Package 'relaxnet'

July 2, 2014

Type Package

Title Relaxation of glmnet models (as in relaxed lasso, Meinshausen 2007)

Version 0.3-2

Date 2013-08-16

Author Stephan Ritter, Alan Hubbard

Maintainer Stephan Ritter <stephanritterRpacks@gmail.com>

Depends glmnet

Suggests parallel

Description Extends the glmnet package with ``relaxation'', done by running glmnet once on the entire predictor matrix, then again on each different subset of variables from along the regularization path. Relaxation may lead to improved prediction accuracy for truly sparse data generating models, as well as fewer false positives (i.e. fewer noncontributing predictors in the final model). Penalty may be lasso (alpha = 1) or elastic net (0 < alpha < 1). For this version, family may be ``gaussian'' or ``binomial'' only. Takes advantage of fast FORTRAN code from the glmnet package.

License GPL (>= 2)

URL http://cran.r-project.org/package=relaxnet

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-16 18:29:00

R topics documented:

`relaxnet-package` *Relaxation (as in Relaxed Lasso, Meinshausen 2007) Applied to glmnet Models*

Description

Extends the glmnet package with "relaxation", done by running glmnet once on the entire predictor matrix, then again on each different subset of variables from along the regularization path. Penalty may be lasso (alpha = 1) or elastic net (0 < alpha < 1). For this version, family may be "gaussian" or "binomial" only. Takes advantage of fast fortran code from the glmnet package.

Note

This is a preliminary release and several additional features are planned for later versions.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

Maintainer: Stephan Ritter <sritter@berkeley.edu>

References

Stephan Ritter and Alan Hubbard, Tech report (forthcoming).

Jerome Friedman, Trevor Hastie, Rob Tibshirani (2010) "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* **33**(1)

Nicolai Meinshausen (2007) "Relaxed Lasso" *Computational Statistics and Data Analysis* **52**(1), 374-393

See Also

`relaxnet`, `cv.relaxnet`, `glmnet`

`cv.relaxnet` *Cross-Validation for relaxnet Models*

Description

Both of these functions will perform v-fold cross-validation to select tuning parameters for relaxnet models. `cv.relaxnet` will cross-validate on the value of lambda for both the main model and for the relaxed models, a two dimensional cross-validation. `cv.alpha.relaxnet` will in addition cross-validate on the value of alpha. For each value of alpha, relaxnet is run once on whole data set, then it is run again v times on subsets of the rows of the data.

Usage

```
cv.relaxnet(x, y, family = c("gaussian", "binomial"),
            nlambda = 100,
            alpha = 1,
            relax = TRUE,
            relax.nlambda = 100,
            relax.max.vars = min(nrow(x), ncol(x)) * 0.8,
            lambda = NULL,
            relax.lambda.index = NULL,
            relax.lambda.list = NULL,
            nfolds = 10,
            foldid,
            multicore = FALSE,
            mc.cores,
            mc.seed = 123,
            ...)

cv.alpha.relaxnet(x, y, family = c("gaussian", "binomial"),
                  nlambda = 100,
                  alpha = c(.1, .3, .5, .7, .9),
                  relax = TRUE,
                  relax.nlambda = 100,
                  relax.max.vars = min(nrow(x), ncol(x)) * 0.8,
                  lambda = NULL,
                  relax.lambda.index = NULL,
                  relax.lambda.list = NULL,
                  nfolds = 10,
                  foldid,
                  multicore = FALSE,
                  mc.cores,
                  mc.seed = 123,
                  ...)
```

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). Must have unique colnames.
y	response variable. Quantitative for family="gaussian". For family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions.
family	Response type (see above).
nlambda	The number of lambda values - default is 100. Determines how fine the grid of lambda values should be.
alpha	Elastic net mixing parameter (see glmnet). For cv.relaxnet, this should be a single value. For cv.alpha.relaxnet it should be a vector of values.
relax	Should the model be relaxed. If FALSE, only the main glmnet model is run and no relaxed models are.
relax.nlambda	Like nlambda but for secondary (relaxed) models.
relax.max.vars	Maximum number of variables for relaxed models. No relaxation will be done for subsets along the regularization path with number of variables greater than relax.max.vars. If ncol(x) > nrow(x) and alpha < 1, it may make sense to use a value > nrow(x), but this may lead to increased computation time.
lambda	See (see glmnet). Optional: default is to let glmnet choose its own sequence.
relax.lambda.index	Vector which indexes the lambda argument and specifyies the values at which a relaxed model should be fit. Optional: default is to let relaxnet determine these values based on the beta matrix from the main glmnet fit. Ignored if lambda argument is NULL.
relax.lambda.list	List of lambda values to use for the relaxed models. Optional: default is to let relaxnet determine these values. Ignored if lambda argument is NULL.
nfolds	Number of folds - default is 10. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds=3.
foldid	An optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfolds can be missing.
multicore	Should execution be parallelized over cv folds (for cv.relaxnet) or over alpha values (for cv.alpha.relaxnet) using multicore functionality from R's parallel package?
mc.cores	Number of cores/cpus to be used for multicore processing. Processing will be most efficient if nfolds (for cv.relaxnet) or the length of alpha (for cv.alpha.relaxnet) is a multiple of mc.cores. Ignored if multicore is FALSE
mc.seed	Integer value with which to seed the RNG when using parallel processing (internally, RNGkind will be called to set the RNG to "L'Ecuyer-CMRG"). Will be ignored if multicore is FALSE. If mulicore is FALSE, one should be able to get reprodicible results by setting the seed normally (with set.seed) prior to running.

