Package: mpath (via r-universe)

October 18, 2026

Title Regularized Linear Models

Version 0.4-2.28

Date 2026-10-15

Maintainer Zhu Wang <zwang145@uthsc.edu>

Description Algorithms compute robust estimators for loss functions in the concave convex (CC) family by the iteratively reweighted convex optimization (IRCO), an extension of the iteratively reweighted least squares (IRLS). The IRCO reduces the weight of the observation that leads to a large loss; it also provides weights to help identify outliers. Applications include robust (penalized) generalized linear models and robust support vector machines. The package also contains penalized Poisson, negative binomial, zero-inflated Poisson, zero-inflated negative binomial regression models and robust models with non-convex loss functions. Wang et al. (2014) <doi:10.1002/sim.6314>, Wang et al. (2015) <doi:10.1002/bimj.201400143>, Wang et al. (2016) <doi:10.1177/0962280214530608>, Wang (2021) <doi:10.1007/s11749-021-00770-2>, Wang (2024) <doi:10.1111/anzs.12409>.

Depends R (>= 3.5.0), methods, glmnet

Imports MASS, pscl, numDeriv, foreach, doParallel, parallel, bst, WeightSVM

Suggests R.rsp

VignetteBuilder R.rsp

License GPL-2

URL https://github.com/zhuwang46/mpath

BugReports https://github.com/zhuwang46/mpath/issues

NeedsCompilation yes

RoxygenNote 7.1.1

Author Zhu Wang [aut, cre], Achim Zeileis [aut], Simon Jackman [aut], Brian Ripley [aut], Patrick Breheny [aut]

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

Date/Publication 2026-10-18 17:10:02 UTC

RemoteUrl https://github.com/cran/mpath

RemoteRef HEAD

RemoteSha c34c01dbd0decaaa1c341ef993868b41f064ef34

Contents

be.zeroinfl . . . 3
breadReg . . . 4
breastfeed . . . 5
compute_g . . . 5
compute_wt . . . 6
conv2glmreg . . . 7
conv2zipath . . . 8
cv.glmreg . . . 8
cv.glmreg_fit . . . 11
cv.glmregNB . . . 13
cv.irglmreg . . . 15
cv.irglmreg_fit . . . 16
cv.irsvm . . . 18
cv.irsvm_fit . . . 20
cv.nclreg . . . 22
cv.nclreg_fit . . . 24
cv.zipath . . . 25
cv.zipath_fit . . . 28
docvisits . . . 30
estfunReg . . . 31
gfunc . . . 32
glmreg . . . 32
glmreg_fit . . . 35
glmregNB . . . 38
hessianReg . . . 41
irglm . . . 42
irglmreg . . . 44
irglmreg_fit . . . 46
irsvm . . . 50
irsvm_fit . . . 51
loss2 . . . 54
loss2_irsvm . . . 55
loss3 . . . 56
meatReg . . . 57
methods . . . 58
ncl . . . 59
ncl_fit . . . 60
nclreg . . . 62

nclreg_fit 64
plot.glmreg 67
predict.glmreg 68
predict.zipath 69
pval.zipath 71
rzi 72
sandwichReg 73
se 74
stan 75
summary.glmregNB 76
tuning.zipath 77
update_wt 79
zipath 80
zipath_fit 82

Index **87**

`be.zeroinfl` *conduct backward stepwise variable elimination for zero inflated count regression*

Description

conduct backward stepwise variable elimination for zero inflated count regression from zeroinfl function

Usage

```
be.zeroinfl(object, data, dist=c("poisson", "negbin", "geometric"), alpha=0.05,
            trace=FALSE)
```

Arguments

`object`	an object from function zeroinfl
`data`	argument controlling formula processing via `model.frame`.
`dist`	one of the distributions in zeroinfl function
`alpha`	significance level of variable elimination
`trace`	logical value, if TRUE, print detailed calculation results

Details

conduct backward stepwise variable elimination for zero inflated count regression from zeroinfl function

Value

an object of zeroinfl with all variables having p-values less than the significance level `alpha`

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

`breadReg` *Bread for Sandwiches in Regularized Estimators*

Description

Generic function for extracting an estimator for the bread of sandwiches.

Usage

```
breadReg(x, which, ...)
```

Arguments

`x`	a fitted model object.
`which`	which penalty parameter(s)?
`...`	arguments passed to methods.

Value

A matrix containing an estimator for the penalized second derivative of log-likelihood function. Typically, this should be an $k \times k$ matrix corresponding to k parameters. The rows and columns should be named as in `coef` or `terms`, respectively.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

meatReg, sandwichReg

Examples

```
data("bioChemists", package = "pscl")
fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10, maxit.em=1)
breadReg(fm_zinb, which=which.min(fm_zinb$bic))
```

---

breastfeed *Breast feeding decision*

Description

In a UK hospital, 135 expectant mothers were surveyed on the decision of breastfeeding their babies or not, along with two-level predictive factors

Usage

```
data(breastfeed)
```

Source

Stephane Heritier, Eva Cantoni, Samuel Copt and Maria-Pia Victoria-Fese (2009). *Robust Methods in Biostatistics*, John Wiley & Sons

Examples

```
data(breastfeed)
str(breastfeed)
```

---

compute_g *Compute concave function values*

Description

Compute concave function values

Usage

```
compute_g(z, cfun, s, delta=0.0001)
```

Arguments

z	vector nonnegative values from dfun, e.g., u^2/2
cfun	integer from 1-8, concave function as in irglm_fit
s	a numeric value, see details in irglmreg_fit
delta	a positive small value, see details in irglmreg_fit

Value

Concave function values

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irglmreg

Examples

```
u <- seq(0, 4, by=0.01)
z <- u^2/2 ### this is dfun
res <- compute_g(z, cfun=1, s=1)
plot(z, res, ylab="Weight", type="l", lwd=2,
     main=expression(paste("hcave", "   (", sigma, "=1)", )))
```

---

compute_wt *Weight value from concave function*

Description

Weight value from concave function

Usage

```
compute_wt(z, weights, cfun, s, delta=0.0001)
```

Arguments

z	vector nonnegative values from dfun, e.g., u^2/2
weights	optional numeric vector of weights.
cfun	integer from 1-8, concave function as in irglm_fit
s	a numeric value, see details in irglm_fit
delta	a positive small value, see details in irglm_fit

Value

Weight value from concave function

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irglmreg

Examples

```
u <- seq(0, 4, by=0.01)
z <- u^2/2 ### this is dfun
res <- compute_wt(z, cfun=1, s=1)
plot(z, res, ylab="Weight", type="l", lwd=2,
     main=expression(paste("hcave", "   (", sigma, "=1)", )))
```

---

conv2glmreg *convert glm object to class glmreg*

Description

convert glm object to class glmreg, which then can be used for other purposes

Usage

```
conv2glmreg(object, family=c("poisson", "negbin"))
```

Arguments

object	an object of class glm
family	one of families in glm class

Value

an object of class glmreg

Author(s)

Zhu Wang <zwang145@uthsc.edu>

`conv2zipath` *convert zeroinfl object to class zipath*

Description

convert zeroinfl object to class zipath, which then can be used to predict new data

Usage

```
conv2zipath(object, family=c("poisson", "negbin", "geometric"))
```

Arguments

`object`	an object of class zeroinfl
`family`	one of families in zeroinfl class

Value

an object of class zipath

Author(s)

Zhu Wang <zwang145@uthsc.edu>

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

Description

Does k-fold cross-validation for glmreg, produces a plot, and returns cross-validated log-likelihood values for `lambda`

Usage

```
## S3 method for class 'formula'
cv.glmreg(formula, data, weights, offset=NULL, contrasts=NULL, ...)
## S3 method for class 'matrix'
cv.glmreg(x, y, weights, offset=NULL, ...)
## Default S3 method:
cv.glmreg(x,  ...)
## S3 method for class 'cv.glmreg'
plot(x,se=TRUE,ylab=NULL, main=NULL, width=0.02, col="darkgrey", ...)
## S3 method for class 'cv.glmreg'
predict(object, newx, ...)
## S3 method for class 'cv.glmreg'
coef(object,which=object$lambda.which, ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
x	x matrix as in glmreg. It could be object of cv.glmreg.
y	response y as in glmreg.
weights	Observation weights; defaults to 1 per observation
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
contrasts	the contrasts corresponding to levels from the respective models
object	object of cv.glmreg
newx	Matrix of values at which predictions are to be made. Not used for type="coefficients"
which	Indices of the penalty parameter lambda at which estimates are extracted. By default, the one which generates the optimal cross-validation value.
se	logical value, if TRUE, standard error curve is also plotted
ylab	ylab on y-axis
main	title of plot
width	width of lines
col	color of standard error curve
...	Other arguments that can be passed to glmreg.

Details

The function runs glmreg nfolds+1 times; the first to compute the lambda sequence, and then to compute the fit with each of the folds omitted. The error or the log-likelihood value is accumulated, and the average value and standard deviation over the folds is computed. Note that cv.glmreg can be used to search for values for alpha: it is required to call cv.glmreg with a fixed vector foldid for different values of alpha.

Value

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

fit	a fitted glmreg object for the full data.
residmat	matrix of log-likelihood values with row values for lambda and column values for kth cross-validation
bic	matrix of BIC values with row values for lambda and column values for kth cross-validation
cv	The mean cross-validated log-likelihood values - a vector of length length(lambda).
cv.error	estimate of standard error of cv.
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in.
lambda	a vector of lambda values
lambda.which	index of lambda that gives maximum cv value.
lambda.optim	value of lambda that gives maximum cv value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

glmreg and plot, predict, and coef methods for "cv.glmreg" object.

Examples

```
data("bioChemists", package = "pscl")
fm_pois <- cv.glmreg(art ~ ., data = bioChemists, family = "poisson")
title("Poisson Family",line=2.5)
predict(fm_pois, newx=bioChemists[,-1])[1:4]
coef(fm_pois)
```

cv.glmreg_fit *Internal function of cross-validation for glmreg*

Description

Internal function to conduct k-fold cross-validation for glmreg, produces a plot, and returns cross-validated log-likelihood values for lambda

Usage

```
cv.glmreg_fit(x, y, weights, offset, lambda=NULL, balance=TRUE,
              family=c("gaussian", "binomial", "poisson", "negbin"),
              type=c("loss", "error"), nfolds=10, foldid, plot.it=TRUE,
              se=TRUE, n.cores=NULL, trace=FALSE, parallel=FALSE, ...)
```

Arguments

x	x matrix as in glmreg.
y	response y as in glmreg.
weights	Observation weights; defaults to 1 per observation
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
lambda	Optional user-supplied lambda sequence; default is NULL, and glmreg chooses its own sequence
balance	for family="binomial" only
family	response variable distribution
type	cross-validation criteria. For type="loss", loss function (log-negative-likelihood) values and type="error" is misclassification error if family="binomial".
nfolds	number of folds >=3, default is 10
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing and will be ignored.
plot.it	a logical value, to plot the estimated log-likelihood values if TRUE.
se	a logical value, to plot with standard errors.
parallel, n.cores	a logical value, parallel computing or not with the number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores. The default n.cores=NULL uses all available cores, capped by nfolds. On Unix/macOS, fork-based parallelization is used (minimal startup overhead); on other platforms a PSOCK cluster is created and stopped automatically, also if an error occurs.
trace	a logical value, print progress of cross validation or not
...	Other arguments that can be passed to glmreg.

Details

The function runs glmreg nfolds+1 times; the first to compute the lambda sequence, and then to compute the fit with each of the folds omitted. The error or the log-likelihood value is accumulated, and the average value and standard deviation over the folds is computed. Note that cv.glmreg can be used to search for values for alpha: it is required to call cv.glmreg with a fixed vector foldid for different values of alpha.

Value

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

fit	a fitted glmreg object for the full data.
residmat	matrix of log-likelihood values with row values for lambda and column values for kth cross-validation
cv	The mean cross-validated log-likelihood values - a vector of length(lambda).
cv.error	estimate of standard error of cv.
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in.
lambda	a vector of lambda values
lambda.which	index of lambda that gives maximum cv value.
lambda.optim	value of lambda that gives maximum cv value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

glmreg and plot, predict, and coef methods for "cv.glmreg" object.

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

Description

Does k-fold cross-validation for glmregNB, produces a plot, and returns cross-validated log-likelihood values for `lambda`

Usage

```
cv.glmregNB(formula, data, weights, offset=NULL, lambda=NULL, nfolds=10,
            foldid, plot.it=TRUE, se=TRUE, n.cores=NULL, trace=FALSE,
            parallel=FALSE, ...)
```

Arguments

`formula`	symbolic description of the model
`data`	arguments controlling formula processing via `model.frame`.
`weights`	Observation weights; defaults to 1 per observation
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
`lambda`	Optional user-supplied lambda sequence; default is `NULL`, and `glmregNB` chooses its own sequence
`nfolds`	number of folds - default is 10. Although `nfolds` can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is `nfolds=3`
`foldid`	an optional vector of values between 1 and `nfold` identifying what fold each observation is in. If supplied, `nfold` can be missing.
`plot.it`	a logical value, to plot the estimated log-likelihood values if `TRUE`.
`se`	a logical value, to plot with standard errors.
`n.cores`	The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores.
`trace`	a logical value, print progress of cross-validation or not
`parallel`	a logical value, parallel computing or not
`...`	Other arguments that can be passed to `glmregNB`.

Details

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

Value

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

fit	a fitted glmregNB object for the full data.
residmat	matrix of log-likelihood values with row values for lambda and column values for kth cross-validation
cv	The mean cross-validated log-likelihood values - a vector of length length(lambda).
cv.error	The standard error of cross-validated log-likelihood values - a vector of length length(lambda).
lambda	a vector of lambda values
foldid	indicators of data used in each cross-validation, for reproductive purposes
lambda.which	index of lambda that gives maximum cv value.
lambda.optim	value of lambda that gives maximum cv value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

glmregNB and plot, predict, and coef methods for "cv.glmregNB" object.

Examples

```
## Not run:
data("bioChemists", package = "pscl")
fm_nb <- cv.glmregNB(art ~ ., data = bioChemists)
plot(fm_nb)

## End(Not run)
```

cv.irglmreg *Cross-validation for irglmreg*

Description

Does k-fold cross-validation for irglmreg, produces a plot, and returns cross-validated log-likelihood values for lambda

Usage

```
## S3 method for class 'formula'
cv.irglmreg(formula, data, weights, offset=NULL, ...)
## S3 method for class 'matrix'
cv.irglmreg(x, y, weights, offset=NULL, ...)
## Default S3 method:
cv.irglmreg(x,  ...)
## S3 method for class 'cv.irglmreg'
plot(x,se=TRUE,ylab=NULL, main=NULL, width=0.02, col="darkgrey", ...)
## S3 method for class 'cv.irglmreg'
coef(object,which=object$lambda.which, ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
x	x matrix as in irglmreg. It could be object of cv.irglmreg.
y	response y as in irglmreg.
weights	Observation weights; defaults to 1 per observation
offset	Not implemented yet
object	object of cv.irglmreg
which	Indices of the penalty parameter lambda at which estimates are extracted. By default, the one which generates the optimal cross-validation value.
se	logical value, if TRUE, standard error curve is also plotted
ylab	ylab on y-axis
main	title of plot
width	width of lines
col	color of standard error curve
...	Other arguments that can be passed to irglmreg.

