Package ‘sparsenet’

November 16, 2024

Type Package

Title Fit Sparse Linear Regression Models via Nonconvex Optimization

Version 1.7

Date 2024-11-14

Description Efficient procedure for fitting regularization paths between L1 and L0, using the MC+ penalty of Zhang, C.H. (2010)<doi:10.1214/09-AOS729>. Implements the methodology described in Mazumder, Friedman and Hastie (2011) <DOI:10.1198/jasa.2011.tm09738>. Sparsenet computes the regularization surface over both the family parameter and the tuning parameter by coordinate descent.

Depends Matrix (>= 1.0-6), shape

Imports methods

License GPL-2

NeedsCompilation yes

URL https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf

RoxygenNote 7.3.1

Author Trevor Hastie [aut, cre],
Rahul Mazumder [aut],
Jerome Friedman [aut]

Maintainer Trevor Hastie <hastie@stanford.edu>

Repository CRAN

Date/Publication 2024-11-16 22:30:02 UTC

Contents

sparsenet-package *Fit a linear model regularized by the nonconvex MC+ sparsity penalty*

Description

Sparsenet uses coordinate descent on the MC+ nonconvex penalty family, and fits a surface of solutions over the two-dimensional parameter space.

Details

At its simplest, provide `x,y` data and it returns the solution paths. There are tools for prediction, cross-validation, plotting and printing.

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit=sparsenet(x,y)
plot(fit)
cvfit=cv.sparsenet(x,y)
plot(cvfit)
```

cv.sparsenet *Cross-validation for sparsenet*

Description

Does k-fold cross-validation for sparsenet, produces a plot, and returns values for gamma, lambda

Usage

```
cv.sparsenet(x, y, weights, type.measure = c("mse", "mae"), ...,nfolds = 10,
       foldid, keep=FALSE, trace.it=FALSE)
```

Arguments

`x` x matrix as in `sparsenet`.

`y` response `y` as in `sparsenet`.

`weights` Observation weights; defaults to 1 per observation

`type.measure` loss to use for cross-validation. Currently two options: squared-error (`type.measure="mse"`) or mean-absolute error (`type.measure="mae"`)

`...` Other arguments that can be passed to `sparsenet`.

`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 whhat fold each observation is in. If supplied, `nfold` can be missing.

`keep` If `TRUE`, we include the prevalidation array as component `fit.preval` on the returned object. Default is `keep = FALSE`.

`trace.it` If `TRUE`, then we get a printout that shows the progress

Details

The function runs `sparsenet` `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.

Value

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

`lambda` the values of `lambda` used in the fits. This is an `nlambda` x `ngamma` matrix

`cvm` The mean cross-validated error - a matrix shaped like 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, gamma` pair.

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

`sparsenet.fit` a fitted sparsenet object for the full data.

`call` The call that produced this object

`parms.min` values of `gamma, lambda` that gives minimum `cvm`.

`which.min` indices for the above

`lambda.1se` `gamma, lambda` of smallest model (df) such that error is within 1 standard error of the minimum.

`which.1se` indices of the above

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, `https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf`

See Also

`glmnet` package, `predict`, `coef`, `print` and `plot` methods, and the `sparsenet` function.

Examples

```
train.data=gendata(100,1000,nonzero=30,rho=0.3,snr=3)
fit=sparsenet(train.data$x,train.data$y)
par(mfrow=c(3,3))
plot(fit)
par(mfrow=c(1,1))
fitcv=cv.sparsenet(train.data$x,train.data$y,trace.it=TRUE)
plot(fitcv)
```

gendata *Generate data for testing sparse model selection*

Description

This function generates x/y data for testing sparsenet and glmnet

Usage

```
gendata(N, p, nonzero, rho, snr = 3, alternate = TRUE)
```

Arguments

`N`	Sample size (eg 500)
`p`	Number of features or variables (eg 1000)
`nonzero`	Number if nonzero coefficients (eg 30)
`rho`	pairwise correlation between features
`snr`	Signal to noise ratio - SD signal/ SD noise - try 3
`alternate`	Alternate sign of coefficients

Details

Generates Gaussian x and y data. The nonzero coefficients decrease linearly in absolute value from nonzero down to 0. If `alternate=TRUE` their signs alternate, else not

Value

A list with components x and y as well some other details about the dataset

Author(s)

Trevor Hastie and Jerome Friedman

Examples

```
train.data=gendata(100,1000,nonzero=30,rho=0.3,snr=3)
fit=sparsenet(train.data$x,train.data$y)
par(mfrow=c(3,3))
plot(fit)
par(mfrow=c(1,1))
fitcv=cv.sparsenet(train.data$x,train.data$y,trace.it=TRUE)
plot(fitcv)
```

`plot.cv.sparsenet` *plot the cross-validation curves produced by cv.sparsenet*

Description

Plots the cross-validation curves for each value of `gamma` in one figure, as a function of the `lambda` values used.

Usage

