Package ‘mpath’

March 24, 2020

Title  Regularized Linear Models
Version  0.3-24
Date  2020-03-24
Author  Zhu Wang, with contributions from Achim Zeileis, Simon Jackman, Brian Ripley, Trevor Hastie, Rob Tibshirani, Balasubramanlan Narasimhan, Gil Chu and Patrick Breheny
Maintainer  Zhu Wang <wangz1@uthscsa.edu>
Description  Algorithms optimize penalized models. Currently the models include penalized Poisson, negative binomial, zero-inflated Poisson, zero-inflated negative binomial regression models and robust models. The penalties include least absolute shrinkage and selection operator (LASSO), smoothly clipped absolute deviation (SCAD), minimax concave penalty (MCP), and each possibly combining with L_2 penalty.

Imports  MASS, pscl, numDeriv, foreach, doParallel, bst
Depends  methods
Suggests  zic, R.rsp, knitr, gdata
VignetteBuilder  R.rsp, knitr
License  GPL-2
URL  https://github.com/zhuwang46/mpath

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

NeedsCompilation  yes
Repository  CRAN

Date/Publication  2020-03-24 17:00:02 UTC

R topics documented:

be.zeroinfl ..................................................  2
breadReg .....................................................  3
conv2glmreg ..................................................  4
conv2zipath ..................................................  5
**be.zeroinfl**

conduct backward stepwise variable elimination for zero inflated count regression

**be.zeroinfl**

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

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 <wangz1@uthscsa.edu>

References


---

*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.
conv2glmreg

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 <wangz1@uthscsa.edu>

References

See Also
`meatReg`, `sandwichReg`

Examples
```r
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))
```

conv2glmreg

convert glm object to class glmreg

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

Usage
```r
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 <wangz1@uthscsa.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 <wangz1@uthscsa.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.
cv.glmregNB

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 <wangz1@uthscsa.edu>

References


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.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=2, trace=FALSE, parallel=FALSE, ...)

cv.glmregNB

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 `nfolds` identifying what fold each observation is in. If supplied, `nfolds` 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 `k`th 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)"
cv.glmreg_fit

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 <wangz1@uthscsa.edu>

References

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.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"), nfolds=10, foldid, plot.it=TRUE, se=TRUE, n.cores=2, 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
- **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 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 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.
cv.nclreg

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 <wangz1@uthscsa.edu>

References


See Also

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

cv.nclreg  Cross-validation for nclreg

Description

Does k-fold cross-validation for nclreg, produces a plot, and returns cross-validated log-likelihood 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 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 <wangz1@uthscsa.edu>

References
cv.nclreg_fit

See Also

cv.nclreg and plot.nclreg 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 log-likelihood values for lambda

Usage

cv.nclreg_fit(x, y, weights, 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=2, 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
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 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 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 log-likelihood 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 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 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 <wangz1@uthscsa.edu>

References


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**

```r
## 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 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
vcv 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 <wangz1@uthscsa.edu>

References


See Also

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

```r
## Not run:
data("bioChemists", package = "pscl")
fm_zip <- cv.zipath(art ~ . | ., data = bioChemists, family = "poisson", nlambda=10)
### prediction from the best model
predict(fm_zip, newdata=bioChemists)
fm_zip_predict <- predict(object=fm_zip$fit, which=fm_zip$lambda.which,
  type="response", model=c("full"))
coef(fm_zip)
fm_znb <- cv.zipath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)
covf(fm_znb)
fm_zinb2 <- cv.zipath(art ~ . +offset(log(phd)) | ., data = bioChemists,
  family = "poisson", nlambda=10)
coef(fm_zinb2)
## 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

```r
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=2,
  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 <wangz1@uthscsa.edu>
References


See Also

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

Examples

```r
## Not run:
data("bioChemists")
fm_zip <- cv.ziopath(art ~ . | ., data = bioChemists, family = "poisson", nlambda=10)
### prediction from the best model
predict(fm_zip, newdata=bioChemists)
fm_zip_predict <- predict(object=fm_zip$fit, which=fm_zip$lambda.which,
type="response", model=c("full"))
coef(fm_zip)
fn_znb <- cv.ziopath(art ~ . | ., data = bioChemists, family = "negbin", nlambda=10)
coef(fm_znb)
fm_zinb2 <- cv.ziopath(art ~ . +offset(log(phd)) | ., data = bioChemists,
family = "poisson", nlambda=10)
coef(fm_zinb2)
## End(Not run)
```

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 <wangz1@uthscsa.edu>

References


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))

---

**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), scd (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

```r
## 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, ...)
```
glmreg

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

\[
\frac{1}{2} \cdot \text{weights} \cdot \text{RSS} + \lambda \cdot \text{penalty},
\]

if `standardize=FALSE` and

\[
\frac{1}{2} \cdot \sum(\text{weights}) \cdot \text{RSS} + \lambda \cdot \text{penalty},
\]

if `standardize=TRUE`. For the other models it is

\[- \sum(\text{weights} \cdot \text{loglik}) + \lambda \cdot \text{penalty}\]

if `standardize=FALSE` and

\[- \frac{\text{weights}}{\sum(\text{weights})} \cdot \text{loglik} + \lambda \cdot \text{penalty}\]

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.
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
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 <wangz1@uthscsa.edu>

References

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)

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=ifelse(!theta.fixed, theta0[1],NULL), link=log, penalty=c("enet","mnet","snet"), method="glmreg_fit", model=TRUE, x.keep=FALSE, y.keep=TRUE, contrasts=NULL, convex=FALSE, ...)