Details

The function runs irglmreg nfolds+1 times; the first to compute the lambda sequence, and then to compute the fit with each of the folds omitted. The error or the loss value is accumulated, and the average value and standard deviation over the folds is computed. Note that cv.irglmreg can be used to search for values for alpha: it is required to call cv.irglmreg with a fixed vector foldid for different values of alpha.

Value

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

fit	a fitted irglmreg object for the full data.
residmat	matrix of log-likelihood values with row values for lambda and column values for kth cross-validation
bic	matrix of BIC values with row values for lambda and column values for kth cross-validation
cv	The mean cross-validated log-likelihood values - a vector of length length(lambda).
cv.error	estimate of standard error of cv.
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in.
lambda	a vector of lambda values
lambda.which	index of lambda that gives minimum cv value.
lambda.optim	value of lambda that gives minimum cv value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irglmreg and plot, predict, and coef methods for "cv.irglmreg" object.

---

cv.irglmreg_fit *Internal function of cross-validation for irglmreg*

Description

Internal function to conduct k-fold cross-validation for irglmreg, produces a plot, and returns cross-validated loss values for lambda

Usage

```
cv.irglmreg_fit(x, y, weights, offset, lambda=NULL, balance=TRUE, cfun=4, dfun=1,
              s=1.5, nfolds=10, foldid, type = c("loss", "error"), plot.it=TRUE,
                se=TRUE, n.cores=NULL, trace=FALSE, parallel=FALSE, ...)
```

Arguments

x	x matrix as in irglmreg.
y	response y as in irglmreg.
weights	Observation weights; defaults to 1 per observation
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
lambda	Optional user-supplied lambda sequence; default is NULL, and irglmreg chooses its own sequence
balance	for dfun=4, 5, 6 only
cfun	a number from 1 to 7, type of convex cap (concave) function
dfun	a number from 1, 4-7, type of convex downward function
s	nonconvex loss tuning parameter for robust regression and classification.
nfolds	number of folds >=3, default is 10
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing and will be ignored.
type	cross-validation criteria. For type="loss", loss function values and type="error" is misclassification error.
plot.it	a logical value, to plot the estimated log-likelihood values if TRUE.
se	a logical value, to plot with standard errors.
n.cores	The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores.
trace	a logical value, print progress of cross validation or not
parallel	a logical value, parallel computing or not
...	Other arguments that can be passed to irglmreg.

Details

The function runs irglmreg nfolds+1 times; the first to compute the lambda sequence, and then to compute the fit with each of the folds omitted. The error or the log-likelihood value is accumulated, and the average value and standard deviation over the folds is computed. Note that cv.irglmreg can be used to search for values for alpha: it is required to call cv.irglmreg with a fixed vector foldid for different values of alpha.

Value

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

fit	a fitted irglmreg object for the full data.
residmat	matrix of loss values or errors with row values for lambda and column values for kth cross-validation

cv	The mean cross-validated loss values or errors - a vector of length length(lambda).
cv.error	estimate of standard error of cv.
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in.
lambda	a vector of lambda values
lambda.which	index of lambda that gives minimum cv value.
lambda.optim	value of lambda that gives minimum cv value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irglmreg and plot, predict, and coef methods for "cv.irglmreg" object.

---

cv.irsvm *Cross-validation for irsvm*

Description

Does k-fold cross-validation for irsvm

Usage

```
## S3 method for class 'formula'
cv.irsvm(formula, data, weights, contrasts=NULL, ...)
## S3 method for class 'matrix'
cv.irsvm(x, y, weights, ...)
## Default S3 method:
cv.irsvm(x,  ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
x	x matrix as in irsvm.
y	response y as in irsvm.
weights	Observation weights; defaults to 1 per observation
contrasts	the contrasts corresponding to levels from the respective models.
...	Other arguments that can be passed to irsvm.

Details

Does a K-fold cross-validation to determine optimal tuning parameters in SVM: `cost` and `gamma` if `kernel` is nonlinear. It can also choose `s` used in `cfun`.

Value

An object contains a list of ingredients of cross-validation including optimal tuning parameters.

`residmat`	matrix with row values for `kernel="linear"` are `s`, `cost`, `error`, `k`, where `k` is the number of cross-validation fold. For nonlinear kernels, row values are `s`, `gamma`, `cost`, `error`, `k`.
`cost`	a value of `cost` that gives minimum cross-validated value in `irsvm`.
`gamma`	a value of `gamma` that gives minimum cross-validated value in `irsvm`
`s`	value of `s` for `cfun` that gives minimum cross-validated value in `irsvm`.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

`irsvm`

Examples

```
## Not run:
x <- matrix(rnorm(40*2), ncol=2)
y <- c(rep(-1, 20), rep(1, 20))
x[y==1,] <- x[y==1, ] + 1
irsvm.opt <- cv.irsvm(x, y, type="C-classification", s=1, kernel="linear", cfun="acave")
irsvm.opt$cost
irsvm.opt$gamma
irsvm.opt$s

## End(Not run)
```

`cv.irsvm_fit` *Internal function of cross-validation for irsvm*

Description

Internal function to conduct k-fold cross-validation for irsvm

Usage

```
cv.irsvm_fit(x, y, weights, cfun="ccave", s=c(1, 5), type=NULL,
             kernel="radial", gamma=2^(-4:10), cost=2^(-4:4),
             epsilon=0.1, balance=TRUE, nfolds=10, foldid,
             trim_ratio=0.9, n.cores=NULL, ...)
```

Arguments

`x`	a data matrix, a vector, or a sparse '*design* matrix' (object of class `Matrix` provided by the **Matrix** package, or of class `matrix.csr` provided by the **SparseM** package, or of class `simple_triplet_matrix` provided by the **slam** package).
`y`	a response vector with one label for each row/component of `x`. Can be either a factor (for classification tasks) or a numeric vector (for regression).
`weights`	the weight of each subject. It should be in the same length of `y`.
`cfun`	character, type of convex cap (concave) function. Valid options are: • `"hcave"` • `"acave"` • `"bcave"` • `"ccave"` • `"dcave"` • `"ecave"` • `"gcave"` • `"tcave"`
`s`	tuning parameter of `cfun`. `s > 0` and can be equal to 0 for `cfun="tcave"`. If `s` is too close to 0 for `cfun="acave"`, `"bcave"`, `"ccave"`, the calculated weights can become 0 for all observations, thus crash the program.
`type`	`irsvm` can be used as a classification machine, or as a regression machine. Depending of whether `y` is a factor or not, the default setting for `type` is `C-classification` or `eps-regression`, respectively, but may be overwritten by setting an explicit value. Valid options are: • `C-classification` • `nu-classification` • `eps-regression`

- nu-regression

kernel, gamma — the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type.

linear: $u'v$
polynomial: $(\gamma u'v + coef0)^{degree}$
radial basis: $e^(- \gamma|u - v|^2)$
sigmoid: $tanh(\gamma u'v + coef0)$

cost — cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation. This is proportional to the inverse of lambda in irglmreg.

epsilon — epsilon in the insensitive-loss function (default: 0.1)

balance — for type="C-classification", "nu-classification" only

nfolds — number of folds >=3, default is 10

foldid — an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing and will be ignored.

trim_ratio — a number between 0 and 1 for trimmed least squares, useful if type="eps-regression" or "nu-regression".

n.cores — The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores.

... — Other arguments that can be passed to irsvm.

Details

This function is the driving force behind cv.irsvm. Does a K-fold cross-validation to determine optimal tuning parameters in SVM: cost and gamma if kernel is nonlinear. It can also choose s used in cfun.

Value

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

residmat — matrix with row values for kernel="linear" are s, cost, error, k, where k is the number of cross-validation fold. For nonlinear kernels, row values are s, gamma, cost, error, k.

cost — a value of cost that gives minimum cross-validated value in irsvm.

gamma — a value of gamma that gives minimum cross-validated value in irsvm

s — value of s for cfun that gives minimum cross-validated value in irsvm.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

cv.irsvm and irsvm

---

cv.nclreg *Cross-validation for nclreg*

Description

Does k-fold cross-validation for nclreg, produces a plot, and returns cross-validated loss values for lambda

Usage

```
## S3 method for class 'formula'
cv.nclreg(formula, data, weights, offset=NULL, ...)
## S3 method for class 'matrix'
cv.nclreg(x, y, weights, offset=NULL, ...)
## Default S3 method:
cv.nclreg(x,  ...)
## S3 method for class 'cv.nclreg'
plot(x,se=TRUE,ylab=NULL, main=NULL, width=0.02, col="darkgrey", ...)
## S3 method for class 'cv.nclreg'
coef(object,which=object$lambda.which, ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
x	x matrix as in nclreg. It could be object of cv.nclreg.
y	response y as in nclreg.
weights	Observation weights; defaults to 1 per observation
offset	Not implemented yet
object	object of cv.nclreg
which	Indices of the penalty parameter lambda at which estimates are extracted. By default, the one which generates the optimal cross-validation value.
se	logical value, if TRUE, standard error curve is also plotted
ylab	ylab on y-axis
main	title of plot
width	width of lines
col	color of standard error curve
...	Other arguments that can be passed to nclreg.

Details

The function runs `nclreg` `nfolds+1` times; the first to compute the `lambda` sequence, and then to compute the fit with each of the folds omitted. The error or the loss value is accumulated, and the average value and standard deviation over the folds is computed. Note that `cv.nclreg` can be used to search for values for `alpha`: it is required to call `cv.nclreg` with a fixed vector `foldid` for different values of `alpha`.

Value

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

`fit`	a fitted nclreg object for the full data.
`residmat`	matrix of loss values with row values for `lambda` and column values for `k`th cross-validation
`bic`	matrix of BIC values with row values for `lambda` and column values for `k`th cross-validation
`cv`	The mean cross-validated loss values - a vector of length `length(lambda)`.
`cv.error`	estimate of standard error of `cv`.
`foldid`	an optional vector of values between 1 and `nfold` identifying what fold each observation is in.
`lambda`	a vector of `lambda` values
`lambda.which`	index of `lambda` that gives minimum `cv` value.
`lambda.optim`	value of `lambda` that gives minimum `cv` value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2021), *MM for Penalized Estimation*, *TEST*, doi:10.1007/s11749021007702

See Also

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

`cv.nclreg_fit` *Internal function of cross-validation for nclreg*

Description

Internal function to conduct k-fold cross-validation for nclreg, produces a plot, and returns cross-validated loss values for `lambda`

Usage

```
cv.nclreg_fit(x, y, weights, offset, lambda=NULL, balance=TRUE,
              rfamily=c("clossR", "closs", "gloss", "qloss"), s=1.5,
              nfolds=10, foldid, type = c("loss", "error"),
              plot.it=TRUE, se=TRUE, n.cores=NULL, trace=FALSE,
              parallel=FALSE, ...)
```

Arguments

`x`	x matrix as in `nclreg`.
`y`	response `y` as in `nclreg`.
`weights`	Observation weights; defaults to 1 per observation
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
`lambda`	Optional user-supplied lambda sequence; default is `NULL`, and `nclreg` chooses its own sequence
`balance`	for `rfamily="closs"`, `"gloss"`, `"qloss"` only
`rfamily`	response variable distribution and nonconvex loss function
`s`	nonconvex loss tuning parameter for robust regression and classification.
`nfolds`	number of folds >=3, default is 10
`foldid`	an optional vector of values between 1 and `nfold` identifying what fold each observation is in. If supplied, `nfold` can be missing and will be ignored.
`type`	cross-validation criteria. For `type="loss"`, loss function values and `type="error"` is misclassification error.
`plot.it`	a logical value, to plot the estimated loss values if `TRUE`.
`se`	a logical value, to plot with standard errors.
`n.cores`	The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores.
`trace`	a logical value, print progress of cross validation or not
`parallel`	a logical value, parallel computing or not
`...`	Other arguments that can be passed to `nclreg`.

Details

The function runs `nclreg` `nfolds+1` times; the first to compute the `lambda` sequence, and then to compute the fit with each of the folds omitted. The error or the loss value is accumulated, and the average value and standard deviation over the folds is computed. Note that `cv.nclreg` can be used to search for values for `alpha`: it is required to call `cv.nclreg` with a fixed vector `foldid` for different values of `alpha`.

Value

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

`fit`	a fitted nclreg object for the full data.
`residmat`	matrix of loss values or errors with row values for `lambda` and column values for kth cross-validation
`cv`	The mean cross-validated loss values or errors - a vector of length `length(lambda)`.
`cv.error`	estimate of standard error of `cv`.
`foldid`	an optional vector of values between 1 and `nfold` identifying what fold each observation is in.
`lambda`	a vector of `lambda` values
`lambda.which`	index of `lambda` that gives minimum `cv` value.
`lambda.optim`	value of `lambda` that gives minimum `cv` value.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2021), *MM for Penalized Estimation*, *TEST*, doi:10.1007/s11749021007702

See Also

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

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

Description

Does k-fold cross-validation for zipath, produces a plot, and returns cross-validated log-likelihood values for `lambda`

Usage

```
## S3 method for class 'formula'
cv.zipath(formula, data, weights, offset=NULL, contrasts=NULL, ...)
## S3 method for class 'matrix'
cv.zipath(X, Z, Y, weights, offsetx=NULL, offsetz=NULL, ...)
## Default S3 method:
cv.zipath(X, ...)
## S3 method for class 'cv.zipath'
predict(object, newdata, ...)
## S3 method for class 'cv.zipath'
coef(object, which=object$lambda.which, model = c("full", "count", "zero"), ...)
```

Arguments

formula	symbolic description of the model with an optional numeric vector `offset` with an a priori known component to be included in the linear predictor of the count model or zero model. Offset must be a variable in `data` if used, while this is optional in `zipath`. See an example below.
data	arguments controlling formula processing via `model.frame`.
weights	Observation weights; defaults to 1 per observation
offset	optional numeric vector with an a priori known component to be included in the linear predictor of the count model or zero model. See below for an example.
X	predictor matrix of the count model
Z	predictor matrix of the zero model
Y	response variable
offsetx, offsetz	optional numeric vector with an a priori known component to be included in the linear predictor of the count model (offsetx)or zero model (offsetz).
contrasts	a list with elements `"count"` and `"zero"` containing the contrasts corresponding to `levels` from the respective models
object	object of class `cv.zipath`.
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
which	Indices of the pair of penalty parameters `lambda.count` and `lambda.zero` at which estimates are extracted. By default, the one which generates the optimal cross-validation value.
model	character specifying for which component of the model the estimated coefficients should be extracted.
...	Other arguments that can be passed to `zipath`.

