Package ‘glmtlp’

February 7, 2018

Type Package

Title Truncated Lasso Regularized Generalized Linear Models

Version 1.1

Date 2018-02-05

Author Chong Wu and Wei Pan

Maintainer Chong Wu <wuxx0845@umn.edu>

Description It provides an extremely efficient procedure for fitting the entire truncated lasso regularization path for linear regression, logistic and multinomial regression models, Poisson regression and the Cox model. The algorithm uses the difference of convex technique. The detail of the algorithm is described in Shen, Pan and Zhu (2012) <doi:10.1080/01621459.2011.645783>. The package is inherited from a popular R package 'glmnet' and many functions in 'glmnet' can be directly used in 'glmtlp'. You can learn more details by the online manual (<http://wuchong.org/glmtlp.html>).

License GPL-2

Depends R (>= 3.1.1), glmnet

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-07 13:50:51 UTC

R topics documented:

glmtlp-package *Truncated lasso model paths for some generalized linear models*

Description

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

Details

Package:	glmtlp
Type:	Package
Version:	1.1
Date:	2018-02-01
License:	GPL-2

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

Author(s)

Chong Wu, Wei Pan
Maintainer: Chong Wu <wuxx0845@umn.edu>

References

Xiaotong Shen , Wei Pan and Yunzhang Zhu (2012) *Likelihood-Based Selection and Sharp Parameter Estimation*, *Journal of the American Statistical Association, 107:497, 223-232*

cv.glmTLP *Cross-validation for glmTLP*

Description

Does k-fold cross-validation for glmTLP, produces a plot, and returns a value for `lambda` with pre-specified `tau`.

Usage

```
cv.glmTLP(x, y, family=c("gaussian","binomial","poisson","multinomial","cox","mgaussian"),
nfolds = 10, weights, offset=NULL, lambda, tau = 0.3,
nlambda=100, penalty.factor = rep(1, nvars),
 lambda.min.ratio=ifelse(nobs<nvars,1e-3,1e-4),
standardize=TRUE,intercept=TRUE,dfmax=nvars+1,
pmax=min(dfmax*2+20,nvars), lower.limits=-Inf,upper.limits=Inf,
standardize.response=FALSE, maxIter=100, Tol=1e-4)
```

Arguments

`x` x matrix as in `glmnet`.

`y` response variable. Quantitative for `family="gaussian"`, or `family="poisson"` (non-negative counts). For `family="binomial"` should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class). For `family="multinomial"`, can be a `nc>=2` level factor, or a matrix with `nc` columns of counts or proportions. For either `"binomial"` or `"multinomial"`, if `y` is presented as a vector, it will be coerced into a factor. For `family="cox"`, `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. The function `Surv()` in package **survival** produces such a matrix. For `family="mgaussian"`, `y` is a matrix of quantitative responses.

`family` Response type (see above)

`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`

`weights` Observation weights; defaults to 1 per observation

`offset` Offset vector (matrix) as in `glmnet`

`lambda` Optional user-supplied lambda sequence; default is `NULL`, and `glmTLP` chooses its own sequence

`tau` Tuning parameter.

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

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

`lambda.min.ratio`
Smallest value for `lambda`, as a fraction of `lambda.max`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default depends on the sample size `nobs` relative to the number of variables `nvars`. If `nobs > nvars`, the default is `0.0001`, close to zero. If `nobs < nvars`, the default is `0.01`. A very small value of `lambda.min.ratio` will lead to a

saturated fit in the nobs < nvars case. This is undefined for "binomial" and "multinomial" models, and glmnet will exit gracefully when the percentage deviance explained is almost 1.

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

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

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

pmax Limit the maximum number of variables ever to be nonzero

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

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

standardize.response

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

maxIter Maximum iteration for TLP.

Tol Tolerance.

Details

The function runs glmTLP nfolds+1 times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The error is accumulated, and the average error and standard deviation over the folds is computed. Note that cv.glmnet does NOT search for values for tau. A specific value should be supplied, else tau= 0.3 is assumed by default.

Value

an object of class "cv.glmnet" is returned, which is a list with the ingredients of the cross-validation fit. Although the implementation is different, we try to mimic returning as "cv.glment" in a popular package glmnet such that users can use truncated lasso as using elastic net.

lambda the values of lambda used in the fits.

cvm The mean cross-validated error - a vector of length length(lambda).

cvsd estimate of standard error of cvm.

cvup upper curve = cvm+cvsd.

cvlo lower curve = cvm-cvsd.

nzero number of non-zero coefficients at each lambda.

name a text string indicating type of measure (for plotting purposes).

glmnet.fit a fitted glmnet object for the full data.

lambda.min value of lambda that gives minimum cvm.

`lambda.1se` largest value of `lambda` such that error is within 1 standard error of the minimum.