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 ≤ α ≤ 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
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

\[- \sum (weights \ast loglik) + \lambda \ast penalty\]

if standardize=FALSE and

\[- \frac{weights}{\sum(weights)} \ast loglik + \lambda \ast penalty\]

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
dev 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 <wangz1@uthscsa.edu>

References


See Also

print, predict, coef and plot methods, and the cv.glmregNB function.
 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 = rep(0, nobs), 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

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

\[
\frac{1}{2} \sum \text{weights} \times RSS + \lambda \times \text{penalty},
\]

if standardize=FALSE and

\[
\frac{1}{2} \sum \text{weights} \times RSS + \lambda \times \text{penalty},
\]

if standardize=TRUE. For the other models it is

\[- \sum (\text{weights} \times \text{loglik}) + \lambda \times \text{penalty}\]

if standardize=FALSE and

\[- \sum (\text{weights} \times \text{loglik}) + \lambda \times \text{penalty}\]

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 \times 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 \( 2*(\text{loglik}_\text{sat} - \text{loglik}) \), where \( \text{loglik}_\text{sat} \) 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 \( 2*(\text{loglik}_\text{sat} - \text{loglik(Null)}) \); The NULL model refers to the intercept model.

nobs number of observations

Author(s)

Zhu Wang <wangz1@uthscsa.edu>
hessianReg

References


See Also
glmreg

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 matReg 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 <wangz1@uthscsa.edu>

References

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))

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 <wangz1@uthscsa.edu>

References

Methods

See Also

sandwichReg, breadReg, estfunReg

Examples

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

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 <wangz1@uthscsa.edu>

References


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

$$\frac{1}{2} \ast \text{weights} \ast \text{loss}$$

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 <wangz1@uthscsa.edu>

References


See Also

print, predict, coef.

Examples

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

---

**nclreg**

*fit a nonconvex loss based robust linear model with lasso (or elastic net), snet or mnet 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

```r
## 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

\[
\frac{1}{2} \ast \text{weights} \ast \text{loss} + \lambda \ast \text{penalty},
\]

if `standardize=FALSE` and

\[
\frac{1}{2} \ast \frac{\text{weights}}{\text{sum(\text{weights})}} \ast \text{loss} + \lambda \ast \text{penalty},
\]

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.
**ncrreg_fit**

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.

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 <wangz1@uthscsa.edu>

**References**


**See Also**

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

**Examples**

```r
# 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 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "active", decreasing = TRUE, type.init = "bst")
fit2 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "active", decreasing = FALSE, type.init = "bst")
fit3 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "nonactive", decreasing = TRUE, type.init = "bst")
fit4 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "nonactive", decreasing = FALSE, type.init = "bst")
fit5 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "active", decreasing = TRUE, type.init = "ncl")
fit6 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "active", decreasing = FALSE, type.init = "ncl")
fit7 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "nonactive", decreasing = TRUE, type.init = "ncl")
fit8 = ncrreg(x, g2, s = 1, rfamily = "closs", type.path = "nonactive", decreasing = FALSE, type.init = "ncl")
```

**ncrreg_fit**

*Internal function to fit a nonconvex loss based robust linear model with lasso (or elastic net), snet and mnet 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

```r
nclreg_fit(x, y, weights, offset=NULL, cost=0.5,
           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"),
           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.
- **cost**: price to pay for false positive, 0 < cost < 1; price of false negative is 1-cost.
- **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.
- **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 $0 < \alpha \leq 1$. $\alpha=1$ 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 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="nc1", 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. direction=FALSE for decreasing sequence of lambda, 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 since 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 critera

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 consequentive lambda values change > epscycle, then re-estimate parameters in an effort to avoid trap of local optimiation.

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

$$\frac{1}{2} * \text{weights} * \text{loss} + \lambda * \text{penalty},$$
if standardize=FALSE and

\[
\frac{1}{2} \sum_{i} \text{weights} \cdot \text{loss} + \lambda \cdot \text{penalty},
\]

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 <wangz1@uthscsa.edu>

References


See Also

nclreg

---

**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, cost=0.5, 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.
- **cost**: price to pay for false positive, 0 < cost < 1; price of false negative is 1-cost.
- **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
- **reitol**: 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

\[ \frac{1}{2} \times \text{weights} \times \text{loss} \]

Value

An object with S3 class "nc1" 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 <wangz1@uthscsa.edu>

References


See Also

nc1
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 wangz1@uthscsa.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)
Model predictions based on a fitted "glmreg" object.