```
## S3 method for class 'cv.sparsenet'
plot(x,  ...)
```

Arguments

`x`	fitted "`cv.sparsenet`" object
`...`	Other graphical parameters to plot

Details

A plot is produced, and nothing is returned.

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, `https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf`

See Also

`glmnet` package, `sparsenet`, `cv.sparsenet` and `print` and `plot` methods for both.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fitcv=cv.sparsenet(x,y)
plot(fitcv)
```

`plot.sparsenet` *plot coefficients from a "sparsenet" object*

Description

Produces a series of coefficient profile plots of the coefficient paths for a fitted "`sparsenet`" object.

Usage

```
## S3 method for class 'sparsenet'
plot(x, xvar = c(”rsq”,”lambda”,”norm”), which.gamma=NULL, label = FALSE,...)
```

Arguments

`x` fitted `"sparsenet"` model

`xvar` What is on the X-axis. `"rsq"` plots against the percent variance explained on the training data, `"lambda"` against the log-lambda sequence, and `"norm"` plots against the L1-norm of the coefficients

`which.gamma` sequence numbers of `gamma` values to be used in the plots; default is all used in the fit

`label` If `TRUE`, label the curves with variable sequence numbers.

`...` Other graphical parameters to plot

Details

A series of coefficient profile plots is produced, one for each `gamma` specified. Users should set up the appropriate layout.

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, `https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf`

See Also

`glmnet` package, `sparsenet`, `cv.sparsenet` and `print` and `plot` methods for both.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit=sparsenet(x,y)
par(mfrow=c(3,3))
plot(fit)
```

predict.cv.sparsenet *make predictions from a "cv.sparsenet" object.*

Description

This function makes predictions from a cross-validated sparsenet model, using the stored "`sparsenet.fit`" object, and the optimal value chosen for `lambda`.

Usage

```
## S3 method for class 'cv.sparsenet'
predict(object, newx, which=c("parms.min","parms.1se"),...)
## S3 method for class 'cv.sparsenet'
coef(object, which=c("parms.min","parms.1se"),...)
```

Arguments

`object` Fitted "`cv.sparsenet`" object.

`newx` Matrix of new values for `x` at which predictions are to be made. Must be a matrix. See documentation for `predict.sparsenet`.

`which` Either the paramaters of the minimum of the CV curves (default "`parms.min`" or the parameters corresponding to the one standard-error rule `parms.1se`)

`...` Not used. Other arguments to predict.

Details

This function makes it easier to use the results of cross-validation to make a prediction.

Value

The object returned depends the ... argument which is passed on to the `predict` method for `sparsenet` objects.

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf

See Also

`glmnet` package, `sparsenet`, `cv.sparsenet` and `print` and `plot` methods for both.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fitcv=cv.sparsenet(x,y)
predict(fitcv,x)
```

`predict.sparsenet` *make predictions from a "sparsenet" object.*

Description

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

Usage

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

Arguments

`object` Fitted `"sparsenet"` model object.

`newx` Matrix of new values for `x` at which predictions are to be made. Must be a matrix. This argument is not used for `type=c("coefficients","nonzero")`

`s` Value(s) of the penalty parameter `lambda` at which predictions are required. Default is the entire sequence used to create the model.

`which.gamma` Index or indices of `gamma` values at which predictions are to be made. Default is all those used in the fit

`type` `"response"` returns fitted predictions at `newx`. Type `"coefficients"` computes the coefficients at the requested values for `s`. Type `"nonzero"` returns lists of the indices of the nonzero coefficients for each value of `s`.

`exact` By default (`exact=FALSE`) the predict function uses linear interpolation to make predictions for values of `s` that do not coincide with those used in the fitting algorithm. Currently `exact=TRUE` is not implemented, but prints an error message telling the user how to achieve the exact predictions. This is done my rerunning the algorithm with the desired values interspersed (in order) with the values used in the original fit

`...` Not used. Other arguments to predict.

Details

The shape of the objects returned depends on which `which.gamma` has more than one element. If more than one element, a list of predictions is returned, one for each gamma.

Value

The object returned depends on type.

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf

See Also

`glmnet` package, `sparsenet`, `cv.sparsenet` and `print` and `plot` methods for both.

Examples

```
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit=sparsenet(x,y)
predict(fit, which.gamma=5,type="nonzero")
predict(fit,x)
```

---

sparsenet *Fit a linear model regularized by the nonconvex MC+ sparsity penalty*

Description

Sparsenet uses coordinate descent on the MC+ nonconvex penalty family, and fits a surface of solutions over the two-dimensional parameter space. This penalty family is indexed by an overall strength paramter lambda (like lasso), and a convexity parameter gamma. Gamma = infinity corresponds to the lasso, and gamma = 1 best subset.

