Package 'hdsvm'

May 8, 2026

Type Package

Title Fast Algorithm for Support Vector Machine

Version 1.0.2

Date 2025-09-25

Maintainer Yikai Zhang <yikai-zhang@uiowa.edu>

Description Implements an efficient algorithm for fitting the entire regularization path of support vector machine models with elastic-net penalties using a generalized coordinate descent scheme. The framework also supports SCAD and MCP penalties. It is designed for high-dimensional datasets and emphasizes numerical accuracy and computational efficiency. This package implements the algorithms proposed in Tang, Q., Zhang, Y., & Wang, B. (2022) <https://openreview.net/pdf?id=RvwMTDYTOb>.

License GPL-2

Encoding UTF-8

Depends R (>= 3.5.0)

Imports stats, Matrix, methods

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 7.2.3

Author Yikai Zhang [aut, cre],
Qian Tang [aut],
Boxiang Wang [aut]

Repository CRAN

Date/Publication 2025-09-26 15:40:02 UTC

Contents

`coef.cv.hdsvm` *Extract Coefficients from a 'cv.hdsvm' Object*

Description

Retrieves coefficients from a cross-validated 'hdsvm()' model, using the stored '"hdsvm.fit"' object and the optimal 'lambda' value determined during cross-validation.

Usage

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

Arguments

`object`	A fitted 'cv.hdsvm()' object from which coefficients are to be extracted.
`s`	Specifies the value(s) of the penalty parameter 'lambda' for which coefficients are desired. The default is 's = "lambda.1se"', which corresponds to the largest value of 'lambda' such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, 's = "lambda.min"' can be used, corresponding to the minimum of the cross-validation error estimate. If 's' is numeric, these are taken as the actual values of 'lambda' to use.
`...`	Not used.

Value

Returns the coefficients at the specified 'lambda' values.

See Also

`cv.hdsvm`, `predict.cv.hdsvm`

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
cv.fit <- cv.hdsvm(x, y, lam2 = 0.01)
coef(cv.fit, s = c(0.02, 0.03))
```

`coef.cv.nc.hdsvm` *Extract Coefficients from a 'cv.nc.hdsvm' Object*

Description

Retrieves coefficients at specified values of 'lambda' from a fitted 'cv.nc.hdsvm()' model. Utilizes the stored '"nchdsvm.fit"' object and the optimal 'lambda' values determined during the cross-validation process.

Usage

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

Arguments

`object`	A fitted 'cv.nc.hdsvm()' object from which coefficients are to be extracted.
`s`	Specifies the 'lambda' values at which coefficients are requested. The default is 's = "lambda.1se"', representing the largest 'lambda' such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, 's = "lambda.min"' corresponds to the 'lambda' yielding the minimum cross-validation error. If 's' is numeric, these values are directly used as the 'lambda' values for coefficient extraction.
`...`	Not used.

Value

Returns a vector or matrix of coefficients corresponding to the specified 'lambda' values.

See Also

`cv.nc.hdsvm`, `predict.cv.nc.hdsvm`

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out = 30))
cv.nc.fit <- cv.nc.hdsvm(x = x, y = y, lambda = lambda, lam2 = lam2, pen = "scad")
coef(cv.nc.fit, s = c(0.02, 0.03))
```

`coef.hdsvm` *Extract Model Coefficients from a 'hdsvm' Object*

Description

Retrieves the coefficients at specified values of 'lambda' from a fitted 'hdsvm()' model.

Usage

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

Arguments

`object`	Fitted 'hdsvm()' object.
`s`	Values of the penalty parameter 'lambda' for which coefficients are requested. Defaults to the entire sequence used during the model fit.
`type`	Type of prediction required. 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`.
`...`	Not used.

Details

This function extracts coefficients for specified 'lambda' values from a 'hdsvm()' object. If 's', the vector of 'lambda' values, contains values not originally used in the model fitting, the 'coef' function employs linear interpolation between the closest 'lambda' values from the original sequence to estimate coefficients at the new 'lambda' values.

Value

Returns a matrix or vector of coefficients corresponding to the specified 'lambda' values.

See Also

hdsvm, predict.hdsvm

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
fit <- hdsvm(x, y, lam2=lam2)
coefs <- coef(fit, s = fit$lambda[3:5])
```

---

coef.nc.hdsvm *Extract Model Coefficients from a 'nc.hdsvm' Object*

Description

Retrieves the coefficients at specified values of 'lambda' from a fitted 'nc.hdsvm()' model.

Usage

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

Arguments

object	Fitted 'nc.hdsvm()' object.
s	Values of the penalty parameter 'lambda' for which coefficients are requested. Defaults to the entire sequence used during the model fit.
type	Type of prediction required. 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.
...	Not used.

Details

This function extracts coefficients for specified 'lambda' values from a 'nc.hdsvm()' object. If 's', the vector of 'lambda' values, contains values not originally used in the model fitting, the 'coef' function employs linear interpolation between the closest 'lambda' values from the original sequence to estimate coefficients at the new 'lambda' values.

Value

Returns a matrix or vector of coefficients corresponding to the specified ‘lambda‘ values.

See Also

nc.hdsvm, predict.nc.hdsvm

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out = 30))
nc.fit <- nc.hdsvm(x = x, y = y, lambda = lambda, lam2 = lam2, pen = "scad")
nc.coefs <- coef(nc.fit, s = nc.fit$lambda[3:5])
```

---

cv.hdsvm *Cross-validation for Selecting the Tuning Parameter in the Penalized SVM*

Description

Performs k-fold cross-validation for hdsvm.