`...` Further aruments passed to glmnet. Use with caution as this has not yet been tested. For example, setting `standardize = FALSE` will probably work correctly, but setting an offset probably won't.

Details

cv.glmnet's `type.measure` argument has not yet been implemented. For type = gaussian models, mean squared error is used, and for type = binomial, binomial deviance is used.

Value

For `cv.relaxnet` – an object of class `"cv.relaxnet"` containing the following slots:

`call` A copy of the call which produced this object

`relax` The value of the relax argument. If this is `FALSE`, then several of the other elements of this result will be set to `NA`.

`lambda` lambda sequence used for this fit

`cvm` The mean cross-validated error - a vector of length `length(lambda)`. For main model.

`cvsd` estimate of standard error of `cvm` for main model.

`cvup` upper curve = `cvm+cvsd` for main model.

`cvlo` lower curve = `cvm-cvsd` for main model.

`nzero` number of non-zero coefficients at each `lambda` for main model.

`name` a text string indicating type of measure.

`relaxnet.fit` Fitted relaxnet object for the full data.

`relax.cvstuff.list`
List containing cvm and cvsd for each of the relaxed models.

`relax.lambda.list.trunc`
List containing the values of lambda used for cross-validation for each relaxed model

`which.model.min`
This will have value `"main"` if the main model "won" the cross-validation, and if not, it will be an integer specifying which relaxed model won (i.e. which element of relaxnet.fit$relax.glmnet.fits).

`overall.lambda.min`
The value of lambda with overall min cvm (i.e. from the submodel specified by `which.model.min`).

`min.cvm` The overall minimum value of cvm

`main.lambda.min`
lambda.min, restricted to main model only.

`main.lambda.1se`
lambda.1se, restricted to main model only (see `cv.glmnet`).

`main.min.cvm` Minimum of cvm, restricted to main model only.

`total.time` Time in seconds to fit this object (full data fit plus all cross-validation)

`full.data.fit.time`
: Time in seconds to fit the relaxnet.fit.

`cv.fit.times`
: Time in seconds to fit the models for each set of cv folds.

For `cv.alpha.relaxnet` – an object of class `"cv.alpha.relaxnet"` containing the following slots:

`call`
: A copy of the call which produced this object

`relax`
: The value of the relax argument. If this is `FALSE`, then several of the other elements of this result will be set to `NA`.

`alpha`
: The alpha values used.

`cv.relaxnet.results`
: List of cv.relaxnet objects, one for each alpha value.

`which.alpha.min`
: The alpha value which "won" the cross-validation.

`total.time`
: Time in seconds to fit this object.

Note

This is a preliminary release and several additional features are planned for later versions.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

References

Stephan Ritter and Alan Hubbard, Tech report (forthcoming).

Jerome Friedman, Trevor Hastie, Rob Tibshirani (2010) "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* **33**(1)

Nicolai Meinshausen (2007) "Relaxed Lasso" *Computational Statistics and Data Analysis* **52**(1), 374-393

See Also

`relaxnet`, `predict.cv.relaxnet`

Examples

```
## generate predictor matrix

nobs <- 100
nvars <- 200

set.seed(23)
```

```
x <- matrix(rnorm(nobs * nvars), nobs, nvars)

## make sure it has unique colnames

colnames(x) <- paste("x", 1:ncol(x), sep = "")

## let y depend on first 5 columns plus noise

y <- rowSums(x[, 1:5]) + rnorm(nrow(x))

## run cv.relaxnet

cv.result <- cv.relaxnet(x, y)

predict(cv.result, type = "nonzero")

## very few false positives compared to glmnet alone

## glmnet min rule

predict(cv.result$relaxnet.fit$main.glmnet.fit,
        type = "nonzero",
        s = cv.result$main.lambda.min)

## glmnet 1se rule

predict(cv.result$relaxnet.fit$main.glmnet.fit,
        type = "nonzero",
        s = cv.result$main.lambda.1se)

## get values of the coefs for cv.relaxnet's chosen fit

coefs <- drop(predict(cv.result, type = "coef"))

coefs[coefs != 0]
```

`predict.cv.relaxnet` *Predict Methods for cv.relaxnet and cv.alpha.relaxnet Objects*

Description

Similar to other predict methods, this functions predicts fitted values, logits, coefficients and more from a fitted `"cv.relaxnet"` or `"cv.alpha.relaxnet"` object. By default, predictions are made at those values of the tuning parameters which "won" the cross-validation.