Usage

```
sparsenet(x, y, weights, exclude, dfmax = nvars + 1, pmax = min(dfmax *2, nvars),
ngamma = 9, nlambda = 50, max.gamma = 150, min.gamma = 1.000001,
lambda.min.ratio = ifelse(nobs < nvars, 0.01, 1e-04), lambda = NULL,
gamma = NULL, parms = NULL, warm = c("lambda", "gamma", "both"),
thresh = 1e-05, maxit = 1e+06)
```

Arguments

`x` Input matrix of nobs x nvars predictors

`y` response vector

`weights` Observation weights; default 1 for each observation

`exclude` Indices of variables to be excluded from the model. Default is none. Since by default `sparsenet` standardizes all the variables before fitting the path (see details), any variable with zero variance is automatically excluded.

`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

`ngamma` Number of gamma values, if `gamma` not supplied; default is 9.

`nlambda` Number of lambda values, if `lambda` not supplied; default is 50

`max.gamma` Largest gamma value to be used, apart from infinity (lasso), if `gamma` not supplied; default is 150

`min.gamma` Smallest value of gamma to use, and should be >1; default is 1.000001

`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.

`lambda` A user supplied `lambda` sequence, in decreasing order. 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. `sparsenet` relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.

`gamma` Sparsity parameter vector, with 1<gamma<infty. Gamma=1 corresponds to best-subset regression, gamma=infty to the lasso. Should be given in decreasing order.

`parms` An optional three-dimensional array: 2x ngamma x nlambda. Here the user can supply exactly the gamma, lambda pairs that are to be traversed by the coordinate descent algorithm.

`warm` How to traverse the grid. Default is "lambda", meaning warm starts from the previous lambda with the same gamma. "gamma" means the opposite, previous gamma for the same lambda. "both" tries both warm starts, and uses the one that improves the criterion the most.

`thresh` Convergence threshold for coordinate descent. Each coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than `thresh` times the null Rss. Defaults value is `1E-5`.

`maxit` Maximum number of passes over the data for all lambda/gamma values; default is 10^6.

Details

This algorithm operates like `glmnet`, with its alpha parameter which moves the penalty between lasso and ridge; here gamma moves it between lasso and best subset. The algorithm traverses the two dimensional gamma/lambda array in a nested loop, with decreasing gamma in the outer loop, and decreasing lambda in the inner loop. Because of the nature of the MC+ penalty, each coordinate update is a convex problem, with a simple two-threshold shrinking scheme: beta< lambda set to zero; beta > lambda*gamma leave alone; beta inbetween, shrink proportionally. Note that this algorithm ALWAYS standardizes the columns of x and y to have mean zero and variance 1 (using the 1/N averaging) before it computes its fit. The coefficients reflect the original scale.

Value

An object of class `"sparsenet"`, with a number of components. Mostly one will access the components via generic functions like `coef()`, `plot()`, `predict()` etc.

`call`	the call that produced this object
`rsq`	The percentage variance explained on the training data; an ngamma x nlambda matrix.
`jerr`	error flag, for warnings and errors (largely for internal debugging).
`coefficients`	A coefficient list with ngamma elements; each of these is a coefficient list with various components: the matrix beta of coefficients, its dimension dim, the vector of intercepts, the lambda sequence, the gamma value, the sequence of df (nonzero coefficients) for each solution.
`parms`	Irrespective how the parameters were input, the three-way array of what was used.
`gamma`	The gamma values used
`lambda`	The lambda values used
`max.lambda`	The entry value for lambda

Author(s)

Rahul Mazumder, Jerome Friedman and Trevor Hastie

Maintainer: Trevor Hastie <hastie@stanford.edu>

References

Mazumder, Rahul, Friedman, Jerome and Hastie, Trevor (2011) *SparseNet: Coordinate Descent with Nonconvex Penalties. JASA, Vol 106(495), 1125-38*, `https://hastie.su.domains/public/Papers/Sparsenet/Mazumder-SparseNetCoordinateDescent-2011.pdf`

See Also

`glmnet` package, `predict`, `coef`, `print` and `plot` methods, and the `cv.sparsenet` function.

Examples

```
train.data=gendata(100,1000,nonzero=30,rho=0.3,snr=3)
fit=sparsenet(train.data$x,train.data$y)
par(mfrow=c(3,3))
plot(fit)
par(mfrow=c(1,1))
fitcv=cv.sparsenet(train.data$x,train.data$y,trace.it=TRUE)
plot(fitcv)
```

Index