Details

The function runs `zipath` `nfolds`+1 times; the first to compute the `(lambda.count, lambda.zero)` sequence, and then to compute the fit with each of the folds omitted. The model is fitted to the training data and then given the fitted model the log-likelihood is evaluated at the observations left out,

i.e., the test data. The average value of log-likelihood and standard deviation over the folds is computed. Note that `cv.zipath` can be used to search for values for `count.alpha` or `zero.alpha`: it is required to call `cv.zipath` with a fixed vector `foldid` for different values of `count.alpha` or `zero.alpha`.

The methods for `coef` and `predict` were deprecated since version 0.3-25. In fact, the `fit` object was removed in the output of cv.zipath so that predict an object of cv.zipath is not feasible, and should be via zipath. See examples below. The reason for such a change is that cv.zipath can take both formula and matrix, hence `predict` on cv. zipath object can easily lead to problems in codes.

When `family="negbin"`, it can be slow because there is a repeated search for the `theta` values by default. One may change the default values from `init.theta=NULL, theta.fixed=FALSE` to `init.theta=MLE, theta.fixed=TRUE`, where MLE is a number from glm.nb in the R package MASS or something desired.

Value

an object of class `"cv.zipath"` is returned, which is a list with the components of the cross-validation fit.

`fit`	a fitted zipath object for the full data.
`residmat`	matrix for cross-validated log-likelihood at each `(count.lambda, zero.lambda)` sequence
`bic`	matrix of BIC values with row values for `lambda` and column values for `k`th cross-validation
`cv`	The mean cross-validated log-likelihood - a vector of length `length(count.lambda)`.
`cv.error`	estimate of standard error of `cv`.
`foldid`	an optional vector of values between 1 and `nfold` identifying what fold each observation is in.
`lambda.which`	index of `(count.lambda, zero.lambda)` that gives maximum `cv`.
`lambda.optim`	value of `(count.lambda, zero.lambda)` that gives maximum `cv`.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath and plot, predict, methods for "cv.zipath" object.

Examples

```
## Not run:
data("bioChemists", package = "pscl")
fm_zip <- zipath(art ~ . | ., data = bioChemists, family = "poisson", nlambda=10)
fm_cvzip <- cv.zipath(art ~ . | ., data = bioChemists, family = "poisson", nlambda=10)
### prediction from the best model
pred <- predict(fm_zip, newdata=bioChemists, which=fm_cvzip$lambda.which)
coef(fm_zip, which=fm_cvzip$lambda.which)
fm_znb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)
fm_cvznb <- cv.zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)
pred <- predict(fm_znb, which=fm_cvznb$lambda.which)
coef(fm_znb, which=fm_cvznb$lambda.which)
fm_zinb2 <- zipath(art ~ . +offset(log(phd))| ., data = bioChemists,
        family = "poisson", nlambda=10)
fm_cvzinb2 <- cv.zipath(art ~ . +offset(log(phd))| ., data = bioChemists,
        family = "poisson", nlambda=10)
pred <- predict(fm_zinb2, which=fm_cvzinb2$lambda.which)
coef(fm_zinb2, which=fm_cvzinb2$lambda.which)

## End(Not run)
```

---

cv.zipath_fit *Cross-validation for zipath*

Description

Internal function k-fold cross-validation for zipath, produces a plot, and returns cross-validated log-likelihood values for lambda

Usage

```
cv.zipath_fit(X, Z, Y, weights, offsetx, offsetz, nlambda=100, lambda.count=NULL,
              lambda.zero=NULL, nfolds=10, foldid, plot.it=TRUE, se=TRUE,
              n.cores=NULL, trace=FALSE, parallel=FALSE, ...)
```

Arguments

X	predictor matrix of the count model
Z	predictor matrix of the zero model
Y	response variable
weights	optional numeric vector of weights.
offsetx	optional numeric vector with an a priori known component to be included in the linear predictor of the count model.

offsetz	optional numeric vector with an a priori known component to be included in the linear predictor of the zero model.
nlambda	number of lambda value, default value is 10.
lambda.count	Optional user-supplied lambda.count sequence; default is NULL
lambda.zero	Optional user-supplied lambda.zero sequence; default is NULL
nfolds	number of folds >=3, default is 10
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing and will be ignored.
plot.it	a logical value, to plot the estimated log-likelihood values if TRUE.
se	a logical value, to plot with standard errors.
n.cores	The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores.
trace	a logical value, print progress of cross-validation or not
parallel	a logical value, parallel computing or not
...	Other arguments that can be passed to zipath.

Details

The function runs zipath nfolds+1 times; the first to compute the (lambda.count, lambda.zero) sequence, and then to compute the fit with each of the folds omitted. The log-likelihood value is accumulated, and the average value and standard deviation over the folds is computed. Note that cv.zipath can be used to search for values for count.alpha or zero.alpha: it is required to call cv.zipath with a fixed vector foldid for different values of count.alpha or zero.alpha.

The method for coef by default return a single vector of coefficients, i.e., all coefficients are concatenated. By setting the model argument, the estimates for the corresponding model components can be extracted.

Value

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

fit	a fitted zipath object for the full data.
residmat	matrix for cross-validated log-likelihood at each (count.lambda, zero.lambda) sequence
bic	matrix of BIC values with row values for lambda and column values for kth cross-validation
cv	The mean cross-validated log-likelihood - a vector of length length(count.lambda).
cv.error	estimate of standard error of cv.
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in.
lambda.which	index of (count.lambda, zero.lambda) that gives maximum cv.
lambda.optim	value of (count.lambda, zero.lambda) that gives maximum cv.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

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

`docvisits` *Doctor visits*

Description

A cohort of 3066 Americans over the age of 50 were studied on health care utilization, doctor office visits.

Usage

```
data(docvisits)
```

Source

Stephane Heritier, Eva Cantoni, Samuel Copt and Maria-Pia Victoria-Fese (2009). *Robust Methods in Biostatistics*, John Wiley & Sons

Examples

```
data(docvisits)
str(docvisits)
```

estfunReg *Extract Empirical First Derivative of Log-likelihood Function*

Description

Generic function for extracting the empirical first derivative of log-likelihood function of a fitted regularized model.

Usage

```
estfunReg(x, ...)
```

Arguments

x	a fitted model object.
...	arguments passed to methods.

Value

A matrix containing the empirical first derivative of log-likelihood functions. Typically, this should be an $n \times k$ matrix corresponding to n observations and k parameters. The columns should be named as in coef or terms, respectively.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath

Examples

```
data("bioChemists", package = "pscl")
fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10, maxit.em=1)
res <- estfunReg(fm_zinb, which=which.min(fm_zinb$bic))
```

gfunc *Convert response value to raw prediction in GLM*

Description

Compute response value to raw prediction such as linear predictor in GLM

Usage

```
gfunc(mu, family, epsbino)
```

Arguments

mu	vector of numbers as response value in GLM, for instance, probability estimation if family=2
family	integer from 1-4, corresponding to "gaussian", "binomial", "poisson", "negbin", respectively
epsbino	a small positive value for family=2 to avoid numeric unstability

Value

linear predictor f=x'b for predictor x and coefficient b if the model is linear

glmreg *fit a GLM with lasso (or elastic net), snet or mnet regularization*

Description

Fit a generalized linear model via penalized maximum likelihood. The regularization path is computed for the lasso (or elastic net penalty), scad (or snet) and mcp (or mnet penalty), at a grid of values for the regularization parameter lambda. Fits linear, logistic, Poisson and negative binomial (fixed scale parameter) regression models.

Usage

```
## S3 method for class 'formula'
glmreg(formula, data, weights, offset=NULL, contrasts=NULL,
x.keep=FALSE, y.keep=TRUE, ...)
## S3 method for class 'matrix'
glmreg(x, y, weights, offset=NULL, ...)
## Default S3 method:
glmreg(x,  ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
weights	optional numeric vector of weights. If standardize=TRUE, weights are renormalized to weights/sum(weights). If standardize=FALSE, weights are kept as original input
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
x	input matrix, of dimension nobs x nvars; each row is an observation vector
y	response variable. Quantitative for family="gaussian". Non-negative counts for family="poisson" or family="negbin". For family="binomial" should be either a factor with two levels or a vector of proportions.
x.keep, y.keep	logical values: keep response variables or keep response variable?
contrasts	the contrasts corresponding to levels from the respective models
...	Other arguments passing to glmreg_fit

Details

The sequence of models implied by lambda is fit by coordinate descent. For family="gaussian" this is the lasso, mcp or scad sequence if alpha=1, else it is the enet, mnet or snet sequence. For the other families, this is a lasso (mcp, scad) or elastic net (mnet, snet) regularization path for fitting the generalized linear regression paths, by maximizing the appropriate penalized log-likelihood. Note that the objective function for "gaussian" is

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

if standardize=FALSE and

$$1/2 * \frac{weights}{\sum(weights)} * RSS + \lambda * penalty,$$

if standardize=TRUE. For the other models it is

$$-\sum(weights * loglik) + \lambda * penalty$$

if standardize=FALSE and

$$-\frac{weights}{\sum(weights)} * loglik + \lambda * penalty$$

if standardize=TRUE.

Value

An object with S3 class "glmreg" for the various types of models.

call	the call that produced this object

b0	Intercept sequence of length length(lambda)
beta	A nvars x length(lambda) matrix of coefficients.
lambda	The actual sequence of lambda values used
offset	the offset vector used.
resdev	The computed deviance (for "gaussian", this is the R-square). The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation).
nulldev	Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)); The NULL model refers to the intercept model.
nobs	number of observations
pll	penalized log-likelihood values for standardized coefficients in the IRLS iterations. For family="gaussian", not implemented yet.
pllres	penalized log-likelihood value for the estimated model on the original scale of coefficients
fitted.values	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Breheny, P. and Huang, J. (2011) *Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection*. *Ann. Appl. Statist.*, **5**: 232-253.

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

print, predict, coef and plot methods, and the cv.glmreg function.

Examples

```
#binomial
x=matrix(rnorm(100*20),100,20)
g2=sample(0:1,100,replace=TRUE)
fit2=glmreg(x,g2,family="binomial")
#poisson and negative binomial
data("bioChemists", package = "pscl")
fm_pois <- glmreg(art ~ ., data = bioChemists, family = "poisson")
coef(fm_pois)
fm_nb1 <- glmreg(art ~ ., data = bioChemists, family = "negbin", theta=1)
coef(fm_nb1)
#offset
```

```
x <- matrix(rnorm(100*20),100,20)
y <- rpois(100, lambda=1)
exposure <- rep(0.5, length(y))
fit2 <- glmreg(x,y, lambda=NULL, nlambda=10, lambda.min.ratio=1e-4,
        offset=log(exposure), family="poisson")
predict(fit2, newx=x, newoffset=log(exposure))
## Not run:
fm_nb2 <- glmregNB(art ~ ., data = bioChemists)
coef(fm_nb2)

## End(Not run)
```

`glmreg_fit` *Internal function to fit a GLM with lasso (or elastic net), snet and mnet regularization*

Description

Fit a generalized linear model via penalized maximum likelihood. The regularization path is computed for the lasso (or elastic net penalty), snet and mnet penalty, at a grid of values for the regularization parameter lambda. Fits linear, logistic, Poisson and negative binomial (fixed scale parameter) regression models.

Usage

```
glmreg_fit(x, y, weights, start=NULL, etastart=NULL, mustart=NULL, offset = NULL,
        nlambda=100, lambda=NULL, lambda.min.ratio=ifelse(nobs<nvars,.05, .001),
           alpha=1, gamma=3, rescale=TRUE, standardize=TRUE, intercept=TRUE,
         penalty.factor = rep(1, nvars), thresh=1e-6, eps.bino=1e-5, maxit=1000,
           eps=.Machine$double.eps, theta,
           family=c("gaussian", "binomial", "poisson", "negbin"),
       penalty=c("enet","mnet","snet"), convex=FALSE, x.keep=FALSE, y.keep=TRUE,
           trace=FALSE)
```

Arguments

`x`	input matrix, of dimension nobs x nvars; each row is an observation vector.
`y`	response variable. Quantitative for `family="gaussian"`. Non-negative counts for `family="poisson"` or `family="negbin"`. For `family="binomial"` should be either a factor with two levels or a vector of proportions.
`weights`	observation weights. Can be total counts if responses are proportion matrices. Default is 1 for each observation
`start`	starting values for the parameters in the linear predictor.
`etastart`	starting values for the linear predictor.
`mustart`	starting values for the vector of means.

`offset` this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.

`nlambda` The number of `lambda` values - default is 100. The sequence may be truncated before `nlambda` is reached if a close to saturated model is fitted. See also `satu`.

`lambda` by default, the algorithm provides a sequence of regularization values, or a user supplied `lambda` sequence. When `alpha=0`, the largest lambda value is not defined (infinity). Thus, the largest `lambda` for `alpha=0.001` is computed, and the sequence of `lambda` values is calculated afterwards.

`lambda.min.ratio`
Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero except the intercept). Note, there is no closed formula for `lambda.max` in general. If `rescale=TRUE`, `lambda.max` is the same for `penalty="mnet"` or `"snet"`. Otherwise, some modifications are required. For instance, for small `gamma` value, half of the square root (if `lambda.max` is too small) of the computed `lambda.max` can be used when `penalty="mnet"` or `"snet"`. The default of `lambda.min.ratio` depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs` > `nvars`, the default is `0.001`, close to zero. If `nobs` < `nvars`, the default is `0.05`.

`alpha` The L_2 penalty mixing parameter, with $0 \leq alpha \leq 1$. `alpha=1` is lasso (mcp, scad) penalty; and `alpha=0` the ridge penalty. However, if `alpha=0`, one must provide `lambda` values.

`gamma` The tuning parameter of the `snet` or `mnet` penalty.

`rescale` logical value, if TRUE, adaptive rescaling of the penalty parameter for `penalty="mnet"` or `penalty="snet"` with `family` other than "gaussian". See reference

`standardize` logical value for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is `standardize=TRUE`.

`intercept` logical value: if TRUE (default), intercept(s) are fitted; otherwise, intercept(s) are set to zero

`penalty.factor` This is a number that multiplies `lambda` to allow differential shrinkage of coefficients. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is same shrinkage for all variables.

`thresh` Convergence threshold for coordinate descent. Defaults value is `1e-6`.

`eps.bino` a lower bound of probabilities to be truncated, for computing weights and related values when `family="binomial"`. It is also used when `family="negbin"`.

`maxit` Maximum number of coordinate descent iterations for each `lambda` value; default is 1000.

`eps` If a coefficient is less than `eps` in magnitude, then it is reported to be 0

`convex` Calculate index for which objective function ceases to be locally convex? Default is FALSE and only useful if `penalty="mnet"` or `"snet"`.

`theta` an overdispersion scaling parameter for `family="negbin"`

family	Response type (see above)
penalty	Type of regularization
x.keep, y.keep	For glmreg: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glmreg_fit: x is a design matrix of dimension n * p, and x is a vector of observations of length n.
trace	If TRUE, fitting progress is reported

Details

The sequence of models implied by lambda is fit by coordinate descent. For family="gaussian" this is the lasso, mcp or scad sequence if alpha=1, else it is the enet, mnet or snet sequence. For the other families, this is a lasso (mcp, scad) or elastic net (mnet, snet) regularization path for fitting the generalized linear regression paths, by maximizing the appropriate penalized log-likelihood. Note that the objective function for "gaussian" is

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

if standardize=FALSE and

$$1/2 * \frac{weights}{\sum(weights)} * RSS + \lambda * penalty,$$

if standardize=TRUE. For the other models it is

$$-\sum(weights * loglik) + \lambda * penalty$$

if standardize=FALSE and

$$-\frac{weights}{\sum(weights)} * loglik + \lambda * penalty$$

if standardize=TRUE.

Value

An object with S3 class "glmreg" for the various types of models.

call	the call that produced the model fit
b0	Intercept sequence of length length(lambda)
beta	A nvars x length(lambda) matrix of coefficients.
lambda	The actual sequence of lambda values used
satu	satu=1 if a saturated model (deviance/null deviance < 0.05) is fit. Otherwise satu=0. The number of nlambda sequence may be truncated before nlambda is reached if satu=1.
dev	The computed deviance (for "gaussian", this is the R-square). The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation).
nulldev	Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)); The NULL model refers to the intercept model.
nobs	number of observations

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Breheny, P. and Huang, J. (2011) *Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection*. *Ann. Appl. Statist.*, **5**: 232-253.

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

glmreg

`glmregNB` *fit a negative binomial model with lasso (or elastic net), snet and mnet regularization*

Description

Fit a negative binomial linear model via penalized maximum likelihood. The regularization path is computed for the lasso (or elastic net penalty), snet and mnet penalty, at a grid of values for the regularization parameter lambda.

Usage