Description

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

Usage

```r
## S3 method for class 'glmreg'
predict(object, newx, which = 1:length(object$lambda),
type = c("link", "response", "class", "coefficients", "nonzero"),
newoffset = NULL,
na.action = na.pass, ...)
## S3 method for class 'glmreg'
coef(object, which = 1:length(object$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 <wangz1@uthscsa.edu>

References


See Also

`glmreg`
Examples

```r
## 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**

```r
## 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.
predict.zipath

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 <wangz1@uthscsa.edu>

References


See Also

zipath
Examples

```r
## 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

```r
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 <wangz1@uthscsa.edu>

References


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 <wangz1@uthscsa.edu>

References


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, ...)
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 <wangz1@uthscsa.edu>

References


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 <wangz1@uthscsa.edu>

References


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 <wangz1@uthscsa.edu>

References

See Also

summary.glm.nb

Examples

## Not run:
data(quine, package="MASS")
summary(glmregNB(Days ~ Eth*Age*Lrn*Sex, quine, link = log))
## End(Not run)

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"
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 <wangz1@uthscsa.edu>

References


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)
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,
zipath

terms a list with elements "count", "zero" and "full" containing the terms objects for the respective models,
theta estimate of the additional $\theta$ 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 <wangz1@uthscsa.edu>

References


See Also

zipath_fit, glmreg, glmregNB
Examples

```r
## 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)
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
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.
zipath_fit

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, rettol = 1e-5, thres=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 bi-
  nomial 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.
alpha.zero The elastic net mixing parameter for the zero part of model.
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: zaph
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 $\theta$ 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 <wangz1@uthscsa.edu>

References


See Also
zipath, glmreg, glmregNB
Index

*Topic **methods**
methods, 31

*Topic **models**
be.zeroinfl, 2
cv.glmreg, 5
cv.glmreg_fit, 9
cv.glmregNB, 7
cv.nclreg, 11
cv.nclreg_fit, 13
cv.zipath, 14
cv.zipath_fit, 17
glmreg, 20
glmreg_fit, 26
glmregNB, 23
ncl, 32
ncl_fit, 38
nclreg, 33
nclreg_fit, 35
plot.glmreg, 40
predict.glmreg, 41
predict.zipath, 42
pval.zipath, 44
rzi, 45
sandwichReg, 46
se, 47
tuning.zipath, 50
zipath, 52
zipath_fit, 54

AIC.glmreg (methods), 31
AIC.zipath (methods), 31

be.zeroinfl, 2
BIC.glmreg (methods), 31
BIC.zipath (methods), 31
breadReg, 3, 30, 31, 47

coef, 4, 7, 9, 11, 13, 14, 16, 18–20, 22, 25, 33, 35, 43, 57
coef.cv.glmreg (cv.glmreg), 5
coef.cv.nclreg (cv.nclreg), 11
coef.cv.zipath (cv.zipath), 14
coef.glmreg (predict.glmreg), 41
coef.zipath (predict.zipath), 42
conv2glmreg, 4
conv2zipath, 5
cv.glmreg, 5, 22
cv.glmreg_fit, 9
cv.glmregNB, 7, 25
cv.nclreg, 11, 35
cv.nclreg_fit, 13
cv.zipath, 14
cv.zipath_fit, 17
deviance.glmreg (glmreg), 20
estfunReg, 19, 30, 31
fitted, 43
fitted.zipath (predict.zipath), 42
glm.nb, 50
glmreg, 7, 11, 20, 29, 41, 53, 58
glmreg_fit, 26
glmregNB, 9, 23, 53, 58
glmregNegbin (glmregNB), 23
hessianReg, 29
logLik, 57
logLik.glmreg (glmreg), 20
logLik.zipath (predict.zipath), 42
meatReg, 4, 30, 30, 47
methods, 31
model.frame, 3, 6, 8, 11, 15, 21, 23, 32, 34, 44, 50, 52
model.matrix.zipath (predict.zipath), 42
ncl, 32, 39
ncl_fit, 38
nclreg, 13, 14, 33, 38
nclreg_fit, 35
plot, 7, 9, 11, 13, 14, 16, 19, 22, 25, 35
plot.cv.glmreg (cv.glmreg), 5
plot.cv.nclreg (cv.nclreg), 11
plot.glmreg, 40
predict, 7, 9, 11, 13, 14, 16, 19, 22, 25, 33, 35, 43, 57
predict.cv.glmreg (cv.glmreg), 5
predict.cv.zipath (predict.zipath), 14
predict.glmreg, 41
predict.zipath, 42, 57
predprob.zipath (predict.zipath), 42
print, 22, 25, 33, 35, 43, 57
print.summary.glmregNB (summary.glmregNB), 49
print.summary.zipath (predict.zipath), 42
pval.zipath, 44
residuals, 43, 57
residuals.zipath (predict.zipath), 42
rzi, 45
sandwichReg, 4, 31, 46
se, 47
stan, 48
summary, 43, 50
summary.glmregNB, 49
summary.zipath (predict.zipath), 42
terms, 4, 20
terms.zipath (predict.zipath), 42
tuning.zipath, 50
zipath, 16, 19, 20, 42, 43, 48, 51, 52, 58
zipath_fit, 53, 54