Package: gcdnet (via r-universe)

October 12, 2026

Title The (Adaptive) LASSO and Elastic Net Penalized Least Squares, Logistic Regression, Hybrid Huberized Support Vector Machines, Squared Hinge Loss Support Vector Machines and Expectile Regression using a Fast Generalized Coordinate Descent Algorithm

Version 1.0.6

Author Yi Yang <yi.yang6@mcgill.ca>, Yuwen Gu <yuwen.gu@uconn.edu>, Hui Zou <hzou@stat.umn.edu>

Maintainer Yi Yang <yi.yang6@mcgill.ca>

Imports grDevices, graphics, stats, methods, Matrix

Description Implements a generalized coordinate descent (GCD) algorithm for computing the solution paths of the hybrid Huberized support vector machine (HHSVM) and its generalizations. Supported models include the (adaptive) LASSO and elastic net penalized least squares, logistic regression, HHSVM, squared hinge loss SVM and expectile regression.

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/emeryyi/gcdnet

RoxygenNote 7.2.0

Suggests testthat

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2022-08-14 01:30:02 UTC

RemoteUrl https://github.com/cran/gcdnet

RemoteRef HEAD

RemoteSha 4e14d2e32651deff9ee005e3230cfc53b7e2fef1

Contents

coef . . . 2
coef.cv.gcdnet . . . 3
coef.gcdnet . . . 4
cv.gcdnet . . . 6
FHT . . . 8
gcdnet . . . 9
plot.cv.gcdnet . . . 14
plot.gcdnet . . . 16
predict . . . 17
predict.cv.gcdnet . . . 18
predict.gcdnet . . . 19
print.gcdnet . . . 20

Index **22**

`coef` *Extract Model Coefficients*

Description

`coef` is a generic function which extracts model coefficients from objects returned by modeling functions. `coefficients` is an *alias* for it.

Usage

```
coef(object, ...)
```

Arguments

`object`	an object for which the extraction of model coefficients is meaningful.
`...`	other arguments.

Value

Coefficients extracted from the model object `object`.

See Also

`coef.gcdnet`, `coef.erpath`, `coef.lspath`, `coef.hsvmpath`, `coef.logitpath`, `coef.sqsvmpath`.

`coef.cv.gcdnet` *Get coefficients or make coefficient predictions from a "cv.gcdnet" object.*

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

```
## 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, Yuwen Gu and Hui Zou

Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

See Also

cv.gcdnet, and predict.cv.gcdnet methods.

Examples

```
data(FHT)
set.seed(2011)
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 a "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

```
## 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, Yuwen Gu and Hui Zou

Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://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")
```

cv.gcdnet *Cross-validation for gcdnet*

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 = NULL,
  pred.loss = c("misclass", "loss"),
  nfolds = 5,
  foldid,
  delta = 2,
  omega = 0.5,
  ...
)
```

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.
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). When use expectile regression loss "er", it gives asymmetric mean square error (AMSE). • "misclass" only available for classification: it gives misclassification error. Default is "loss".
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.
delta	parameter δ only used in HHSVM for computing margin based loss function, only available for pred.loss = "loss".
omega	parameter ω only used in expectile regression. Only available for 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, Yuwen Gu and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

See Also

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

Examples

```
# fit an elastic net penalized HHSVM with lambda2 = 0.1 for the L2 penalty.
# Use the misclassification rate as the cross validation prediction loss.
# Use five-fold CV to choose the optimal lambda for the L1 penalty.

data(FHT)
set.seed(2011)
cv <- cv.gcdnet(FHT$x, FHT$y, method = "hhsvm",
                lambda2 = 0.1, pred.loss = "misclass",
                nfolds = 5, delta = 1.5)
plot(cv)

# fit an elastic net penalized least squares
# with lambda2 = 0.1 for the L2 penalty. Use the
# least square loss as the cross validation
# prediction loss. Use five-fold CV to choose
# the optimal lambda for the L1 penalty.

set.seed(2011)
cv1 <- cv.gcdnet(FHT$x, FHT$y_reg, method ="ls",
                 lambda2 = 0.1, pred.loss = "loss",
                 nfolds = 5)
plot(cv1)

# To fit a LASSO penalized logistic regression
# we set lambda2 = 0 to disable the L2 penalty. Use the
# logistic loss as the cross validation
# prediction loss. Use five-fold CV to choose
# the optimal lambda for the L1 penalty.

set.seed(2011)
cv2 <- cv.gcdnet(FHT$x, FHT$y, method ="logit",
                 lambda2 = 0, pred.loss="loss",
                 nfolds=5)
plot(cv2)
```

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).

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

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

Examples

```
data(FHT)
```

`gcdnet` *Fits the regularization paths for large margin classifiers*

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", "er"),
  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 = FALSE,
  intercept = TRUE,
  eps = 1e-08,
```

```
  maxit = 1e+06,
  delta = 2,
  omega = 0.5
)
```