```
glmregNB(formula, data, weights, offset=NULL, nlambda = 100, lambda=NULL,
         lambda.min.ratio = ifelse(nobs<nvars,0.05,0.001), alpha=1, gamma=3,
         rescale=TRUE, standardize = TRUE, penalty.factor = rep(1, nvars),
         thresh = 0.001, maxit.theta = 10, maxit=1000, eps=.Machine$double.eps,
         trace=FALSE, start = NULL, etastart = NULL, mustart = NULL,
         theta.fixed=FALSE, theta0=NULL, init.theta=NULL, link=log,
         penalty=c("enet","mnet","snet"), method="glmreg_fit", model=TRUE,
         x.keep=FALSE, y.keep=TRUE, contrasts=NULL, convex=FALSE,
         parallel=TRUE, n.cores=NULL, ...)
```

Arguments

`formula`	formula used to describe a model.
`data`	argument controlling formula processing via `model.frame`.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector. Default is a vector of 1s with equal weight for each observation.
`offset`	optional numeric vector with an a priori known component to be included in the linear predictor of the model.

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

`lambda` A user supplied `lambda` sequence

`lambda.min.ratio`
Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs` > `nvars`, the default is `0.001`, close to zero. If `nobs` < `nvars`, the default is `0.05`.

`alpha` The L2 penalty mixing parameter, with $0 \leq \alpha \leq 1$. `alpha=1` is lasso (mcp, scad) penalty; and `alpha=0` the ridge penalty.

`gamma` The tuning parameter of the `snet` or `mnet` penalty.

`rescale` logical value, if TRUE, adaptive rescaling of the penalty parameter for `penalty="mnet"` or `penalty="snet"` with `family` other than "gaussian". See reference

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

`penalty.factor` This is a number that multiplies `lambda` to allow differential shrinkage of coefficients. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is same shrinkage for all variables.

`thresh` Convergence threshold for coordinate descent. Defaults value is `1e-6`.

`maxit.theta` Maximum number of iterations for estimating `theta` scaling parameter

`maxit` Maximum number of coordinate descent iterations for each `lambda` value; default is 1000.

`eps` If a number is less than `eps` in magnitude, then this number is considered as 0

`trace` If `TRUE`, fitting progress is reported

`start, etastart, mustart, ...`
arguments for the `link{glmreg}` function

`init.theta` initial scaling parameter `theta`

`theta.fixed` Estimate scale parameter theta? Default is FALSE. Note, the algorithm may become slow. In this case, one may use `glmreg` function with `family="negbin"`, and a fixed `theta`

.

`theta0` initial scale parameter vector theta, with length `nlambda` if `theta.fixed=TRUE`. Default is NULL

`convex` Calculate index for which objective function ceases to be locally convex? Default is FALSE and only useful if `penalty="mnet"` or `"snet"`.

`link` link function, default is `log`

`penalty` Type of regularization

`method` estimation method

`model, x.keep, y.keep`
logicals. If `TRUE` the corresponding components of the fit (model frame, response, model matrix) are returned.

`contrasts` the contrasts corresponding to `levels` from the respective models

`parallel, n.cores`
a logical value, parallel computing or not for sequence of `lambda` with the number of CPU cores to use. The `lambda` loop will attempt to send different `lambda` off to different cores.

Details

The sequence of models implied by `lambda` is fit by coordinate descent. This is a lasso (mcp, scad) or elastic net (mnet, snet) regularization path for fitting the negative binomial linear regression paths, by maximizing the penalized log-likelihood. Note that the objective function is

$$-\sum(weights * loglik) + \lambda * penalty$$

if `standardize=FALSE` and

$$-\frac{weights}{\sum(weights)} * loglik + \lambda * penalty$$

if `standardize=TRUE`.

Value

An object with S3 class `"glmreg", "glmregNB"` for the various types of models.

`call`	the call that produced the model fit
`b0`	Intercept sequence of length `length(lambda)`
`beta`	A `nvars` x `length(lambda)` matrix of coefficients.
`lambda`	The actual sequence of `lambda` values used
`resdev`	The computed deviance. The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation).
`nulldev`	Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)); The NULL model refers to the intercept model.
`nobs`	number of observations

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

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

Examples

```
## Not run:
data("bioChemists", package = "pscl")
system.time(fm_nb1 <- glmregNB(art ~ ., data = bioChemists, parallel=FALSE))
system.time(fm_nb2 <- glmregNB(art ~ ., data = bioChemists, parallel=TRUE, n.cores=NULL))
coef(fm_nb1)
### ridge regression
fm <- glmregNB(art ~ ., alpha=0, data = bioChemists, lambda=seq(0.001, 1, by=0.01))
fm <- cv.glmregNB(art ~ ., alpha=0, data = bioChemists, lambda=seq(0.001, 1, by=0.01))

## End(Not run)
```

---

hessianReg *Hessian Matrix of Regularized Estimators*

Description

Constructing Hessian matrix for regularized regression parameters.

Usage

```
hessianReg(x, which, ...)
```

Arguments

x	a fitted model object.
which	which penalty parameter(s)?
...	arguments passed to the meatReg function.

Details

hessianReg is a function to compute the Hessian matrix estimate of non-zero regularized estimators. Implemented only for zipath object with family="negbin" in the current version.

Value

A matrix containing the Hessian matrix estimate for the non-zero parameters.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

breadReg, meatReg

Examples

```
data("bioChemists", package = "pscl")
fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10, maxit.em=1)
hessianReg(fm_zinb, which=which.min(fm_zinb$bic))
```

---

irglm *fit a robust generalized linear models*

Description

Fit a robust GLM where the loss function is a composite function cfunodfun.

Usage

```
## S3 method for class 'formula'
irglm(formula, data, weights, offset=NULL, contrasts=NULL,
 cfun="ccave", dfun=gaussian(), s=NULL, delta=0.1, fk=NULL, init.family=NULL,
 iter=10, reltol=1e-5, theta, x.keep=FALSE, y.keep=TRUE, trace=FALSE, ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
weights	optional numeric vector of weights.
x	input matrix, of dimension nobs x nvars; each row is an observation vector
y	response variable. Quantitative for dfun=1 and -1/1 for classification.
contrasts	the contrasts corresponding to levels from the respective models
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
cfun	character, type of concave function, which has a cap shape. Valid options are: • "hcave" • "acave" • "bcave" • "ccave" • "dcave" • "ecave" • "gcave"

- "tcave"

dfun — character, type of convex component. Convex is often referred to as convex down: if you connect any two points on the curve by a line segment then the segment is always below the curve.
Valid options are:

- gaussian()
- binomial()
- poisson()

init.family — character value for initial family, one of "clossR","closs","gloss","qloss", which can be used to derive an initial estimator, if the selection is different from the default value

s — tuning parameter of cfun. s > 0 and can be equal to 0 for cfun="tcave". If s is too close to 0 for cfun="acave", "bcave", "ccave", the calculated weights can become 0 for all observations, thus crash the program.

delta — a small positive number provided by user only if cfun="gcave" and 0 < s <1

fk — predicted values at an iteration in the IRGLM algorithm

iter — number of iteration in the IRGLM algorithm

reltol — convergency criteria in the IRGLM algorithm

theta — an overdispersion scaling parameter for family=negbin()

x.keep, y.keep — logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value, x is a design matrix of dimension n * p, and x is a vector of observations of length n.

trace — if TRUE, fitting progress is reported

... — other arguments passing to irglm

Details

A robust linear, logistic or Poisson regression model is fit by the iteratively reweighted GLM (IRGLM). The output weights_update is a useful diagnostic to the outlier status of the observations.

Value

An object with S3 class "irglm", "glm" for various types of models.

call — the call that produced the model fit

weights — original weights used in the model

weights_update — weights in the final iteration of the IRGLM algorithm

cfun, s, dfun — original input arguments

is.offset — is offset used?

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

print, predict, coef.

Examples

```
x=matrix(rnorm(100*20),100,20)
g2=sample(c(-1,1),100,replace=TRUE)
fit=irglm(g2~x,data=data.frame(cbind(x, g2)), s=1, cfun="ccave", dfun=gaussian())
fit$weights_update
```

---

irglmreg *Fit a robust penalized generalized linear models*

Description

Fit a robust penalized GLM where the loss function is a composite function cfunodfun + penalty. This is the wrapper function of irglmreg_fit

Usage

```
## S3 method for class 'formula'
irglmreg(formula, data, weights, offset=NULL, contrasts=NULL, ...)
## S3 method for class 'matrix'
irglmreg(x, y, weights, offset=NULL, ...)
## Default S3 method:
irglmreg(x,  ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
weights	optional numeric vector of weights. If standardize=TRUE, weights are renormalized to weights/sum(weights). If standardize=FALSE, weights are kept as original input
x	input matrix, of dimension nobs x nvars; each row is an observation vector
y	response variable. Quantitative for rfamily="clossR" and -1/1 for classification.
offset	Not implemented yet
contrasts	the contrasts corresponding to levels from the respective models
...	Other arguments passing to irglmreg_fit

Details

The computing is done by the iteratively reweighted penalized GLM, an application of the iteratively reweighted convex optimization (IRCO). Here convex is the loss function induced by `dfun`, not the penalty function. The output `weights_update` is a useful diagnostic to the outlier status of the observations. The regularization path is computed for the lasso (or elastic net penalty), scad (or snet) and mcp (or mnet penalty), at a grid of values for the regularization parameter lambda. The sequence of robust models implied by `lambda` is fit by the IRCO along with coordinate descent. Note that the objective function is

$$weights * loss + \lambda * penalty,$$

if `standardize=FALSE` and

$$\frac{weights}{\sum(weights)} * loss + \lambda * penalty,$$

if `standardize=TRUE`.

Value

An object with S3 class `"irglmreg"` for the various types of models.

`call`	the call that produced this object
`b0`	Intercept sequence of length `length(lambda)`
`beta`	A `nvars` x `length(lambda)` matrix of coefficients.
`lambda`	The actual sequence of `lambda` values used
`nobs`	number of observations
`risk`	if `type.path="nonactive"`, a matrix with number of rows `iter` and number of columns `nlambda`, loss values along the regularization path. If `type.path="fast"`, a vector of length `nlambda`, loss values along the regularization path
`pll`	if `type.path="nonactive"`, a matrix with number of rows `iter` and number of columns `nlambda`, penalized loss values along the regularization path. If `type.path="fast"`, a vector of length `nlambda`, penalized loss values along the regularization path
`fitted.values`	predicted values depending on `standardize`, internal use only

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

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

Examples

```
#binomial
x=matrix(rnorm(100*20),100,20)
g2=sample(c(-1,1),100,replace=TRUE)
fit1=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="active",
              decreasing=TRUE,type.init="bst")
#fit1$risk
## Not run:
### different solution paths via a combination of type.path, decreasing and type.init
fit1=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="active",
           decreasing=TRUE,type.init="bst")
fit2=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="active",
           decreasing=FALSE,type.init="bst")
fit3=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="nonactive",
           decreasing=TRUE,type.init="bst")
fit4=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="nonactive",
           decreasing=FALSE,type.init="bst")
fit5=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="active",
           decreasing=TRUE,type.init="co")
fit6=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="active",
           decreasing=FALSE,type.init="co")
fit7=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="nonactive",
           decreasing=TRUE,type.init="co")
fit8=irglmreg(x,g2,s=1,cfun="ccave",dfun="gaussian",type.path="nonactive",
           decreasing=FALSE,type.init="co")

## End(Not run)
```

`irglmreg_fit` *Internal function for robust penalized generalized linear models*

Description

Fit a robust penalized GLM where the loss function is a composite function `cfunodfun` + penalty. This does computing for `irglmreg`.

Usage