Usage

```
## S3 method for class 'cv.relaxnet'
predict(object,
        newx,
```

```
        which.model = object$which.model.min,
        s = object$overall.lambda.min,
        type = c("link", "response", "coefficients", "nonzero", "class"),
        exact = FALSE,
        ...)

## S3 method for class 'cv.alpha.relaxnet'
predict(object,
        newx,
        alpha.val = object$which.alpha.min,
        type = c("link", "response", "coefficients", "nonzero", "class"),
        ...)
```

Arguments

`object`	The object from which predictions are to be made.
`newx`	Matrix of new values for `x` at which predictions are to be made. Must be a matrix; can be sparse as in `Matrix` package. This argument is not used for `type=c("coefficients","nonzero")`
`alpha.val`	Value of alpha at which predictions are to be made. Default is to use that value which "won" the cross-validation.
`which.model`	Specifies the submodel from which predictions are required. `"main"` indicates the main glmnet model, while an integer indicates one of the relaxed models. Default for both functions is to use the submodel which "won" the cross-validation.
`s`	Value(s) of the penalty parameter `lambda` at which predictions are required. Default for both functions is to use that value which "won" the cross-validation.
`type`	See `link[glmnet]{predict.glmnet}`.
`exact`	Only the default, `FALSE`, is supported. See `link[glmnet]{predict.glmnet}`.
`...`	Further arguments passed to predict.relaxnet or to predict.cv.relaxnet (for the alpha version).

Value

The object returned depends on type.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

See Also

`relaxnet`, `cv.relaxnet`, `cv.alpha.relaxnet`, `predict.relaxnet`, `predict.glmnet`

predict.relaxnet *Predict Method for* "relaxnet" *Objects*

Description

Similar to other predict methods, this functions predicts fitted values, logits, coefficients and more from a fitted "relaxnet" object.

Usage

```
## S3 method for class 'relaxnet'
predict(object,
        newx,
        which.model,
        s = NULL,
        type = c("link", "response", "coefficients", "nonzero", "class"),
        exact = FALSE,
...)
```

Arguments

object	The "relaxnet" object from which to get predictions.
newx	Matrix of new values for x at which predictions are to be made. Must be a matrix; can be sparse as in Matrix package. This argument is not used for type=c("coefficients","nonzero")
which.model	Specifies the submodel from which predictions are required. "main" indicates the main glmnet model, while an integer indicates one of the relaxed models.
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence of lambda values for the model specified by which.model.
type	See link[glmnet]{predict.glmnet}.
exact	Only the default, FALSE, is supported. See link[glmnet]{predict.glmnet}.
...	Further arguments passed to predict.glmnet. In the current version, these are not guaranteed to work correctly (for example, offset has not yet been implemented for relaxnet).

Value

The object returned depends on type.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

See Also

relaxnet, predict.glmnet

---

print.relaxnet *Print Method for relaxnet Objects*

Description

This function just calls print(summary(x)). See summary.relaxnet.

Usage

```
## S3 method for class 'relaxnet'
print(x, digits, ...)
```

Arguments

x	The "relaxnet" object to be printed.
digits	Passed to print.summary.relaxnet.
...	Passed to print.summary.relaxnet.

Value

Returns x invisibly.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

See Also

relaxnet, summary.relaxnet

relaxnet *Relaxation (as in Relaxed Lasso, Meinshausen 2007) applied to glmnet Models*

Description

Runs glmnet once on the full x matrix, then again on each distinct subset of columns from along the solution path. The penalty may be lasso (alpha = 1) or elastic net (0 < alpha < 1). The outcome (y) may be continuous or binary.

Usage

```
relaxnet(x, y, family = c("gaussian", "binomial"),
         nlambda = 100,
         alpha = 1,
         relax = TRUE,
         relax.nlambda = 100,
         relax.max.vars = min(nrow(x), ncol(x)) * 0.8,
         lambda = NULL,
         relax.lambda.index = NULL,
         relax.lambda.list = NULL,
         ...)
```

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). Must have unique colnames.
y	response variable. Quantitative for family="gaussian". For family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions.
family	Response type (see above).
nlambda	The number of lambda values - default is 100. Determines how fine the grid of lambda values should be.
alpha	Elastic net mixing parameter (see glmnet).
relax	Should the model be relaxed. If FALSE, only the main glmnet model is run and no relaxed models are.
relax.nlambda	Like nlambda but for secondary (relaxed) models.
relax.max.vars	Maximum number of variables for relaxed models. No relaxation will be done for subsets along the regularization path with number of variables greater than relax.max.vars. If ncol(x) > nrow(x) and alpha < 1, it may make sense to use a value > nrow(x), but this may lead to increased computation time.
lambda	See (see glmnet). Optional and meant primarily for use by cv.relaxnet.