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. • "er" expectile regression loss. Default is "hhsvm".
lambda.factor	The factor for getting the minimal lambda in lambda sequence, where min(lambda) = lambda.factor * max(lambda), where 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.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 λ_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 β 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 β 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 β 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*1.2,p)`.
standardize	logical flag for variable standardization, prior to fitting the model sequence. If `TRUE`, `x` matrix is normalized such that `x` is centered (i.e. $\sum_{i=1}^{N} x_{ij} = 0$), and sum squares of each column $\sum_{i=1}^{N} x_{ij}^2/N = 1$. If `x` matrix is standardized, the ending coefficients will be transformed back to the original scale. Default is `FALSE`.
intercept	logical flag to indicate whether to include or exclude the intercept in the model.
eps	convergence threshold for coordinate majorization descent. Each inner coordinate majorization descent loop continues until the relative change in any coefficient (i.e., $\max_j(\beta_j^{new}-\beta_j^{old})^2$) is less than `eps`. For HHSVM, i.e., `method="hhsvm"`, it is $\frac{2}{\delta}\max_j(\beta_j^{new}-\beta_j^{old})^2$. For expectile regression, i.e., `method="er"`, it is $2\max(1-\omega,\omega)\max_j(\beta_j^{new}-\beta_j^{old})^2$. 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 δ in the HHSVM model. The value must be greater than 0. Default is 2.
omega	the parameter ω in the expectile regression model. The value must be in (0,1). Default is 0.5.

Details

Note that the objective function in `gcdnet` is

$$Loss(y, X, \beta)/N + \lambda_1\|\beta\|_1 + 0.5\lambda_2\|\beta\|_2^2$$

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, logistic regression and expectile regression loss. Users can also tweak the penalty by choosing different $lambda2$ and penalty factor.

For computing speed reason, if models are not converging or running slow, consider increasing `eps`, decreasing `nlambda`, or increasing `lambda.factor` 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: `lambda2` is the regularize parameter for L2 penalty part. To use LASSO, set `lambda2=0`. To use elastic net, set `lambda2` as nonzero.

`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 `pf2` is the L2 penalty factor of length p.

To use adaptive LASSO, you should set `lambda2=0` and also specify `pf` and `pf2`. To use adaptive elastic net, you should set `lambda2` as nonzero and specify `pf` and `pf2`,

For example:

```
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 0.5 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)) %*% nbeta):
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[,-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().
lambda	the actual sequence of lambda values used
df	the number of nonzero coefficients for each value of lambda.
dim	dimension of coefficient matrix (ices)
npasses	total number of iterations (the most inner loop) summed over all lambda values
jerr	error flag, for warnings and errors, 0 if no error.

Author(s)

Yi Yang, Yuwen Gu and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

See Also

plot.gcdnet

Examples

```
data(FHT)
# 1. solution paths for the LASSO penalized least squares.
# To use LASSO set lambda2 = 0.

m1 <- gcdnet(x = FHT$x, y = FHT$y_reg, lambda2 = 0, method = "ls")
plot(m1)

# 2. solution paths for the elastic net penalized HHSVM.
# lambda2 is the parameter controlling the L2 penalty.
m2 <- gcdnet(x = FHT$x, y = FHT$y, delta = 1, lambda2 = 1, method = "hhsvm")
plot(m2)

# 3. solution paths for the adaptive LASSO penalized SVM
# with the squared hinge loss. To use the adaptive LASSO,
# set lambda2 = 0 and meanwhile specify the L1 penalty weights.
p <- ncol(FHT$x)
# set the first three L1 penalty weights as 0.1 and the rest are 1
pf = c(0.1, 0.1, 0.1, rep(1, p-3))
m3 <- gcdnet(x = FHT$x, y = FHT$y, pf = pf, lambda2 = 0, method = "sqsvm")
plot(m3)

# 4. solution paths for the adaptive elastic net penalized
# logistic regression.

p <- ncol(FHT$x)
# set the first three L1 penalty weights as 10 and the rest are 1.
pf <- c(10, 10, 10, rep(1, p-3))
# set the last three L2 penalty weights as 0.1 and the rest are 1.
pf2 <- c(rep(1, p-3), 0.1, 0.1, 0.1)
# set the L2 penalty parameter lambda2=0.01.
m4 <- gcdnet(x = FHT$x, y = FHT$y, pf = pf, pf2 = pf2,
             lambda2 = 0.01, method = "logit")
plot(m4)

# 5. solution paths for the LASSO penalized expectile regression
# with the asymmetric least square parameter omega=0.9.

m5 <- gcdnet(x = FHT$x, y = FHT$y_reg, omega = 0.9,
             lambda2 = 0, method = "er")
plot(m5)
```

---

plot.cv.gcdnet *Plot the cross-validation curve produced by cv.gcdnet*

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

```
## S3 method for class 'cv.gcdnet'
plot(x, sign.lambda = 1, ...)
```

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, Yuwen Gu and Hui Zou

Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

See Also

`cv.gcdnet`.

Examples