```
irglmreg_fit(x, y, weights, offset, cfun="ccave", dfun="gaussian", s=NULL,
             delta=0.1, fk=NULL, iter=10, reltol=1e-5,
             penalty=c("enet","mnet","snet"), nlambda=100, lambda=NULL,
             type.path=c("active", "nonactive"), decreasing=TRUE,
             lambda.min.ratio=ifelse(nobs<nvars,.05, .001), alpha=1, gamma=3,
             rescale=TRUE, standardize=TRUE, intercept=TRUE,
             penalty.factor= NULL, maxit=1000, type.init=c("bst", "co", "heu"),
             init.family=NULL, mstop.init=10, nu.init=0.1,
             eps=.Machine$double.eps, epscycle=10, thresh=1e-6, parallel=FALSE,
             n.cores=NULL, theta, trace=FALSE, tracelevel=1)
```

Arguments

x	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable. Quantitative for dfun=1 and -1/1 otherwise for classifications.
weights	observation weights. Can be total counts if responses are proportion matrices. Default is 1 for each observation
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
cfun	character, type of convex cap (concave) function. Valid options are: • "hcave" • "acave" • "bcave" • "ccave" • "dcave" • "ecave" • "gcave" • "tcave"
dfun	character, type of convex downward function. Valid options are: • "gaussian" • "gaussianC" • "binomial"
s	tuning parameter of cfun. s > 0 and can be equal to 0 for cfun="tcave". If s is too close to 0 for cfun="acave", "bcave", "ccave", the calculated weights can become 0 for all observations, thus crash the program.
delta	a small positive number provided by user only if cfun="gcave" and 0 < s <1
fk	predicted values at an iteration in the IRCO algorithm
nlambda	The number of lambda values - default is 100. The sequence may be truncated before nlambda is reached if a close to saturated model is fitted. See also satu.
lambda	by default, the algorithm provides a sequence of regularization values, or a user supplied lambda sequence
type.path	solution path for parallel=FALSE. If type.path="active", then cycle through only the active set in the next increasing lambda sequence. If type.path="nonactive", no active set for each element of the lambda sequence and cycle through all the predictor variables.
lambda.min.ratio	Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero except the intercept). Note, there is no closed formula for lambda.max. The default of lambda.min.ratio depends on the sample size nobs relative to the number of variables nvars. If nobs > nvars, the default is 0.001, close to zero. If nobs < nvars, the default is 0.05.

alpha	The L_2 penalty mixing parameter, with $0 \leq alpha \leq 1$. alpha=1 is lasso (mcp, scad) penalty; and alpha=0 the ridge penalty. However, if alpha=0, one must provide lambda values.
gamma	The tuning parameter of the snet or mnet penalty.
rescale	logical value, if TRUE, adaptive rescaling of the penalty parameter for penalty="mnet" or penalty="snet" with dfun="binomial". See glmreg_fit
standardize	logical value for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is standardize=TRUE.
intercept	logical value: if TRUE (default), intercept(s) are fitted; otherwise, intercept(s) are set to zero
penalty.factor	This is a number that multiplies lambda to allow differential shrinkage of coefficients. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is same shrinkage for all variables.
type.init	a method to determine the initial values. If type.init="ncl", an intercept-only model as initial parameter and run irglmreg regularization path forward from lambda_max to lambda_min. If type.init="heu", heuristic initial parameters and run irglmreg path backward or forward depending on decreasing, between lambda_min and lambda_max. If type.init="bst", run a boosting model with bst in package bst, depending on mstop.init, nu.init and run irglmreg backward or forward depending on decreasing.
init.family	character value for initial family, one of "clossR", "closs","gloss","qloss", which can be used to derive an initial estimator, if the selection is different from the default value
mstop.init	an integer giving the number of boosting iterations when type.init="bst"
nu.init	a small number (between 0 and 1) defining the step size or shrinkage parameter when type.init="bst".
decreasing	only used if lambda=NULL, a logical value used to determine regularization path direction either from lambda_max to a potentially modified lambda_min or vice versa if type.init="bst", "heu". Since this is a nonconvex optimization, it is possible to generate different estimates for the same lambda depending on decreasing. The choice of decreasing picks different starting values.
iter	number of iteration in the IRCO algorithm
maxit	Within each IRCO algorithm iteration, maximum number of coordinate descent iterations for each lambda value; default is 1000.
reltol	convergency criteria in the IRCO algorithm
eps	If a coefficient is less than eps in magnitude, then it is reported to be 0
epscycle	If nlambda > 1 and the relative loss values from two consecutive lambda values change > epscycle, then re-estimate parameters in an effort to avoid trap of local optimization.
thresh	Convergence threshold for coordinate descent. Defaults value is 1e-6.
penalty	Type of regularization
theta	an overdispersion scaling parameter for family="negbin"

`parallel`, `n.cores`
: If `TRUE`, to compute solution of `lambda` with parallel computing in number of `n.cores`. If `FALSE`, sequential computing. If `NULL`, still sequential computing with a different convergence criteria based on penalized loss values

`trace`, `tracelevel`
: If `TRUE`, fitting progress is reported. If `tracelevel=2`, deeper level of fitting progress is reported.

Details

A case weighted penalized least squares or GLM is fit by the iteratively reweighted convex optimization (IRCO), where the loss function is a composite function `cfunodfun` + penalty. Here convex is the loss function induced by `dfun`, not the penalty function. The sequence of robust models implied by `lambda` is fit by IRCO along with coordinate descent. Note that the objective function is

$$weights * loss + \lambda * penalty,$$

if `standardize=FALSE` and

$$\frac{weights}{\sum(weights)} * loss + \lambda * penalty,$$

if `standardize=TRUE`.

Value

An object with S3 class `"irglmreg"` for the various types of models.

`call`	the call that produced the model fit
`b0`	Intercept sequence of length `length(lambda)`
`beta`	A `nvars` x `length(lambda)` matrix of coefficients.
`lambda`	The actual sequence of `lambda` values used
`weights_update`	A `nobs` x `length(lambda)` matrix of weights computed by the IRCO algorithm. The entry of i-th row and j-th column is the weight for the i-th observation and j-th `lambda` value.
`decreasing`	if `lambda` is an increasing sequence or not, used to determine regularization path direction either from lambda_max to a potentially modified lambda_min or vice versa if `type.init="bst", "heu"`.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

`irglmreg`

irsvm *fit case weighted support vector machines with robust loss functions*

Description

Fit case weighted support vector machines with robust loss functions. This is the wrapper function of `irsvm_fit`, which does the computing.

Usage

```
## S3 method for class 'formula'
irsvm(formula, data, weights, contrasts=NULL, ...)
## S3 method for class 'matrix'
irsvm(x, y, weights, ...)
## Default S3 method:
irsvm(x,  ...)
```

Arguments

`formula`	symbolic description of the model, see details.
`data`	argument controlling formula processing via `model.frame`.
`weights`	optional numeric vector of weights
`x`	input matrix, of dimension nobs x nvars; each row is an observation vector
`y`	response variable. Quantitative for `type="eps-regression"`, `"nu-regression"` and -1/1 for `type="C-classification"`, `"nu-Classification"`.
`contrasts`	the contrasts corresponding to `levels` from the respective models
`...`	Other arguments passing to `irsvm_fit`

Details

Fit a robust SVM where the loss function is a composite function `cfunotype` + penalty. The model is fit by the iteratively reweighted SVM, an application of the iteratively reweighted convex optimization (IRCO). Here convex is the loss function induced by `type`.

For linear kernel, the coefficients of the regression/decision hyperplane can be extracted using the `coef` method.

Value

An object with S3 class `"wsvm"` for various types of models.

`call`	the call that produced this object
`weights_update`	weights in the final iteration of the IRCO algorithm
`cfun, s`	original input arguments
`delta`	delta value used for `cfun="gcave"`

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irsvm_fit, print, predict, coef.

Examples

```
#binomial
x=matrix(rnorm(100*20),100,20)
g2=sample(c(-1,1),100,replace=TRUE)
fit=irsvm(x,g2,s=1,cfun="ccave",type="C-classification")
```

---

irsvm_fit — *Fit iteratively reweighted support vector machines for robust loss functions*

Description

irsvm_fit is used to train a subject weighted support vector machine where the weights are provided iteratively from robust loss function with the iteratively reweighted convex optimization (IRCO). It can be used to carry out robust regression and binary classification. This does computing for the wrapper function irsvm.

Usage

```
irsvm_fit(x, y, weights, cfun="ccave", s=NULL, delta=0.0001, type = NULL,
          kernel="radial", cost=1, epsilon = 0.1, iter=10, reltol=1e-5,
          trace=FALSE, ...)
```

Arguments

x	a data matrix, a vector, or a sparse '*design* matrix' (object of class Matrix provided by the **Matrix** package, or of class matrix.csr provided by the **SparseM** package, or of class simple_triplet_matrix provided by the **slam** package).
y	a response vector with one label for each row/component of x. Can be either a factor (for classification tasks) or a numeric vector (for regression).
weights	the weight of each subject. It should be in the same length of y.
cfun	character, type of convex cap (concave) function. Valid options are:

- "hcave"
- "acave"
- "bcave"
- "ccave"
- "dcave"
- "ecave"
- "gcave"
- "tcave"

s — tuning parameter of cfun. s > 0 and can be equal to 0 for cfun="tcave". If s is too close to 0 for cfun="acave", "bcave", "ccave", the calculated weights can become 0 for all observations, thus crash the program.

delta — a small positive number provided by user only if cfun="gcave" and 0 < s <1

type — irsvm_fit can be used as a classification machine, or as a regression machine. Depending of whether y is a factor or not, the default setting for type is C-classification or eps-regression, respectively, but may be overwritten by setting an explicit value.
Valid options are:

- C-classification
- nu-classification
- eps-regression
- nu-regression

kernel — the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type.

linear: $u'v$
polynomial: $(\gamma u'v + coef0)^{degree}$
radial basis: $e^(- \gamma|u - v|^2)$
sigmoid: $tanh(\gamma u'v + coef0)$

cost — cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation. This is proportional to the inverse of lambda in irglmreg.

epsilon — epsilon in the insensitive-loss function (default: 0.1)

iter — number of iteration in the IRCO algorithm

reltol — convergency criteria in the IRCO algorithm

trace — If TRUE, fitting progress is reported

... — additional parameters for function wsvm in package **WeightSVM**

Details

A case weighted SVM is fit by the IRCO algorithm, where the loss function is a composite function of cfunotype, plus a L_2 penalty. Additional arguments include degree, gamma, coef0, class.weights, cachesize, tolerance, shrinking, propbability, fitted, the same as "wsvm" in package **WeightSVM**.

Value

An object of class "wsvm" (see package **WeightSVM**) containing the fitted model, including:

SV	The resulting support vectors (possibly scaled).
index	The index of the resulting support vectors in the data matrix. Note that this index refers to the preprocessed data (after the possible effect of na.omit and subset)
coefs	The corresponding coefficients times the training labels.
rho	The negative intercept.
sigma	In case of a probabilistic regression model, the scale parameter of the hypothesized (zero-mean) laplace distribution estimated by maximum likelihood.
probA, probB	numeric vectors of length 2, number of classes, containing the parameters of the logistic distributions fitted to the decision values of the binary classifiers (1 / (1 + exp(a x + b))).

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irsvm, print, predict, coef and plot.

Examples

```
data(iris)
 iris <- subset(iris, Species %in% c("setosa", "versicolor"))
 # default with factor response:
  model <- irsvm(Species ~ ., data = iris, kernel="linear", trace=TRUE)
  model <- irsvm(Species ~ ., data = iris)
 # alternatively the traditional interface:
  x <- subset(iris, select = -Species)
  y <- iris$Species
model <- irsvm(x, y)
  # test with train data
  pred <- predict(model, x)
  # (same as:)
  pred <- fitted(model)

  # Check accuracy:
  table(pred, y)
 # compute decision values and probabilities:
  pred <- predict(model, x, decision.values = TRUE)
  attr(pred, "decision.values")
```

```
# visualize (classes by color, SV by crosses):
plot(cmdscale(dist(iris[,-5])),
     col = as.integer(iris[,5]),
     pch = c("o","+")[1:100 %in% model$index + 1])

## try regression mode on two dimensions

# create data
x <- seq(0.1, 5, by = 0.05)
y <- log(x) + rnorm(x, sd = 0.2)

# estimate model and predict input values
m   <- irsvm(x, y)
new <- predict(m, x)

# visualize
plot(x, y)
 points(x, log(x), col = 2)
 points(x, new, col = 4)
```

`loss2` *Composite Loss Value*

Description

Compute composite loss value

Usage

```
loss2(y, f, weights, cfun, dfun, s, delta=0.0001)
```

Arguments

`y`	response variable values
`f`	linear predictor values of `y`. If `f` is predicted `response` of model, use function `loss3` instead
`weights`	observation weights, same length as `y`
`cfun`	integer from 1-8, concave function as in `irglm_fit`
`dfun`	integer from 1-7, convex function as in `irglm_fit`
`s`	tuning parameter of `cfun`. `s > 0` and can be equal to 0 for `cfun="tcave"`.
`delta`	a small positive number provided by user only if `cfun="gcave"` and `0 < s <1`

Details

An internal function. For large `s` values, the loss can be 0 with `cfun=2,3,4`, or "acave", "bcave", "ccave".

Value

Weighted loss values

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

loss3 irglm irglmreg loss2_irsvm

`loss2_irsvm` *Composite Loss Value for epsilon-insensitive Type*

Description

Compute composite loss value for epsilon-insensitive type function

Usage

```
loss2_irsvm(y, f, weights, cfun, dfun, s, eps, delta=0.0001)
```

Arguments

`y`	response variable values
`f`	fitted values of `y`
`weights`	observation weights, same length as `y`
`cfun`	integer from 1-8, concave function as in `irsvm_fit`
`dfun`	integer value, only `dfun=2` is implemented for now. Convex function as in `irsvm_fit`
`s`	tuning parameter of `cfun`. `s > 0` and can be equal to 0 for `cfun="tcave"`.
`delta`	a small positive number provided by user only if `cfun="gcave"` and `0 < s <1`
`eps`	non-negative parameter for epsilon-insensitive loss

Details

For large `s` values, the loss can be 0 with `cfun=2,3,4`, or "acave", "bcave", "ccave".

Value

Weighted loss values

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

irglmreg, loss2

---

loss3 *Composite Loss Value for GLM*

Description

Compute composite loss value

Usage

```
loss3(y, mu, theta, weights, cfun, family, s, delta)
```

Arguments

y	response variable values, 0/1 if family=2, or binomial
mu	response prediction of y. If mu is linear predictor, use function loss2 instead
theta	scale parameter for family=4, negative binomial
weights	observation weights, same length as y
cfun	integer from 1-8, concave function as in irglm_fit
family	integer 2, 3 or 4, convex function binomial, Poisson or negative binomial, respectively
s	tuning parameter of cfun. s > 0 and can be equal to 0 for cfun="tcave".
delta	a small positive number provided by user only if cfun="gcave" and 0 < s <1

Details

For large s values, the loss can be 0 with cfun=2,3,4, or "acave", "bcave", "ccave".

Value

Weighted loss values

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

loss2 irglm irglmreg loss2_irsvm

---

meatReg *Meat Matrix Estimator*

Description

Estimating the variance of the first derivative of log-likelihood function

Usage

```
meatReg(x, which, ...)
```

Arguments

x	a fitted model object. Currently only implemented for zipath object with family="negbin"
which	which penalty parameter(s)?
...	arguments passed to the estfunReg function.

Details

See reference below

Value

A

$$k \times k$$

covariance matrix of first derivative of log-likelihood function

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

sandwichReg, breadReg, estfunReg

Examples

```
data("bioChemists", package = "pscl")
fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10, maxit.em=1)
meatReg(fm_zinb, which=which.min(fm_zinb$bic))
```

---

methods *Methods for mpath Objects*

Description

Methods for models fitted by coordinate descent algorithms.

Usage

