Description

This function gets coefficients or makes coefficient predictions from a cross-validated gcdnet model, using the stored "gcdnet.fit" object, and the optimal value chosen for lambda.

Usage

```r
# S3 method for class 'cv.gcdnet'
coef(object, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

- `object`: fitted `cv.gcdnet` object.
- `s`: value(s) of the penalty parameter lambda at which predictions are required. Default is the value `s="lambda.1se"` stored on the CV object, it is the largest value of lambda such that error is within 1 standard error of the minimum. Alternatively `s="lambda.min"` can be used, it is the optimal value of lambda that gives minimum cross validation error cvm. If `s` is numeric, it is taken as the value(s) of lambda to be used.
- `...`: not used. Other arguments to predict.

Details

This function makes it easier to use the results of cross-validation to get coefficients or make coefficient predictions.

Value

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

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


http://www.jstatsoft.org/v33/i01/

See Also
cv.gcdnet, and predict.cv.gcdnet methods.

Examples
data(FHT)
set.seed(2011)
cv=cv.cv.gcdnet(FHT$x, FHT$y,
lambda2 = 1,nfolds=5)
coef(cv,s="lambda.min")

**coef.gcdnet**

get coefficients or make coefficient predictions from an "gcdnet" object.

Description
Computes the coefficients or returns a list of the indices of the nonzero coefficients at the requested values for lambda from a fitted gcdnet object.

Usage
```r
## S3 method for class 'gcdnet'
coef(object, s = NULL,
type=c("coefficients","nonzero"), ...)
```

Arguments

- **object**: fitted gcdnet model object.
- **s**: value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
- **type**: type "coefficients" computes the coefficients at the requested values for s. Type "nonzero" returns a list of the indices of the nonzero coefficients for each value of s. Default is "coefficients".
- **...**: not used. Other arguments to predict.

Details

s is the new vector at which predictions are requested. If s is not in the lambda sequence used for fitting the model, the coef function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right lambda indices.
Value

The object returned depends on type.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


[http://www.jstatsoft.org/v33/i01/](http://www.jstatsoft.org/v33/i01/)

See Also

* predict.gcdnet * method

Examples

data(FHT)
fit1=gcdnet(x=FHT$x, y=FHT$y)
coef(fit1,type="coef", s=c(0.1, 0.005))
coef(fit1,type="nonzero")

Description

Does k-fold cross-validation for gcdnet, produces a plot, and returns a value for lambda. This function is modified based on the cv function from the *glmnet* package.

Usage

cv.gcdnet(x, y, lambda, pred.loss, nfolds, foldid, delta,...)
**Arguments**

- **x**
  - x matrix as in `gcdnet`.

- **y**
  - response variable or class label y as in `gcdnet`.

- **lambda**
  - optional user-supplied lambda sequence; default is NULL, and `gcdnet` chooses its own sequence.

- **nfolds**
  - number of folds - default is 5. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds=3.

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

- **pred.loss**
  - loss function to use for cross-validation error. Valid options are:
    - "loss" Margin based loss function. When use least square loss "ls", it gives mean square error (MSE).
    - "misclass" only available for classification: it gives misclassification error.
  - Default is "loss".

- **delta**
  - parameter δ only used in HHSVM for computing margin based loss function, only available with pred.loss = "loss".

- **...**
  - other arguments that can be passed to gcdnet.

**Details**

The function runs `gcdnet` 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 average error and standard deviation over the folds are computed.

**Value**

An object of class `cv.gcdnet` is returned, which is a list with the ingredients of the cross-validation fit.

- **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.

- **cvupper**
  - upper curve = cvm+cvsd.

- **cvlower**
  - lower curve = cvm-cvsd.

- **nzero**
  - number of non-zero coefficients at each lambda.

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

- **gcdnet.fit**
  - a fitted `gcdnet` object for the full data.

- **lambda.min**
  - The optimal value of lambda that gives minimum cross validation error cvm.

- **lambda.1se**
  - The largest value of lambda such that error is within 1 standard error of the minimum.
Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


[http://www.jstatsoft.org/v33/i01/](http://www.jstatsoft.org/v33/i01/)

See Also

gcdnet, plot.cv.gcdnet, predict.cv.gcdnet, and coef.cv.gcdnet methods.

Examples

data(FHT)
set.seed(2011)
cv=cv.gcdnet(FHT$x,FHT$y,
lambda=0.1,pred.loss="loss", nfolds=5)
plot(cv)

FHT

*FHT data introduced in Friedman et al. (2010).*

Description

The FHT data set has n = 50 observations and p = 100 predictors. The covariance between predictors Xj and Xj' has the same correlation 0.5. See details in Friedman et al. (2010).

Usage

data(FHT)

Format

This data frame contains the following columns:

- **x** a matrix with 100 rows and 5000 columns
- **y** class labels
- **y_reg** response variable for regression
gcdnet

References


http://www.jstatsoft.org/v33/i01/

Examples

data(FHT)

<table>
<thead>
<tr>
<th>gcdnet</th>
<th>Fits the regularization paths for large margin classifiers</th>
</tr>
</thead>
</table>

Description

Fits a regularization path for large margin classifiers at a sequence of regularization parameters lambda.

