific model based on the true coefficients (see references). Note the censoring indicator locates at the last column of `x.mat` for `cox`.

Value

An object with list class containing

`x.mat`	design matrix.
`y.vec`	responses.
`b.vec`	true coefficients.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

References

Kwon, S., Lee, S. and Kim, Y. (2016). Moderately clipped LASSO. *Computational Statistics and Data Analysis*, 92C, 53-67. Kwon, S. and Kim, Y. (2012). Large sample properties of the SCAD-penalized maximum likelihood estimation on high dimensions. *Statistica Sinica*, 629-653.

See Also

ncpen

Examples

```
### linear regression
sam =  sam.gen.ncpen(n=200,p=20,q=5,cf.min=0.5,cf.max=1,corr=0.5)
x.mat = sam$x.mat; y.vec = sam$y.vec
head(x.mat); head(y.vec)
```

---

same.base *Check whether column names are derivation of a same base.*

Description

This is internal use only function.

Usage

```
same.base(base.cols, a, b)
```

Arguments

base.cols	vector of base column names.
a	first column to be compared.
b	second column to be compared.

Value

TRUE if same base, FALSE otherwise.

to.indicators — *Construct Indicator Matrix*

Description

to.indicators converts a categorical variable into a data.frame with indicator (0 or 1) variables for each category.

Usage

```
to.indicators(vec, exclude.base = TRUE, base = NULL, prefix = NULL)
```

Arguments

vec	a categorical vector.
exclude.base	FALSE means to include all the categories. TRUE means to exclude one category as a base case. If base is not specified, a random category will be removed.
base	a base category removed from the indicator matrix. This option works only when the type variable is set to "exclude.base".
prefix	a prefix to be used for column names of the output matrix. Default is "cat_" if prefix is NULL. For example, if a category vector has values of c("aa", "bb", "cc"), column names of the output matrix will be "cat_aa", "cat_bb" and "cat_cc". If vec is a data.frame and prefix is NULL, then the vec's column name followed by "_" will be used as a prefix.

Value

This returns an object of matrix which contains indicators.

Examples

```
a1 = 4:10;
b1 = c("aa", "bb", "cc");

to.indicators(a1, base = 10);
to.indicators(b1, base = "bb", prefix = "T_");
to.indicators(as.data.frame(b1), base = "bb");
```

to.ncpen.x.mat *Convert a* `data.frame` *to a* ncpen *usable* `matrix`.

Description

This automates the processes of `to.indicators` and `interact.data`. First, it converts categorical variables to a series of indicators. All other numerical and logical variables are preserved. Then, if `interact.all == TRUE`, all the variables are interacted.

Usage

```
to.ncpen.x.mat(df, base = NULL, interact.all = FALSE,
  base.cols = NULL, exclude.pair = NULL)
```

Arguments

df	a `data.frame` which includes numerical, logical and categorical columns.
base	a base category removed from the indicator variables. This base will work as the base case for all the categorical variables.
interact.all	indicates whether to interact all the columns (`TRUE`) or not (`FALSE`).
base.cols	indicates columns derived from a same column. For example, if `age_sq` is `age^2`, then `"age"` is a base column. Categorical columns will be automatically considered as base columns.
exclude.pair	the pairs will be excluded from interactions. This should be a `list` object of pairs. For example, `list(c("a1", "a2"), c("d1", "d2"))`.

Value

This returns an object of `matrix`.

Examples

```
df = data.frame(num = c(1, 2, 3, 4, 5),
                ctr = c("K", "O", "R", "R", "K"),
                logi = c(TRUE, TRUE, FALSE, FALSE, TRUE),
                age = c(10, 20, 30, 40, 50),
                age_sq = c(10, 20, 30, 40, 50)^2,
                loc = c("b", "a", "c", "a", "b"),
                FTHB = c(1,0,1,0,1),
                PRM  = c(0,1,0,1,0),
                PMI  = c(1,1,0,0,0));

to.ncpen.x.mat(df, interact.all = TRUE,
   base.cols = c("age"),
   exclude.pair = list(c("FTHB", "PRM")));
```

Index