```
## S3 method for class 'glmreg'
AIC(object, ..., k)
## S3 method for class 'zipath'
AIC(object, ..., k)
## S3 method for class 'glmreg'
BIC(object, ...)
## S3 method for class 'zipath'
BIC(object, ...)
```

Arguments

object	objects of class glmreg or zipath.
...	additional arguments passed to calls.
k	numeric, the *penalty* per parameter to be used; the default k = 2 is the classical AIC. k has been hard coded in the function and there is no impact to the value of AIC if k is changed

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

`ncl` *fit a nonconvex loss based robust linear model*

Description

Fit a linear model via penalized nonconvex loss function.

Usage

```
## S3 method for class 'formula'
ncl(formula, data, weights, offset=NULL, contrasts=NULL,
x.keep=FALSE, y.keep=TRUE, ...)
## S3 method for class 'matrix'
ncl(x, y, weights, offset=NULL, ...)
## Default S3 method:
ncl(x,  ...)
```

Arguments

`formula`	symbolic description of the model, see details.
`data`	argument controlling formula processing via `model.frame`.
`weights`	optional numeric vector of weights. If `standardize=TRUE`, weights are renormalized to weights/sum(weights). If `standardize=FALSE`, weights are kept as original input
`x`	input matrix, of dimension nobs x nvars; each row is an observation vector
`y`	response variable. Quantitative for `rfamily="clossR"` and -1/1 for classification.
`offset`	Not implemented yet
`contrasts`	the contrasts corresponding to `levels` from the respective models
`x.keep, y.keep`	For glmreg: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For ncl_fit: x is a design matrix of dimension n * p, and x is a vector of observations of length n.
`...`	Other arguments passing to `ncl_fit`

Details

The robust linear model is fit by majorization-minimization along with linear regression. Note that the objective function is

$$weights * loss$$

.

Value

An object with S3 class "ncl" for the various types of models.

call	the call that produced this object
fitted.values	predicted values
h	pseudo response values in the MM algorithm

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2021), *MM for Penalized Estimation*, *TEST*, doi:10.1007/s11749021007702

See Also

print, predict, coef.

Examples

```
#binomial
x=matrix(rnorm(100*20),100,20)
g2=sample(c(-1,1),100,replace=TRUE)
fit=ncl(x,g2,s=1,rfamily="closs")
```

---

ncl_fit *Internal function to fit a nonconvex loss based robust linear model*

Description

Fit a linear model via penalized nonconvex loss function.

Usage

```
ncl_fit(x,y, weights, offset=NULL,
        rfamily=c("clossR", "closs", "gloss", "qloss"),
        s=NULL, fk=NULL, iter=10, reltol=1e-5, trace=FALSE)
```

Arguments

x	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable. Quantitative for rfamily="clossR" and -1/1 for classifications.
weights	observation weights. Can be total counts if responses are proportion matrices. Default is 1 for each observation
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
rfamily	Response type and relevant loss functions (see above)
s	nonconvex loss tuning parameter for robust regression and classification.
fk	predicted values at an iteration in the MM algorithm
iter	number of iteration in the MM algorithm
reltol	convergency criteria
trace	If TRUE, fitting progress is reported

Details

The robust linear model is fit by majorization-minimization along with least squares. Note that the objective function is

$$weights * loss$$

.

Value

An object with S3 class "ncl" for the various types of models.

call	the call that produced the model fit
fitted.values	predicted values
h	pseudo response values in the MM algorithm

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2021), *MM for Penalized Estimation*, *TEST*, doi:10.1007/s11749021007702

See Also

ncl

nclreg *Optimize a nonconvex loss with regularization*

Description

Fit a linear model via penalized nonconvex loss function. The regularization path is computed for the lasso (or elastic net penalty), scad (or snet) and mcp (or mnet penalty), at a grid of values for the regularization parameter lambda. The name refers to **N**on**C**onvex **L**oss with **REG**ularization.

Usage

```
## S3 method for class 'formula'
nclreg(formula, data, weights, offset=NULL, contrasts=NULL, ...)
## S3 method for class 'matrix'
nclreg(x, y, weights, offset=NULL, ...)
## Default S3 method:
nclreg(x,  ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
weights	optional numeric vector of weights. If standardize=TRUE, weights are renormalized to weights/sum(weights). If standardize=FALSE, weights are kept as original input
x	input matrix, of dimension nobs x nvars; each row is an observation vector
y	response variable. Quantitative for rfamily="clossR" and -1/1 for classification.
offset	Not implemented yet
contrasts	the contrasts corresponding to levels from the respective models
...	Other arguments passing to nclreg_fit

Details

The sequence of robust models implied by lambda is fit by majorization-minimization along with coordinate descent. Note that the objective function is

$$weights * loss + \lambda * penalty,$$

if standardize=FALSE and

$$\frac{weights}{\sum(weights)} * loss + \lambda * penalty,$$

if standardize=TRUE.

Value

An object with S3 class "nclreg" for the various types of models.

call	the call that produced this object
b0	Intercept sequence of length length(lambda)
beta	A nvars x length(lambda) matrix of coefficients.
lambda	The actual sequence of lambda values used
nobs	number of observations
risk	if type.path="nonactive", a matrix with number of rows iter and number of columns nlambda, loss values along the regularization path. If type.path="fast", a vector of length nlambda, loss values along the regularization path
pll	if type.path="nonactive", a matrix with number of rows iter and number of columns nlambda, penalized loss values along the regularization path. If type.path="fast", a vector of length nlambda, penalized loss values along the regularization path
fitted.values	predicted values depending on standardize, internal use only

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2021), *MM for Penalized Estimation*, *TEST*, doi:10.1007/s11749021007702

See Also

print, predict, coef and plot methods, and the cv.nclreg function.

Examples

```
#binomial
x=matrix(rnorm(100*20),100,20)
g2=sample(c(-1,1),100,replace=TRUE)
### different solution paths via a combination of type.path, decreasing and type.init
fit1=nclreg(x,g2,s=1,rfamily="closs",type.path="active",decreasing=TRUE,type.init="bst")
fit2=nclreg(x,g2,s=1,rfamily="closs",type.path="active",decreasing=FALSE,type.init="bst")
fit3=nclreg(x,g2,s=1,rfamily="closs",type.path="nonactive",decreasing=TRUE,type.init="bst")
fit4=nclreg(x,g2,s=1,rfamily="closs",type.path="nonactive",decreasing=FALSE,type.init="bst")
fit5=nclreg(x,g2,s=1,rfamily="closs",type.path="active",decreasing=TRUE,type.init="ncl")
fit6=nclreg(x,g2,s=1,rfamily="closs",type.path="active",decreasing=FALSE,type.init="ncl")
fit7=nclreg(x,g2,s=1,rfamily="closs",type.path="nonactive",decreasing=TRUE,type.init="ncl")
fit8=nclreg(x,g2,s=1,rfamily="closs",type.path="nonactive",decreasing=FALSE,type.init="ncl")
```

nclreg_fit *Internal function to fitting a nonconvex loss based robust linear model with regularization*

Description

Fit a linear model via penalized nonconvex loss function. The regularization path is computed for the lasso (or elastic net penalty), scad (or snet) and mcp (or mnet penalty), at a grid of values for the regularization parameter lambda.

Usage

```
nclreg_fit(x, y, weights, offset, rfamily=c("clossR", "closs", "gloss", "qloss"),
           s=NULL, fk=NULL, iter=10, reltol=1e-5,
           penalty=c("enet","mnet","snet"), nlambda=100,lambda=NULL,
           type.path=c("active", "nonactive", "onestep"), decreasing=FALSE,
           lambda.min.ratio=ifelse(nobs<nvars,.05, .001), alpha=1, gamma=3,
           standardize=TRUE, intercept=TRUE, penalty.factor=NULL, maxit=1000,
           type.init=c("bst", "ncl", "heu"), mstop.init=10, nu.init=0.1,
           eps=.Machine$double.eps, epscycle=10, thresh=1e-6, trace=FALSE)
```

Arguments

x	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable. Quantitative for rfamily="clossR" and -1/1 for classifications.
weights	observation weights. Can be total counts if responses are proportion matrices. Default is 1 for each observation
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. Currently only one offset term can be included in the formula.
rfamily	Response type and relevant loss functions (see above)
s	nonconvex loss tuning parameter for robust regression and classification. The s value is for robust nonconvex loss where smaller s value is more robust to outliers with rfamily="closs", and larger s value more robust with rfamily="clossR", "gloss", "qloss".
fk	predicted values at an iteration in the MM algorithm
nlambda	The number of lambda values - default is 100. The sequence may be truncated before nlambda is reached if a close to saturated model is fitted. See also satu.
lambda	by default, the algorithm provides a sequence of regularization values, or a user supplied lambda sequence

`type.path` solution path. If `type.path="active"`, then cycle through only the active set in the next increasing `lambda` sequence. If `type.path="nonactive"`, no active set for each element of the lambda sequence and cycle through all the predictor variables. If `type.path="onestep"`, update for one element of lambda depending on `decreasing=FALSE` (last element of lambda) or `decreasing=TRUE` (then first element of lambda) in each MM iteration, and iterate until convergency of prediction. Then fit a solution path based on the sequence of lambda.

`lambda.min.ratio` Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero except the intercept). Note, there is no closed formula for `lambda.max`. The default of `lambda.min.ratio` depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs > nvars`, the default is `0.001`, close to zero. If `nobs < nvars`, the default is `0.05`.

`alpha` The L_2 penalty mixing parameter, with $0 \leq alpha \leq 1$. `alpha=1` is lasso (mcp, scad) penalty; and `alpha=0` the ridge penalty. However, if `alpha=0`, one must provide `lambda` values.

`gamma` The tuning parameter of the `snet` or `mnet` penalty.

`standardize` logical value for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is `TRUE`.

`intercept` logical value: if TRUE (default), intercept(s) are fitted; otherwise, intercept(s) are set to zero

`penalty.factor` This is a number that multiplies `lambda` to allow differential shrinkage of coefficients. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is same shrinkage for all variables.

`type.init` a method to determine the initial values. If `type.init="ncl"`, an intercept-only model as initial parameter and run `nclreg` regularization path forward from lambda_max to lambda_min. If `type.init="heu"`, heuristic initial parameters and run `nclreg` path backward or forward depending on `decreasing`, between lambda_min and lambda_max. If `type.init="bst"`, run a boosting model with `bst` in package bst, depending on `mstop.init`, `nu.init` and run `nclreg` backward or forward depending on `decreasing`.

`mstop.init` an integer giving the number of boosting iterations when `type.init="bst"`

`nu.init` a small number (between 0 and 1) defining the step size or shrinkage parameter when `type.init="bst"`.

`decreasing` only used if `lambda=NULL`, a logical value used to determine regularization path direction either from lambda_max to a potentially modified lambda_min or vice versa if `type.init="bst","heu"`. Since this is a nonconvex optimization, it is possible to generate different estimates for the same `lambda` depending on `decreasing`. The choice of `decreasing` picks different starting values.

`iter` number of iteration in the MM algorithm

`maxit` Within each MM algorithm iteration, maximum number of coordinate descent iterations for each `lambda` value; default is 1000.

`reltol` convergency criteria

eps	If a coefficient is less than eps in magnitude, then it is reported to be 0
epscycle	If nlambda > 1 and the relative loss values from two consecutive lambda values change > epscycle, then re-estimate parameters in an effort to avoid trap of local optimization.
thresh	Convergence threshold for coordinate descent. Defaults value is 1e-6.
penalty	Type of regularization
trace	If TRUE, fitting progress is reported

Details

The sequence of robust models implied by lambda is fit by majorization-minimization along with coordinate descent. Note that the objective function is

$$weights * loss + \lambda * penalty,$$

if standardize=FALSE and

$$\frac{weights}{\sum(weights)} * loss + \lambda * penalty,$$

if standardize=TRUE.

Value

An object with S3 class "nclreg" for the various types of models.

call	the call that produced the model fit
b0	Intercept sequence of length length(lambda)
beta	A nvars x length(lambda) matrix of coefficients.
lambda	The actual sequence of lambda values used
decreasing	if lambda is an increasing sequence or not, used to determine regularization path direction either from lambda_max to a potentially modified lambda_min or vice versa if type.init="bst", "heu".

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2021), *MM for Penalized Estimation*, *TEST*, doi:10.1007/s11749021007702

See Also

nclreg

plot.glmreg *plot coefficients from a "glmreg" object*

Description

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

Usage

```
## S3 method for class 'glmreg'
plot(x, xvar = c("norm", "lambda", "dev"), label = FALSE, shade=TRUE, ...)
```

Arguments

x	fitted "glmreg" model
xvar	What is on the X-axis. "norm" plots against the L1-norm of the coefficients, "lambda" against the log-lambda sequence, and "dev" against the percent deviance explained.
label	If TRUE, label the curves with variable sequence numbers.
shade	Should nonconvex region be shaded? Default is TRUE. Code developed for all weights=1 only
...	Other graphical parameters to plot

Details

A coefficient profile plot is produced.

Author(s)

Zhu Wang zwang145@uthsc.edu

See Also

glmreg, and print, predict and coef methods.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=glmreg(x,y)
plot(fit1)
plot(fit1,xvar="lambda",label=TRUE)
```

predict.glmreg *Model predictions based on a fitted "glmreg" object.*

Description

This function returns predictions from a fitted "glmreg" object.

Usage

```
## S3 method for class 'glmreg'
predict(object,newx,newoffset,which=1:length(object$lambda),
type=c("link","response","class","coefficients","nonzero"), na.action=na.pass, ...)
## S3 method for class 'glmreg'
coef(object,which=1:length(object$lambda),...)
```

Arguments

object	Fitted "glmreg" model object.
newx	Matrix of values at which predictions are to be made. Not used for type="coefficients"
which	Indices of the penalty parameter lambda at which predictions are required. By default, all indices are returned.
type	Type of prediction: "link" returns the linear predictors; "response" gives the fitted values; "class" returns the binomial outcome with the highest probability; "coefficients" returns the coefficients.
newoffset	an offset term used in prediction
na.action	action for missing data value
...	arguments for predict

Value

The returned object depends on type.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

glmreg

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
fit <- glmreg(counts ~ outcome + treatment, data=d.AD, family="poisson")
predict(fit, newx=d.AD[,1:2])
summary(fit)
coef(fit)
```

`predict.zipath` *Methods for zipath Objects*

Description

Methods for extracting information from fitted penalized zero-inflated regression model objects of class `"zipath"`.

Usage

```
## S3 method for class 'zipath'
predict(object, newdata,  which = 1:object$nlambda,
  type = c("response", "prob", "count", "zero", "nonzero"), na.action = na.pass,
  at = NULL, ...)
## S3 method for class 'zipath'
residuals(object, type = c("pearson", "response"), ...)

## S3 method for class 'zipath'
coef(object, which=1:object$nlambda, model = c("full", "count", "zero"), ...)

## S3 method for class 'zipath'
terms(x, model = c("count", "zero"), ...)
## S3 method for class 'zipath'
model.matrix(object, model = c("count", "zero"), ...)
```

Arguments

`object`, `x`	an object of class `"zipath"` as returned by `zipath`.
`newdata`	optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
`which`	Indices of the penalty parameters `lambda` at which predictions are required. By default, all indices are returned.
`type`	character specifying the type of predictions or residuals, respectively. For details see below.

na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
at	optionally, if type = "prob", a numeric vector at which the probabilities are evaluated. By default 0:max(y) is used where y is the original observed response.
model	character specifying for which component of the model the terms or model matrix should be extracted.
...	currently not used.