Usage

gcdnet(x, y, nlambda = 100,
method = c("hhsvm", "logit", "sqsvm", "ls"),
lambda.factor = ifelse(nobs < nvars, 0.01, 1e-04),
lambda = NULL, lambda2 = 0,
 pf = rep(1, nvars), pf2 = rep(1, nvars), exclude,
dfmax = nvars + 1, pmax = min(dfmax * 1.2,
 nvars), standardize = TRUE, eps = 1e-8, maxit = 1e6, delta = 2)

Arguments

x matrix of predictors, of dimension N \times p; each row is an observation vector.
y response variable. This argument should be a two-level factor for classification.
nlambda the number of lambda values - default is 100.
method a character string specifying the loss function to use, valid options are:

- "hhsvm" Huberized squared hinge loss,
- "sqsvm" Squared hinge loss,
- "logit" logistic loss,
- "ls" least square loss.
Default is "hhsvm".
lambda.factor

The factor for getting the minimal lambda in lambda sequence, where \( \min(\lambda) = \lambda_{\text{factor}} \times \max(\lambda) \). \( \max(\lambda) \) is the smallest value of \( \lambda \) for which all coefficients are zero. The default depends on the relationship between \( N \) (the number of rows in the matrix of predictors) and \( p \) (the number of predictors). If \( N > p \), the default is 0.0001, close to zero. If \( N < p \), the default is 0.01. A very small value of \( \lambda_{\text{factor}} \) will lead to a saturated fit. It takes no effect if there is user-defined lambda sequence.

lambda

A user supplied lambda sequence. Typically, by leaving this option unspecified users can have the program compute its own lambda sequence based on nlambda and lambda.factor. Supplying a value of lambda overrides this. It is better to supply a decreasing sequence of lambda values than a single (small) value, if not, the program will sort user-defined lambda sequence in decreasing order automatically.

lambda2

Regularization parameter \( \lambda_2 \) for the quadratic penalty of the coefficients.

pf

L1 penalty factor of length \( p \) used for adaptive LASSO or adaptive elastic net. Separate L1 penalty weights can be applied to each coefficient of \( \beta \) to allow differential L1 shrinkage. Can be 0 for some variables, which implies no L1 shrinkage, and results in that variable always being included in the model. Default is 1 for all variables (and implicitly infinity for variables listed in exclude).

pf2

L2 penalty factor of length \( p \) used for adaptive LASSO or adaptive elastic net. Separate L2 penalty weights can be applied to each coefficient of \( \beta \) to allow differential L2 shrinkage. Can be 0 for some variables, which implies no L2 shrinkage. Default is 1 for all variables.

exclude

Indices of variables to be excluded from the model. Default is none. Equivalent to an infinite penalty factor.

dfmax

Limit the maximum number of variables in the model. Useful for very large \( p \), if a partial path is desired. Default is \( p + 1 \).

pmax

Limit the maximum number of variables ever to be nonzero. For example once \( \beta \) enters the model, no matter how many times it exits or re-enters model through the path, it will be counted only once. Default is \( \min(dfmax \times 1.2, p) \).

standardize

Logical flag for variable standardization, prior to fitting the model sequence. If TRUE, x matrix is normalized such that sum squares of each column \( \sum_{i=1}^{N} x_{ij}^2 / N = 1 \). Note that x is always centered (i.e. \( \sum_{i=1}^{N} x_{ij} = 0 \)) no matter standardize is TRUE or FALSE. The coefficients are always returned on the original scale. Default is TRUE.

eps

Convergence threshold for coordinate majorization descent. Each inner coordinate majorization descent loop continues until the relative change in any coefficient (i.e. \( \max_j (\beta_{j}^{\text{new}} - \beta_{j}^{\text{old}})^2 \)) is less than eps. For HH SVM i.e. method="hhsvm", it is \( \frac{2}{\delta} \max_j (\beta_{j}^{\text{new}} - \beta_{j}^{\text{old}})^2 \). Defaults value is 1e-8.

maxit

Maximum number of outer-loop iterations allowed at fixed lambda value. Default is 1e6. If models do not converge, consider increasing maxit.

delta

The parameter \( \delta \) in HH SVM model. Default is 2.
Details

Note that the objective function in `gcdnet` is

\[
\text{Loss}(y, X, \beta)/N + \lambda_1 |\beta| + 0.5 \times \lambda_2 \beta^2
\]

where the penalty is a combination of L1 and L2 term. Users can specify the loss function to use, options include Huberized squared hinge loss, Squared hinge loss, least square loss and logistic regression. Users can also tweak the penalty by choosing different \(\lambda_2\) and penalty factor.

For computing speed reason, if models are not converging or running slow, consider increasing `eps`, decreasing `nlambda`, or increasing `lambdaNfactor` before increasing `maxit`.

FAQ:

**Question:** “I couldn’t get an idea how to specify an option to get adaptive LASSO, how to specify an option to get elastic net and adaptive elastic net? Could you please give me a quick hint?”

**Answer:** \(\lambda_2\) is the regularize parameter for L2 penalty part. To use LASSO, set \(\lambda_2=0\). To use elastic net, set \(\lambda_2\) as nonzero.

\(\text{pf}\) is the L1 penalty factor of length \(p\) (\(p\) is the number of predictors). Separate L1 penalty weights can be applied to each coefficient to allow differential L1 shrinkage. Similiarly \(\text{pfR}\) is the L2 penalty factor of length \(p\).

To use adaptive LASSO, you should set \(\lambda_2=0\) and also specify pf and pf2. To use adaptive elastic net, you should set \(\lambda_2\) as nonzero and specify pf and pf2.

For example