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

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

Author(s)

Chong Wu
Maintainer: Chong Wu <wuxx0845@umn.edu>

References

Xiaotong Shen , Wei Pan and Yunzhang Zhu (2012) *Likelihood-Based Selection and Sharp Parameter Estimation*, *Journal of the American Statistical Association, 107:497, 223-232*

Examples

```
data("QuickStartExample")
fit = cv.glmTLP(x,y,tau = 1,nfolds = 2, lambda = c(0.1,05))
#We set nflods and lambda just to speed it up
# and pass the CRAN check. You should either use
# the default setting or search a larger space.
```

`glmTLP` *Fit a GLM with truncated lasso regularization*

Description

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

Usage

```
glmTLP(x, y, family=c("gaussian","binomial","poisson","multinomial","cox","mgaussian"),
    weights, offset=NULL, lambda, tau = 0.3, nlambda=100,
   penalty.factor = rep(1, nvars), lambda.min.ratio=ifelse(nobs<nvars,1e-3,1e-4),
    standardize=TRUE,intercept=TRUE,dfmax=nvars+1,pmax=min(dfmax*2+20,nvars),
    lower.limits=-Inf,upper.limits=Inf,
    standardize.response=FALSE, maxIter=100, Tol=1e-4)
```

Arguments

`x` input matrix, of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class `"sparseMatrix"` as in package `Matrix`; not yet available for `family="cox"`)

`y` response variable. Quantitative for `family="gaussian"`, or `family="poisson"` (non-negative counts). For `family="binomial"` should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class). For `family="multinomial"`, can be a `nc>=2` level factor, or a matrix with `nc` columns of counts or proportions. For either `"binomial"` or `"multinomial"`, if `y` is presented as a vector, it will be coerced into a factor. For `family="cox"`, `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. The function `Surv()` in package **survival** produces such a matrix. For `family="mgaussian"`, `y` is a matrix of quantitative responses.

`family` Response type (see above)

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

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

`tau` Write something about `tau`

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

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

`lambda.min.ratio`

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

`lambda` A user supplied `lambda` sequence. Typical usage is to have the program compute its own `lambda` sequence based on `nlambda` and `lambda.min.ratio`. Supplying a value of `lambda` overrides this. WARNING: use with care. Do not supply a single value for `lambda` (for predictions after CV use `predict()` instead). Supply instead a decreasing sequence of `lambda` values. `glmnet` relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.

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

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

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

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

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

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

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

`maxIter` Maximum iteration for TLP.

`Tol` Tolerance.

Details

Write something about the details.

Value

An object that inherits from `glmnet`.

`call` the call that produced this object

`a0` Intercept sequence of length `length(lambda)`

`beta` For `"elnet"`, `"lognet"`, `"fishnet"` and `"coxnet"` models, a `nvars x length(lambda)` matrix of coefficients, stored in sparse column format (`"CsparseMatrix"`). For `"multnet"` and `"mgaussian"`, a list of `nc` such matrices, one for each class.

`lambda` The actual sequence of `lambda` values used.

`dev.ratio` The fraction of (null) deviance explained (for `"elnet"`, this is the R-square). The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Hence dev.ratio=1-dev/nulldev.

`nulldev` Null deviance (per observation). This is defined to be 2*(loglike_sat -loglike(Null)); The NULL model refers to the intercept model, except for the Cox, where it is the 0 model.

`df` The number of nonzero coefficients for each value of `lambda`.

`dim` dimension of coefficient matrix (ices)

`nobs` number of observations

`npasses` total passes over the data summed over all lambda values

`offset` a logical variable indicating whether an offset was included in the model

`jerr` error flag, for warnings and errors (largely for internal debugging).

Author(s)

Chong Wu, Wei Pan
Maintainer: Chong Wu <wuxx0845@umn.edu>

References

Xiaotong Shen , Wei Pan and Yunzhang Zhu (2012) *Likelihood-Based Selection and Sharp Parameter Estimation*, *Journal of the American Statistical Association, 107:497, 223-232*

Examples

```
data("QuickStartExample")
fit = glmTLP(x,y, nlambda = 3)
#We set nlambda just to speed it up
# and pass the CRAN check. You should either use
# the default setting or search a larger space.
```

`QuickStartExample` *An example data set*

Description

This is a toy example data taken from glmnet package.

Usage

```
data(QuickStartExample)
```

Examples

```
data(QuickStartExample)
x
y
```

`x` *An example data set: x*

Description

This is a toy example data taken from glmnet package, QuickStartExample.

Usage

```
data(QuickStartExample)
```

Examples

```
data(QuickStartExample)
x
```

---

y *An example data set: y*

Description

This is a toy example data taken from glmnet package, QuickStartExample.

Usage

```
data(QuickStartExample)
```

Examples

```
data(QuickStartExample)
y
```

Index