`relax.lambda.index`
: Vector which indexes the lambda argument and specifyies the values at which a relaxed model should be fit. Optional and meant primarily for use by `cv.relaxnet`. Ignored if lambda argument is `NULL`.

`relax.lambda.list`
: List of lambda values to use for the relaxed models. Optional and meant primarily for use by `cv.relaxnet`. Ignored if lambda argument is `NULL`.

`...`
: Further aruments passed to glmnet. Use with caution as this has not yet been tested. For example, setting `standardize = FALSE` will probably work correctly, but setting an offset probably won't.

Details

Version 1.9-5 of glmnet no longer allows single-column x. This broke relaxnet. As a temporary fix, relaxed models containing a single variable now just run glm instead of glmnet, and only the full least squares (or logistic regression, for family = "binomial") solution is considered for that relaxed model. All relaxed models containing more than one variable, as well as the main model, still use the complete glmnet solution path.

Value

Object of class code"relaxnet" with the following components:

`call`
: A copy of the call which produced this object

`main.glmnet.fit`
: The object resulting from running `glmnet` on the entire `x` matrix.

`relax`
: The value of the relax argument. If this is `FALSE`, then several of the other elements of this result will be set to `NA`.

`relax.glmnet.fits`
: A list containing the secondary `glmnet` fits gotten by running `glmnet` on the distinct subsets of the columns of `x` resulting along the solution path of lambda values.

`relax.num.vars`
: Vector giving the number of variables in each "relaxed" model.

`relax.lambda.index`
: This vector indexes result$main.glmnet.fit$lambda and gives the lambda values at which the relax.glmnet.fits were obtained.

`total.time`
: Total time in seconds to produce this result.

`main.fit.time`
: Time in seconds to produce the main glmnet fit.

`relax.keep`
: In certain cases some of the relaxed models are removed after fitting. `relax.fit.times` records times for these removed models as well. This logical vector shows which of the models whose timings are given in `relax.fit.times` were actually kept and have results given in `relax.glmnet.fits` above. Hopefully this will not be necessary in later versions.

`relax.fit.times`
: Vector of times in seconds to produce secondary "relaxed" models.

Note

This is a preliminary release and several additional features are planned for later versions.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

References

Stephan Ritter and Alan Hubbard, Tech report (forthcoming).

Jerome Friedman, Trevor Hastie, Rob Tibshirani (2010) "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* **33**(1)

Nicolai Meinshausen (2007) "Relaxed Lasso" *Computational Statistics and Data Analysis* **52**(1), 374-393

See Also

glmnet, cv.relaxnet, predict.relaxnet

Examples

```
## generate predictor matrix

nobs <- 100
nvars <- 200

set.seed(23)

x <- matrix(rnorm(nobs * nvars), nobs, nvars)

## make sure it has unique colnames

colnames(x) <- paste("x", 1:ncol(x), sep = "")

## let y depend on first 5 columns plus noise

y <- rowSums(x[, 1:5]) + rnorm(nrow(x))

## default is family = "gaussian"

result1 <- relaxnet(x, y)

summary(result1)

## now fit family = "binomial" model

y.bin <- rbinom(nrow(x), 1, prob = plogis(0.2 * rowSums(x[, 1:5])))
```

```
result2 <- relaxnet(x, y.bin, family = "binomial")

summary(result2)
```

`summary.relaxnet` *Generate and print summaries of class* `"relaxnet"` *objects.*

Description

Print summaries of relaxnet objects.

Usage

```
## S3 method for class 'relaxnet'
summary(object, ...)
## S3 method for class 'summary.relaxnet'
print(x, digits = 3, ...)
```

Arguments

`object`	The `"relaxnet"` object to summarize.
`x`	An object of class `"summary.relaxnet"`.
`digits`	Passed to `print.default`.
`...`	For the summary method: ignored. For the print method: passed to `print.default`.

Value

For the summary method: an object of type `"summary.relaxnet"` containing a subset of the elements of `object`.

For the print method: returns `x` invisibly.

Author(s)

Stephan Ritter, with design contributions from Alan Hubbard.

Much of the code (and some help file content) is adapted from the **glmnet** package, whose authors are Jerome Friedman, Trevor Hastie and Rob Tibshirani.

See Also

`relaxnet`, `print.relaxnet`

Index