Details

Re-uses the design of function zeroinfl in package pscl (see reference). A set of standard extractor functions for fitted model objects is available for objects of class "zipath", including methods to the generic functions print and summary which print the estimated coefficients along with some further information. As usual, the summary method returns an object of class "summary.zipath" containing the relevant summary statistics which can subsequently be printed using the associated print method.

The methods for coef by default return a single vector of coefficients and their associated covariance matrix, respectively, i.e., all coefficients are concatenated. By setting the model argument, the estimates for the corresponding model components can be extracted.

Both the fitted and predict methods can compute fitted responses. The latter additionally provides the predicted density (i.e., probabilities for the observed counts), the predicted mean from the count component (without zero inflation) and the predicted probability for the zero component. The residuals method can compute raw residuals (observed - fitted) and Pearson residuals (raw residuals scaled by square root of variance function).

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath

Examples

```
## Not run:
data("bioChemists", package = "pscl")
fm_zip <- zipath(art ~ . | ., data = bioChemists, nlambda=10)
plot(residuals(fm_zip) ~ fitted(fm_zip))
coef(fm_zip, model = "count")
coef(fm_zip, model = "zero")
summary(fm_zip)
logLik(fm_zip)

## End(Not run)
```

`pval.zipath` *compute p-values from penalized zero-inflated model with multi-split data*

Description

compute p-values from penalized zero-inflated Poisson, negative binomial and geometric model with multi-split data

Usage

```
pval.zipath(formula, data, weights, subset, na.action, offset, standardize=TRUE,
            family = c("poisson", "negbin", "geometric"),
            penalty = c("enet", "mnet", "snet"), gamma.count = 3,
            gamma.zero = 3, prop=0.5, trace=TRUE, B=10, ...)
```

Arguments

`formula`	symbolic description of the model, see details.
`data`	argument controlling formula processing via `model.frame`.
`weights`	optional numeric vector of weights. If `standardize=TRUE`, weights are renormalized to weights/sum(weights). If `standardize=FALSE`, weights are kept as original input
`subset`	subset of data
`na.action`	how to deal with missing data
`offset`	Not implemented yet
`standardize`	logical value, should variables be standardized?
`family`	family to fit `zipath`
`penalty`	penalty considered as one of `enet`, `mnet`, `snet`.
`gamma.count`	The tuning parameter of the `snet` or `mnet` penalty for the count part of model.
`gamma.zero`	The tuning parameter of the `snet` or `mnet` penalty for the zero part of model.
`prop`	proportion of data split, default is 50/50 split
`trace`	logical value, if TRUE, print detailed calculation results
`B`	number of repeated multi-split replications
`...`	Other arguments passing to `glmreg_fit`

Details

compute p-values from penalized zero-inflated Poisson, negative binomial and geometric model with multi-split data

Value

`count.pval`	raw p-values in the count component
`zero.pval`	raw p-values in the zero component
`count.pval.q`	Q value for the count component
`zero.pval.q`	Q value for the zero component

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Nicolai Meinshausen, Lukas Meier and Peter Buehlmann (2013) *p-Values for High-Dimensional Regression*, *Journal of the American Statistical Association*, 104(488), 1671–1681.

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

---

rzi *random number generation of zero-inflated count response*

Description

random number generation of zero-inflated count response

Usage

```
rzi(n, x, z, a, b, theta=1, family=c("poisson", "negbin", "geometric"), infl=TRUE)
```

Arguments

`n`	sample size of random number generation
`x`	design matrix of count model
`z`	design matrix of zero model
`a`	coefficient vector for `x`, length must be the same as column size of `x`
`b`	coefficient vector for `z`, length must be the same as column size of `z`
`theta`	dispersion parameter for `family="negbin"`
`family`	distribution of count model
`infl`	logical value, if TRUE, zero-inflated count response

Details

random number generation of zero-inflated count response

Value

numeric vector of zero-inflated count response

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

`sandwichReg` *Making Sandwiches with Bread and Meat for Regularized Estimators*

Description

Constructing sandwich covariance matrix estimators by multiplying bread and meat matrices for regularized regression parameters.

Usage

```
sandwichReg(x, breadreg.=breadReg, meatreg.=meatReg, which, log=FALSE, ...)
```

Arguments

`x`	a fitted model object.
`breadreg.`	either a breadReg matrix or a function for computing this via `breadreg.(x)`.
`meatreg.`	either a breadReg matrix or a function for computing this via `meatreg.(x, ...)`.
`which`	which penalty parameters(s) to compute?
`log`	if TRUE, the corresponding element is with respect to log(theta) in negative binomial regression. Otherwise, for theta
`...`	arguments passed to the `meatReg` function.

Details

sandwichReg is a function to compute an estimator for the covariance of the non-zero parameters. It takes a breadReg matrix (i.e., estimator of the expectation of the negative derivative of the penalized estimating functions) and a meatReg matrix (i.e., estimator of the variance of the log-likelihood function) and multiplies them to a sandwich with meat between two slices of bread. By default breadReg and meatReg are called. Implemented only for zipath object with family="negbin" in the current version.

Value

A matrix containing the sandwich covariance matrix estimate for the non-zero parameters.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

breadReg, meatReg

Examples

```
data("bioChemists", package = "pscl")
fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10, maxit.em=1)
sandwichReg(fm_zinb, which=which.min(fm_zinb$bic))
```

---

se *Standard Error of Regularized Estimators*

Description

Generic function for computing standard errors of non-zero regularized estimators

Usage

```
se(x, which, log=TRUE, ...)
```

Arguments

x	a fitted model object.
which	which penalty parameter(s)?
log	if TRUE, the computed standard error is for log(theta) for negative binomial regression, otherwise, for theta.
...	arguments passed to methods.

Value

A vector containing standard errors of non-zero regularized estimators.

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath

Examples

```
data("bioChemists", package = "pscl")
fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10, maxit.em=1)
res <- se(fm_zinb, which=which.min(fm_zinb$bic))
```

---

stan *standardize variables*

Description

Standardize variables. For each column, return mean 0 and mean value of sum of squares = 1.

Usage

```
stan(x, weights)
```

Arguments

x	numeric variables, can be a matrix or vector
weights	numeric positive vector of weights

Value

A list with the following items.

x	standardized variables with each column: mean value 0 and mean value of sum of squares = 1.
meanx	a vector of means for each column in the original x
normx	a vector of scales for each column in the original x

Author(s)

Zhu Wang <zwang145@uthsc.edu>

---

summary.glmregNB *Summary Method Function for Objects of Class 'glmregNB'*

Description

Summary results of fitted penalized negative binomial regression model

Usage

```
## S3 method for class 'glmregNB'
summary(object, ...)
```

Arguments

object	fitted model object of class glmregNB.
...	arguments passed to or from other methods.

Details

This function is a method for the generic function summary() for class "glmregNB". It can be invoked by calling summary(x) for an object x of the appropriate class, or directly by calling summary.glmregNB(x) regardless of the class of the object.

Value

Summary of fitted penalized negative binomial model

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

See Also

summary, glm.nb

---

tuning.zipath *find optimal path for penalized zero-inflated model*

Description

Fit penalized zero-inflated models, generate multiple paths with varying penalty parameters, therefore determine optimal path with respect to a particular penalty parameter

Usage

```
tuning.zipath(formula, data, weights, subset, na.action, offset, standardize=TRUE,
              family = c("poisson", "negbin", "geometric"),
              penalty = c("enet", "mnet", "snet"), lambdaCountRatio = .0001,
              lambdaZeroRatio = c(.1, .01, .001), maxit.theta=1, gamma.count=3,
              gamma.zero=3, ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
weights	optional numeric vector of weights. If standardize=TRUE, weights are renormalized to weights/sum(weights). If standardize=FALSE, weights are kept as original input
subset	subset of data
na.action	how to deal with missing data
offset	Not implemented yet
standardize	logical value, should variables be standardized?
family	family to fit
penalty	penalty considered as one of enet, mnet, snet.
lambdaCountRatio, lambdaZeroRatio	Smallest value for lambda.count and lambda.zero, respectively, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero except the intercepts). This lambda.max can be a surrogate value for penalty="mnet" or "snet"
maxit.theta	For family="negbin", the maximum iteration allowed for estimating scale parameter theta. Note, the default value 1 is for computing speed purposes, and is typically too small and less desirable in real data analysis
gamma.count	The tuning parameter of the snet or mnet penalty for the count part of model.
gamma.zero	The tuning parameter of the snet or mnet penalty for the zero part of model.
...	Other arguments passing to zipath

Details

From the default `lambdaZeroRatio = c(.1, .01, .001)` values, find optimal lambdaZeroRatio for penalized zero-inflated Poisson, negative binomial and geometric model

Value

An object of class zipath with the optimal lambdaZeroRatio

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath

Examples

```
## Not run:
## data
data("bioChemists", package = "pscl")

## inflation with regressors
## ("art ~ . | ." is "art ~ fem + mar + kid5 + phd + ment | fem + mar + kid5 + phd + ment")
fm_zip2 <- tuning.zipath(art ~ . | ., data = bioChemists, nlambda=10)
summary(fm_zip2)
fm_zinb2 <- tuning.zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)
summary(fm_zinb2)

## End(Not run)
```

update_wt *Compute weight value*

Description

Compute weight value

Usage

```
update_wt(y, ypre, weights, cfun, s, dfun, delta=0.0001)
```

Arguments

y	input value of response variable
ypre	predicted value of response variable
weights	optional numeric vector of weights.
cfun	integer from 1-8, concave function as in irglm_fit
dfun	integer value, convex function as in irglm_fit
s	a numeric value, see details in irglm_fit
delta	a positive small value, see details in irglm_fit

Value

Weight value

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang (2024) *Unified Robust Estimation*, *Australian & New Zealand Journal of Statistics*. 66(1):77-102.

See Also

compute_wt

zipath *Fit zero-inflated count data linear model with lasso (or elastic net), snet or mnet regularization*

Description

Fit zero-inflated regression models for count data via penalized maximum likelihood.

Usage

```
## S3 method for class 'formula'
zipath(formula, data, weights, offset=NULL, contrasts=NULL, ... )
## S3 method for class 'matrix'
zipath(X, Z, Y, weights, offsetx=NULL, offsetz=NULL, ...)
## Default S3 method:
zipath(X, ...)
```

Arguments

formula	symbolic description of the model, see details.
data	argument controlling formula processing via model.frame.
weights	optional numeric vector of weights.
offset	optional numeric vector with an a priori known component to be included in the linear predictor of the count model or zero model. See below for an example.
contrasts	a list with elements "count" and "zero" containing the contrasts corresponding to levels from the respective models
X	predictor matrix of the count model
Z	predictor matrix of the zero model
Y	response variable
offsetx, offsetz	optional numeric vector with an a priori known component to be included in the linear predictor of the count model (offsetx)or zero model (offsetz).
...	Other arguments which can be passed to glmreg or glmregNB

Value

An object of class "zipath", i.e., a list with components including

coefficients	a list with elements "count" and "zero" containing the coefficients from the respective models,
residuals	a vector of raw residuals (observed - fitted),
fitted.values	a vector of fitted means,
weights	the case weights used,

terms	a list with elements "count", "zero" and "full" containing the terms objects for the respective models,
theta	estimate of the additional θ parameter of the negative binomial model (if a negative binomial regression is used),
loglik	log-likelihood of the fitted model,
family	character string describing the count distribution used,
link	character string describing the link of the zero-inflation model,
linkinv	the inverse link function corresponding to link,
converged	logical value, TRUE indicating successful convergence of zipath, FALSE indicating otherwise
call	the original function call
formula	the original formula
levels	levels of the categorical regressors
contrasts	a list with elements "count" and "zero" containing the contrasts corresponding to levels from the respective models,
model	the full model frame (if model = TRUE),
y	the response count vector (if y = TRUE),
x	a list with elements "count" and "zero" containing the model matrices from the respective models (if x = TRUE),

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath_fit, glmreg, glmregNB

Examples

```
## data
data("bioChemists", package = "pscl")
## with simple inflation (no regressors for zero component)
fm_zip <- zipath(art ~ 1 | ., data = bioChemists, nlambda=10)
summary(fm_zip)
fm_zip <- zipath(art ~ . | 1, data = bioChemists, nlambda=10)
summary(fm_zip)
## Not run:
fm_zip <- zipath(art ~ . | 1, data = bioChemists, nlambda=10)
summary(fm_zip)
fm_zinb <- zipath(art ~ . | 1, data = bioChemists, family = "negbin", nlambda=10)
summary(fm_zinb)
## inflation with regressors
## ("art ~ . | ." is "art ~ fem + mar + kid5 + phd + ment | fem + mar + kid5 + phd + ment")
fm_zip2 <- zipath(art ~ . | ., data = bioChemists, nlambda=10)
summary(fm_zip2)
fm_zinb2 <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)
summary(fm_zinb2)
### non-penalized regression, compare with zeroinfl
fm_zinb3 <- zipath(art ~ . | ., data = bioChemists, family = "negbin",
lambda.count=0, lambda.zero=0, reltol=1e-12)
summary(fm_zinb3)
library("pscl")
fm_zinb4 <- zeroinfl(art ~ . | ., data = bioChemists, dist = "negbin")
summary(fm_zinb4)
### offset
exposure <- rep(0.5, dim(bioChemists)[1])
fm_zinb <- zipath(art ~ . +offset(log(exposure))| ., data = bioChemists,
    family = "poisson", nlambda=10)
coef <- coef(fm_zinb)
### offset can't be specified in predict function as it has been contained
pred <- predict(fm_zinb)
## without inflation
## ("art ~ ." is "art ~ fem + mar + kid5 + phd + ment")
fm_pois <- glmreg(art ~ ., data = bioChemists, family = "poisson")
coef <- coef(fm_pois)
fm_nb <- glmregNB(art ~ ., data = bioChemists)
coef <- coef(fm_nb)
### high-dimensional
#R CMD check --use-valgrind can be too time extensive for the following model
#bioChemists <- cbind(matrix(rnorm(915*100), nrow=915), bioChemists)
#fm_zinb <- zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)

## End(Not run)
```

`zipath_fit` *Internal function to fit zero-inflated count data linear model with lasso (or elastic net), snet or mnet regularization*

Description

Fit zero-inflated regression models for count data via penalized maximum likelihood.

Usage