```r
library('gcdnet')

# Dataset N = 100, p = 10
x_log <- matrix(rnorm(100*10),100,10)
y_log <- sample(c(-1,1),100,replace=TRUE)

# LASSO
m <- gcdnet(x=x_log,y=y_log,lambda2=0,method="log")
plot(m)

# elastic net with lambda2 = 1
m <- gcdnet(x=x_log,y=y_log,lambda2=1,method="log")
plot(m)

# adaptive lasso with penalty factor
# pf = 0.5 0.5 0.5 0.5 1.0 1.0 1.0 1.0 1.0 1.0
m <- gcdnet(x=x_log,y=y_log,lambda2=0,method="log", pf=c(rep(0.5,5),rep(1.5)))
plot(m)

# adaptive elastic net with lambda2 = 1 and penalty factor pf = c(rep(0.5,5),rep(1,5))
# pf2 = 3 3 3 3 3 1 1 1 1 1
m <- gcdnet(x=x_log,y=y_log,lambda2=1,method="log",
```
pf=c(rep(0.5,5),rep(1,5)),
pf2 = c(rep(3,5),rep(1,5)))
plot(m)

**Question:** "What is the meaning of the parameter pf? On the package documentation, it said pf is
the penalty weight applied to each coefficient of beta?"

**Answer:** Yes, pf and pf2 are L1 and L2 penalty factor of length p used for adaptive LASSO or
adaptive elastic net. 0 means that the feature (variable) is always excluded, 1 means that the feature
(variable) is included with weight 1.

**Question:** "Does gcdnet deal with both continuous and categorical response variables?"

**Answer:** Yes, both are supported, you can use a continuous type response variable with the least
squares regression loss, or a categorical type response with losses for classification problem.

**Question:** "Why does predict function not work? predict should return the predicted probability of
the positive class. Instead I get:"

Error in as.matrix(as.matrix(cbind2(1, newx))
  error in evaluating the argument 'x' in selecting
a method for function 'as.matrix': Error in t.Call(Csparse_dense_crossprod, y,  
t(x))
  error in evaluating the argument 'x' in selecting
a method for function 't': Error: Cholmod error 'X and/or Y have wrong dimensions'
at file ../MatrixOps/cholmod_sdmult.c, line 90?

"Using the Arcene dataset and executing the following code will give the above error:"

library(gcdnet)
arc <- read.csv("arcene.csv", header=FALSE)
fit <- gcdnet(arc[,1:10001], arc[,10001], standardize=FALSE, method="logit")
pred <- rnorm(10000)
predict(fit, pred, type="link")

**Answer:** It is actually NOT a bug of gcdnet. When make prediction using a new matrix x, each
observation of x should be arranged as a row of a matrix. In your code, because "pred" is a vector,
you need to convert "pred" into a matrix, try the following code:

```
pred <- rnorm(10000)
pred <- matrix(pred,1,10000)
predict(fit, pred, type="link")
```

**Value**

An object with S3 class `gcdnet`.

`call` the call that produced this object

`b0` intercept sequence of length `length(lambda)`

`beta` a p*length(`lambda`) matrix of coefficients, stored as a sparse matrix (dgCMatrix
class, the standard class for sparse numeric matrices in the `Matrix` package.). To
convert it into normal type matrix use `as.matrix()`.
**plot.cv.gcdnet**

the actual sequence of lambda values used

the number of nonzero coefficients for each value of lambda.

dimension of coefficient matrix (ices)

total number of iterations (the most inner loop) summed over all lambda values

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


See Also

plot.gcdnet

Examples

data(FHT)

# solution paths for the least squares
m0 <- gcdnet(x=FHT$x,y=FHT$y_reg,lambda2=1,method="ls")
plot(m0)

# solution paths for the HH SVM
m1 <- gcdnet(x=FHT$x,y=FHT$y,delta=1,lambda2=1,method="hhsvm")
plot(m1)

# solution paths for the penalized SVM with the squared hinge loss
m2 <- gcdnet(x=FHT$x,y=FHT$y,lambda2=0.1,method="sqsvm")
plot(m2)

# solution paths for the penalized logistic regression
m3 <- gcdnet(x=FHT$x,y=FHT$y,lambda2=0.01,method="logit")
plot(m3)

---

**Description**

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the lambda values used. This function is modified based on the `plot.cv` function from the glmnet package.
Usage

```r
## S3 method for class 'cv.gcdnet'
plot(x, sign.lambda, ...)
```

Arguments

- `x`: fitted `cv.gcdnet` object
- `sign.lambda`: either plot against log(lambda) (default) or its negative if `sign.lambda=-1`.
- `...`: other graphical parameters to plot

Details

A plot is produced.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


[http://www.jstatsoft.org/v33/i01/](http://www.jstatsoft.org/v33/i01/)

See Also

cv.gcdnet.

Examples

data(FHT)
seset.seed(2011)
cv=cv.gcdnet(FHT$x, FHT$y,
lambda2 = 1, pred.loss="loss", nfolds=5)
plot(cv)
plot.gcdnet

**Plot coefficients from a "gcdnet" object**

**Description**

Produces a coefficient profile plot of the coefficient paths for a fitted `gcdnet` object. This function is modified based on the `plot` function from the `glmnet` package.

**Usage**

```r
## S3 method for class 'gcdnet'
plot(x, xvar = c("norm", "lambda"), color = FALSE, label = FALSE, ...)
```

**Arguments**

- `x`: fitted `gcdnet` model
- `xvar`: what is on the X-axis. "norm" plots against the L1-norm of the coefficients, "lambda" against the log-lambda sequence.
- `color`: if TRUE, plot the curves with rainbow colors. FALSE is gray colors. Default is FALSE
- `label`: if TRUE, label the curves with variable sequence numbers. Default is FALSE
- `...`: other graphical parameters to plot

**Details**

A coefficient profile plot is produced.

**Author(s)**

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

**References**


predict.cv.gcdnet

Description

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

Usage

## S3 method for class 'cv.gcdnet'
predict(object, newx, s=c("lambda.1se","lambda.min"),...)

Arguments

- **object**: fitted cv.gcdnet object.
- **newx**: matrix of new values for x at which predictions are to be made. Must be a matrix. See documentation for predict.gcdnet.
- **s**: value(s) of the penalty parameter lambda at which predictions are required. Default is the value s="lambda.1se" stored on the CV object. Alternatively s="lambda.min" can be used. If s is numeric, it is taken as the value(s) of lambda to be used.
- **...**: 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 gcdnet objects.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>
References


[http://www.jstatsoft.org/v33/i01/](http://www.jstatsoft.org/v33/i01/)

See Also

`cv.gcdnet`, and `coef.cv.gcdnet` methods.

Examples