```
# fit an elastic net penalized logistic regression with lambda2 = 1 for the
# L2 penalty. Use the logistic loss as the cross validation prediction loss.
# Use five-fold CV to choose the optimal lambda for the L1 penalty.
data(FHT)
set.seed(2011)
cv=cv.gcdnet(FHT$x, FHT$y, method ="logit", 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

```
## 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, Yuwen Gu and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

Examples

```
data(FHT)
m1 <- gcdnet(x = FHT$x,y = FHT$y)
par(mfrow = c(1,3))
plot(m1) # plots against the L1-norm of the coefficients
plot(m1,xvar = "lambda",label = TRUE) # plots against the log-lambda sequence
plot(m1,color = TRUE)
```

`predict` *Model predictions*

Description

`predict` is a generic function for predictions from the results of various model fitting functions. The function invokes particular *methods* which depend on the `class` of the first argument.

Usage

```
predict(object, ...)
```

Arguments

`object`	a model object for which prediction is desired.
`...`	additional arguments affecting the predictions produced.

Value

The form of the value returned by `predict` depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

`predict.gcdnet`, `predict.erpath`, `predict.lspath`, `predict.hsvmpath`, `predict.logitpath`, `predict.sqsvmpath`.

`predict.cv.gcdnet` *Make predictions from a "cv.gcdnet" object.*

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, Yuwen Gu and Hui Zou

Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

See Also

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

Examples

```
data(FHT)
set.seed(2011)
cv=cv.gcdnet(FHT$x, FHT$y, lambda2 = 1, pred.loss="misclass",
             lambda.factor=0.05, nfolds=5)
pre = 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

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

Arguments

object	fitted gcdnet model object.
newx	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 newx.
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 of prediction required. • 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, Yuwen Gu and Hui Zou

Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://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 a gcdnet object*

Description

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

Usage

```
## 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 columns is the number of nonzero coefficients and the second column is Lambda.

Author(s)

Yi Yang, Yuwen Gu and Hui Zou

Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2012). "An Efficient Algorithm for Computing The HHSVM and Its Generalizations." *Journal of Computational and Graphical Statistics*, 22, 396-415.
BugReport: https://github.com/emeryyi/gcdnet

Gu, Y., and Zou, H. (2016). "High-dimensional generalizations of asymmetric least squares regression and their applications." *The Annals of Statistics*, 44(6), 2661–2694.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software, 33, 1.*
https://www.jstatsoft.org/v33/i01/

Examples

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

Index

∗ **datasets**
FHT, 8
∗ **models**
coef.cv.gcdnet, 3
coef.gcdnet, 4
cv.gcdnet, 6
gcdnet, 9
plot.cv.gcdnet, 14
plot.gcdnet, 16
predict.cv.gcdnet, 18
predict.gcdnet, 19
print.gcdnet, 20
∗ **regression**
coef.cv.gcdnet, 3
coef.gcdnet, 4
cv.gcdnet, 6
gcdnet, 9
plot.cv.gcdnet, 14
plot.gcdnet, 16
predict.cv.gcdnet, 18
predict.gcdnet, 19
print.gcdnet, 20

class, *17*
coef, 2, *20*
coef.cv.gcdnet, 3, *7*, *19*
coef.erpath, *2*
coef.erpath *(*coef.gcdnet*)*, 4
coef.gcdnet, *2*, 4
coef.hsvmpath, *2*
coef.hsvmpath *(*coef.gcdnet*)*, 4
coef.logitpath, *2*
coef.logitpath *(*coef.gcdnet*)*, 4
coef.lspath, *2*
coef.lspath *(*coef.gcdnet*)*, 4
coef.sqsvmpath, *2*
coef.sqsvmpath *(*coef.gcdnet*)*, 4
cv.erpath *(*cv.gcdnet*)*, 6
cv.gcdnet, *3, 4*, 6, *7*, *15*, *18, 19*
cv.hsvmpath *(*cv.gcdnet*)*, 6
cv.logitpath *(*cv.gcdnet*)*, 6
cv.lspath *(*cv.gcdnet*)*, 6
cv.sqsvmpath *(*cv.gcdnet*)*, 6

FHT, 8

gcdnet, *3, 4*, *6, 7*, 9, *13*, *16*, *18, 19*, *21*

plot.cv.gcdnet, *7*, 14
plot.gcdnet, 16
predict, *3*, 17, *18*
predict.cv.gcdnet, *4*, *7*, 18
predict.erpath, *17*
predict.erpath *(*predict.gcdnet*)*, 19
predict.gcdnet, *5*, *17*, 19
predict.hsvmpath, *17*
predict.hsvmpath *(*predict.gcdnet*)*, 19
predict.logitpath, *17*
predict.logitpath *(*predict.gcdnet*)*, 19
predict.lspath, *17*
predict.lspath *(*predict.gcdnet*)*, 19
predict.sqsvmpath, *17*
predict.sqsvmpath *(*predict.gcdnet*)*, 19
print.gcdnet, 20