```
zipath_fit(X, Z, Y, weights, offsetx, offsetz, standardize=TRUE,
           intercept = TRUE, family = c("poisson", "negbin", "geometric"),
           link = c("logit", "probit", "cloglog", "cauchit", "log"),
           penalty = c("enet", "mnet",  "snet"), start = NULL, y = TRUE,
           x = FALSE, nlambda=100, lambda.count=NULL, lambda.zero=NULL,
           type.path=c("active", "nonactive"), penalty.factor.count=NULL,
           penalty.factor.zero=NULL, lambda.count.min.ratio=.0001,
           lambda.zero.min.ratio=.1, alpha.count=1, alpha.zero=alpha.count,
           gamma.count=3, gamma.zero=gamma.count, rescale=FALSE,
           init.theta=NULL, theta.fixed=FALSE, EM=TRUE, maxit.em=200,
           convtype=c("count", "both"), maxit= 1000, maxit.theta =10,
           reltol = 1e-5, thresh=1e-6, eps.bino=1e-5, shortlist=FALSE,
           trace=FALSE, ...)
```

Arguments

X	predictor matrix of the count model
Z	predictor matrix of the zero model
Y	response variable
weights	optional numeric vector of weights.
offsetx	optional numeric vector with an a priori known component to be included in the linear predictor of the count model.
offsetz	optional numeric vector with an a priori known component to be included in the linear predictor of the zero model.
intercept	Should intercept(s) be fitted (default=TRUE) or set to zero (FALSE)
standardize	Logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is standardize=TRUE.
family	character specification of count model family (a log link is always used).
link	character specification of link function in the binary zero-inflation model (a binomial family is always used).
y, x	logicals. If TRUE the corresponding response and model matrix are returned.
penalty	penalty considered as one of enet, mnet, snet.
start	starting values for the parameters in the linear predictor.
nlambda	number of lambda value, default value is 100. The sequence may be truncated before nlambda is reached if a close to saturated model for the zero component is fitted.
lambda.count	A user supplied lambda.count sequence. Typical usage is to have the program compute its own lambda.count and lambda.zero sequence based on nlambda and lambda.min.ratio.

`lambda.zero` A user supplied `lambda.zero` sequence.

`type.path` solution path with default value `"active"`, which is less time computing than `"nonactive"`. If `type.path="nonactive"`, no active set for each element of the lambda sequence and cycle through all the predictor variables. If `type.path="active"`, then cycle through only the active set, then cycle through all the variables for the same penalty parameter. See details below.

`penalty.factor.count`, `penalty.factor.zero`
These are numeric vectors with the same length as predictor variables. that multiply `lambda.count, lambda.zero`, respectively, to allow differential shrinkage of coefficients. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is same shrinkage for all variables.

`lambda.count.min.ratio`, `lambda.zero.min.ratio`
Smallest value for `lambda.count` and `lambda.zero`, respectively, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero except the intercepts). Note, there is a closed formula for `lambda.max` for `penalty="enet"`. If `rescale=TRUE`, `lambda.max` is the same for `penalty="mnet"` or `"snet"`. Otherwise, some modifications are required. In the current implementation, for small `gamma` value, the square root of the computed `lambda.zero[1]` is used when `penalty="mnet"` or `"snet"`.

`alpha.count` The elastic net mixing parameter for the count part of model. The default value 1 implies no L_2 penalty, as in LASSO.

`alpha.zero` The elastic net mixing parameter for the zero part of model. The default value 1 implies no L_2 penalty, as in LASSO.

`gamma.count` The tuning parameter of the `snet` or `mnet` penalty for the count part of model.

`gamma.zero` The tuning parameter of the `snet` or `mnet` penalty for the zero part of model.

`rescale` logical value, if TRUE, adaptive rescaling

`init.theta` The initial value of `theta` for `family="negbin"`. This is set to `NULL` since version 0.3-24.

`theta.fixed` Logical value only used for `family="negbin"`. If `TRUE` and `init.theta` is provided with a numeric value > 0, then `init.theta` is not updated. If `theta.fixed=FALSE`, then `init.theta` will be updated. In this case, if `init.theta=NULL`, its initial value is computed with intercept-only zero-inflated negbin model.

`EM` Using `EM` algorithm. Not implemented otherwise

`convtype` convergency type, default is for count component only for speedy computation

`maxit.em` Maximum number of EM algorithm

`maxit` Maximum number of coordinate descent algorithm

`maxit.theta` Maximum number of iterations for estimating `theta` scaling parameter if family="negbin". Default value `maxit.theta` may be increased, yet may slow the algorithm

`eps.bino` a lower bound of probabilities to be claimed as zero, for computing weights and related values when `family="binomial"`.

`reltol` Convergence criteria, default value 1e-5 may be reduced to make more accurate yet slow

thresh	Convergence threshold for coordinate descent. Defaults value is 1e-6.
shortlist	logical value, if TRUE, limited results return
trace	If TRUE, progress of algorithm is reported
...	Other arguments which can be passed to glmreg or glmregNB

Details

The algorithm fits penalized zero-inflated count data regression models using the coordinate descent algorithm within the EM algorithm. The returned fitted model object is of class "zipath" and is similar to fitted "glm" and "zeroinfl" objects. For elements such as "coefficients" a list is returned with elements for the zero and count component, respectively.

If type.path="active", the algorithm iterates for a pair (lambda_count, lambda_zero) in a loop:
Step 1: For initial coefficients start_count of the count model and start_zero of the zero model, the EM algorithm is iterated until convergence for the active set with non-zero coefficients determined from start_count and start_zero, respectively.
Step 2: EM is iterated for all the predict variables once.
Step 3: If active set obtained from Step 2 is the same as in Step 1, stop; otherwise, repeat Step 1 and Step 2.
If type.path="nonactive", the EM algorithm iterates for a pair (lambda_count, lambda_zero) with all the predict variables until convergence.

A set of standard extractor functions for fitted model objects is available for objects of class "zipath", including methods to the generic functions print, coef, logLik, residuals, predict. See predict.zipath for more details on all methods.

The program may terminate with the following message:

```
Error in: while (j <= maxit.em && !converged) { :
Missing value, where TRUE/FALSE is necessary
Calls: zipath
Additionally: Warning:
In glmreg_fit(Znew, probi, weights = weights, standardize = standardize, :
saturated model, exiting ...
Execution halted
```

One possible reason is that the fitted model is too complex for the data. There are two suggestions to overcome the error. One is to reduce the number of variables. Second, find out what lambda values caused the problem and omit them. Try with other lambda values instead.

Value

An object of class "zipath", i.e., a list with components including

coefficients	a list with elements "count" and "zero" containing the coefficients from the respective models,
residuals	a vector of raw residuals (observed - fitted),
fitted.values	a vector of fitted means,
weights	the case weights used,

terms	a list with elements "count", "zero" and "full" containing the terms objects for the respective models,
theta	estimate of the additional θ parameter of the negative binomial model (if a negative binomial regression is used),
loglik	log-likelihood of the fitted model,
family	character string describing the count distribution used,
link	character string describing the link of the zero-inflation model,
linkinv	the inverse link function corresponding to link,
converged	logical value, TRUE indicating successful convergence of zipath, FALSE indicating otherwise
call	the original function call
formula	the original formula
levels	levels of the categorical regressors
model	the full model frame (if model = TRUE),
y	the response count vector (if y = TRUE),
x	a list with elements "count" and "zero" containing the model matrices from the respective models (if x = TRUE),

Author(s)

Zhu Wang <zwang145@uthsc.edu>

References

Zhu Wang, Shuangge Ma, Michael Zappitelli, Chirag Parikh, Ching-Yun Wang and Prasad Devarajan (2014) *Penalized Count Data Regression with Application to Hospital Stay after Pediatric Cardiac Surgery*, *Statistical Methods in Medical Research*. 2014 Apr 17. [Epub ahead of print]

Zhu Wang, Shuangge Ma, Ching-Yun Wang, Michael Zappitelli, Prasad Devarajan and Chirag R. Parikh (2014) *EM for Regularized Zero Inflated Regression Models with Applications to Postoperative Morbidity after Cardiac Surgery in Children*, *Statistics in Medicine*. 33(29):5192-208.

Zhu Wang, Shuangge Ma and Ching-Yun Wang (2015) *Variable selection for zero-inflated and overdispersed data with application to health care demand in Germany*, *Biometrical Journal*. 57(5):867-84.

See Also

zipath, glmreg, glmregNB

Index

∗ **classification**
compute_g, 5
compute_wt, 6
gfunc, 32
loss2, 54
loss2_irsvm, 55
loss3, 56
update_wt, 79
∗ **classif**
irsvm_fit, 51
∗ **datasets**
breastfeed, 5
docvisits, 30
∗ **methods**
methods, 58
∗ **models**
be.zeroinfl, 3
cv.glmreg, 8
cv.glmreg_fit, 11
cv.glmregNB, 13
cv.irglmreg, 15
cv.irglmreg_fit, 16
cv.irsvm, 18
cv.irsvm_fit, 20
cv.nclreg, 22
cv.nclreg_fit, 24
cv.zipath, 25
cv.zipath_fit, 28
glmreg, 32
glmreg_fit, 35
glmregNB, 38
irglm, 42
irglmreg, 44
irglmreg_fit, 46
irsvm, 50
ncl, 59
ncl_fit, 60
nclreg, 62
nclreg_fit, 64
plot.glmreg, 67
predict.glmreg, 68
pval.zipath, 71
rzi, 72
summary.glmregNB, 76
tuning.zipath, 77
∗ **neural**
irsvm_fit, 51
∗ **nonlinear**
irsvm_fit, 51
∗ **regression**
be.zeroinfl, 3
breadReg, 4
compute_g, 5
compute_wt, 6
cv.glmreg, 8
cv.glmreg_fit, 11
cv.glmregNB, 13
cv.irglmreg, 15
cv.irglmreg_fit, 16
cv.irsvm, 18
cv.irsvm_fit, 20
cv.nclreg, 22
cv.nclreg_fit, 24
cv.zipath, 25
cv.zipath_fit, 28
estfunReg, 31
gfunc, 32
glmreg, 32
glmreg_fit, 35
glmregNB, 38
hessianReg, 41
irglm, 42
irglmreg, 44
irglmreg_fit, 46
irsvm, 50
loss2, 54
loss2_irsvm, 55
loss3, 56

meatReg, 57
ncl, 59
ncl_fit, 60
nclreg, 62
nclreg_fit, 64
plot.glmreg, 67
predict.glmreg, 68
predict.zipath, 69
pval.zipath, 71
rzi, 72
sandwichReg, 73
se, 74
tuning.zipath, 77
update_wt, 79
zipath, 80
zipath_fit, 82

AIC.glmreg *(*methods*)*, 58
AIC.zipath *(*methods*)*, 58

be.zeroinfl, 3
BIC.glmreg *(*methods*)*, 58
BIC.zipath *(*methods*)*, 58
breadReg, 4, *42*, *58*, *74*
breastfeed, 5

coef, *4*, *10*, *12*, *14*, *16*, *18*, *23*, *25*, *27*, *29–31*, *34*, *40*, *44*, *45*, *51*, *53*, *60*, *63*, *70*, *85*
coef.cv.glmreg *(*cv.glmreg*)*, 8
coef.cv.irglmreg *(*cv.irglmreg*)*, 15
coef.cv.nclreg *(*cv.nclreg*)*, 22
coef.cv.zipath *(*cv.zipath*)*, 25
coef.glmreg *(*predict.glmreg*)*, 68
coef.irsvm *(*irsvm*)*, 50
coef.zipath *(*predict.zipath*)*, 69
compute_g, 5
compute_wt, 6, *79*
conv2glmreg, 7
conv2zipath, 8
cv.glmreg, 8, *34*
cv.glmreg_fit, 11
cv.glmregNB, 13, *40*
cv.irglmreg, 15, *45*
cv.irglmreg_fit, 16
cv.irsvm, 18, *22*
cv.irsvm_fit, 20
cv.nclreg, 22, *63*
cv.nclreg_fit, 24
cv.zipath, 25
cv.zipath_fit, 28

deviance.glmreg *(*glmreg*)*, 32
docvisits, 30

estfunReg, 31, *57*, *58*

fitted, *70*
fitted.zipath *(*predict.zipath*)*, 69

gfunc, 32
glm.nb, *77*
glmreg, *10*, *12*, 32, *38*, *68*, *81*, *86*
glmreg_fit, 35
glmregNB, *14*, 38, *81*, *86*
glmregNegbin *(*glmregNB*)*, 38

hessianReg, 41

irglm, 42, *55*, *57*
irglmreg, *6*, *7*, *16*, *18*, 44, *49*, *55–57*
irglmreg_fit, 46
irsvm, *19*, *22*, 50, *53*
irsvm_fit, 51, *51*

logLik, *85*
logLik.glmreg *(*glmreg*)*, 32
logLik.zipath *(*predict.zipath*)*, 69
loss2, 54, *56*, *57*
loss2_irsvm, 55, *55*, *57*
loss3, *55*, 56

Matrix, *20*, *51*
meatReg, *5*, *42*, 57, *74*
methods, 58
model.frame, *3*, *9*, *13*, *15*, *18*, *22*, *26*, *33*, *38*, *42*, *44*, *50*, *59*, *62*, *71*, *77*, *80*
model.matrix.zipath *(*predict.zipath*)*, 69

ncl, 59, *61*
ncl_fit, 60
nclreg, *23*, *25*, 62, *66*
nclreg_fit, 64

plot, *10*, *12*, *14*, *16*, *18*, *23*, *25*, *28*, *30*, *34*, *40*, *45*, *53*, *63*
plot.cv.glmreg *(*cv.glmreg*)*, 8
plot.cv.irglmreg *(*cv.irglmreg*)*, 15
plot.cv.nclreg *(*cv.nclreg*)*, 22
plot.glmreg, 67

predict, *10*, *12*, *14*, *16*, *18*, *23*, *25*, *27, 28*, *30*, *34*, *40*, *44, 45*, *51*, *53*, *60*, *63*, *70*, *85*
predict.cv.glmreg *(*cv.glmreg*)*, 8
predict.cv.zipath *(*cv.zipath*)*, 25
predict.glmreg, 68
predict.zipath, 69, *85*
predprob.zipath *(*predict.zipath*)*, 69
print, *34*, *40*, *44, 45*, *51*, *53*, *60*, *63*, *70*, *85*
print.summary.glmregNB *(*summary.glmregNB*)*, 76
print.summary.zipath *(*predict.zipath*)*, 69
pval.zipath, 71

residuals, *70*, *85*
residuals.zipath *(*predict.zipath*)*, 69
rzi, 72

sandwichReg, *5*, *58*, 73
se, 74
stan, 75
summary, *70*, *77*
summary.glmregNB, 76
summary.zipath *(*predict.zipath*)*, 69

terms, *4*, *31*
terms.zipath *(*predict.zipath*)*, 69
tuning.zipath, 77

update_wt, 79

zipath, *28*, *30, 31*, *69, 70*, *75*, *78*, 80, *86*
zipath_fit, *81*, 82