```r
data(FHT)
set.seed(2011)
cv=cv.gcdnet(FHT$x, FHT$y,
lambda2 = 1, pred.loss="misclass",
lambda.factor=0.05,nfolds=5)
pred = predict(cv$gcdnet.fit, newx = FHT$x,
s = cv$lambda.1se, type = "class")
```

```
predict.gcdnet
make predictions from a "gcdnet" object.
```

Description

Similar to other predict methods, this functions predicts fitted values and class labels from a fitted `gcdnet` object.

Usage

```r
## S3 method for class 'gcdnet'
predict(object, newx, s = NULL,
type=c("class","link"), ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>fitted <code>gcdnet</code> model object.</td>
</tr>
<tr>
<td>newx</td>
<td>matrix of new values for x at which predictions are to be made. NOTE: newx must be a matrix, predict function does not accept a vector or other formats of codenewx.</td>
</tr>
<tr>
<td>s</td>
<td>value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.</td>
</tr>
<tr>
<td>type</td>
<td>type of prediction required.</td>
</tr>
</tbody>
</table>
• Type "link" gives the linear predictors for classification problems and gives predicted response for regression problems.

• Type "class" produces the class label corresponding to the maximum probability. Only available for classification problems.

... Not used. Other arguments to predict.

Details

$s$ is the new vector at which predictions are requested. If $s$ is not in the lambda sequence used for fitting the model, the `predict` function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right lambda indices.

Value

The object returned depends on type.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


[http://www.jstatsoft.org/v33/i01/](http://www.jstatsoft.org/v33/i01/)

See Also

`coef` method

Examples

data(FHT)
m1 = gcdnet(x=FHT$x, y=FHT$y)
print(predict(m1, type="class", newx=FHT$x[2:5,]))
print.gcdnet

print.gcdnet  print a gcdnet object

Description

Print a summary of the gcdnet path at each step along the path.

Usage

```r
## S3 method for class 'gcdnet'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

- `x`: fitted `gcdnet` object
- `digits`: significant digits in printout
- `...`: additional print arguments

Details

The call that produced the `gcdnet` object is printed, followed by a two-column matrix with columns `df` and `lambda`. The `df` column is the number of nonzero coefficients.

Value

A two-column matrix, the first column is the number of nonzero coefficients and the second column is `lambda`.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yiyang@umn.edu>

References


Examples

```r
data(FHT)
m1 <- gcdnet(x=FHT$x,y=FHT$y,delta=1,lambda2=0.1)
print(m1)
```
Index

*Topic **datasets**
  - FHT, 6

*Topic **models**
  - coef.cv.gcdnet, 2
  - coef.gcdnet, 3
  - cv.gcdnet, 4
  - gcdnet, 7
  - plot.cv.gcdnet, 11
  - plot.gcdnet, 13
  - predict.cv.gcdnet, 14
  - predict.gcdnet, 15
  - print.gcdnet, 17

*Topic **regression**
  - coef.cv.gcdnet, 2
  - coef.gcdnet, 3
  - cv.gcdnet, 4
  - gcdnet, 7
  - plot.cv.gcdnet, 11
  - plot.gcdnet, 13
  - predict.cv.gcdnet, 14
  - predict.gcdnet, 15
  - print.gcdnet, 17

  coef, 16
  coef.cv.gcdnet, 2, 6, 15
  coef.gcdnet, 3
  coef.hsvmpath(coef.gcdnet), 3
  coef.logitpath(coef.gcdnet), 3
  coef.lspath(coef.gcdnet), 3
  coef.sqsvmpath(coef.gcdnet), 3
  cv.gcdnet, 2, 3, 4, 5, 12, 14, 15
  cv.hsvmpath(cv.gcdnet), 4
  cv.logitpath(cv.gcdnet), 4
  cv.lspath(cv.gcdnet), 4
  cv.sqsvmpath(cv.gcdnet), 4

  FHT, 6
  gcdnet, 2, 3, 5, 6, 7, 10, 13–15, 17
  plot.cv.gcdnet, 6, 11

plot.gcdnet, 13
predict, 2, 14
predict.cv.gcdnet, 3, 6, 14
predict.gcdnet, 4, 15
predict.hsvmpath(predict.gcdnet), 15
predict.logitpath(predict.gcdnet), 15
predict.lspath(predict.gcdnet), 15
predict.sqsvmpath(predict.gcdnet), 15
print.gcdnet, 17