Usage

```
cv.hdsvm(x, y, lambda = NULL, nfolds = 5L, foldid, ...)
```

Arguments

x	A numerical matrix with n rows (observations) and p columns (variables).
y	Response variable.
lambda	Optional; a user-supplied sequence of lambda values. If NULL, hdsvm selects its own sequence.
nfolds	Number of folds for cross-validation. Defaults to 5.
foldid	Optional vector specifying the indices of observations in each fold. If provided, it overrides nfolds.
...	Additional arguments passed to hdsvm.

Details

This function computes the average cross-validation error and provides the standard error.

Value

An object with S3 class `cv.hdsvm` consisting of

`lambda`	Candidate `lambda` values.
`cvm`	Mean cross-validation error.
`cvsd`	Standard error of the mean cross-validation error.
`cvup`	Upper confidence curve: `cvm` + `cvsd`.
`cvlo`	Lower confidence curve: `cvm` - `cvsd`.
`lambda.min`	`lambda` achieving the minimum cross-validation error.
`lambda.1se`	Largest `lambda` within one standard error of the minimum error.
`cv.min`	Cross-validation error at `lambda.min`.
`cv.1se`	Cross-validation error at `lambda.1se`.
`hdsvm.fit`	a fitted `hdsvm` object for the full data.
`nzero`	Number of non-zero coefficients at each `lambda`.

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
fit <- cv.hdsvm(x, y, lam2=lam2)
```

`cv.nc.hdsvm` *Cross-validation for Selecting the Tuning Parameter of Nonconvex Penalized SVM*

Description

Conducts k-fold cross-validation for the `nc.hdsvm` function.

Usage

```
cv.nc.hdsvm(x, y, lambda = NULL, nfolds = 5L, foldid, ...)
```

Arguments

x	A numerical matrix with dimensions (n rows and p columns), where each row represents an observation.
y	Response variable.
lambda	Optional user-supplied sequence of lambda values.
nfolds	Number of folds in the cross-validation, default is 5.
foldid	An optional vector that assigns each observation to a specific fold. If provided, this parameter overrides nfolds.
...	Additional arguments passed to nc.hdsvm.

Details

This function estimates the average cross-validation error and its standard error across folds. It is primarily used to identify the optimal lambda value for fitting nonconvex penalized SVM models.

Value

An object of class cv.nc.hdsvm is returned, which is a list with the ingredients of the cross-validated 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).
nchdsvm.fit	a fitted nc.hdsvm 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.

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out=30))
cv.nc.fit <- cv.nc.hdsvm(x=x, y=y, lambda=lambda, lam2=lam2, pen="scad")
```

hdsvm *Solve Penalized SVM*

Description

Fits a penalized support vector machine (SVM) model using a range of lambda values, allowing for detailed control over regularization parameters and model complexity.

Usage

```
hdsvm(
  x,
  y,
  nlambda = 100,
  lambda.factor = ifelse(nobs < nvars, 0.01, 1e-04),
  lambda = NULL,
  lam2 = 0,
  hval = 1,
  pf = rep(1, nvars),
  pf2 = rep(1, nvars),
  exclude,
  dfmax = nvars + 1,
  pmax = min(dfmax * 1.2, nvars),
  standardize = TRUE,
  eps = 1e-08,
  maxit = 1e+06,
  sigma = 0.9,
  is_exact = FALSE
)
```

Arguments

x	Matrix of predictors, with dimensions (n observations by p variables).
y	Response variable vector of length n.
nlambda	Number of lambda values to consider (default is 100).
lambda.factor	The factor for getting the minimal value in the lambda sequence, where min(lambda) = lambda.factor * max(lambda) and max(lambda) is the smallest value of lambda for which all coefficients (except the intercept when it is present) are penalized to zero. The default depends on the relationship between n (the number of rows in the design matrix) and p (the number of predictors). If $n < p$, it defaults to 0.05. If $n > p$, the default is 0.001, closer to zero. A very small value of lambda.factor will lead to a saturated fit. The argument takes no effect if there is a user-supplied 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. It is better to supply, if necessary, a decreasing sequence

	of lambda values than a single (small) value. The program will ensure that the user-supplied lambda sequence is sorted in decreasing order before
lam2	Regularization parameter lambda2 for the quadratic penalty of the coefficients. Unlike lambda, only one value of lambda2 is used for each fitting process.
hval	Smoothing parameter for the smoothed hinge loss, default is 1.
pf	L1 penalty factor of length p used for the adaptive LASSO or adaptive elastic net. Separate L1 penalty weights can be applied to each coefficient to allow different L1 shrinkage. Can be 0 for some variables (but not all), which imposes no shrinkage, and results in that variable always being included in the model. Default is 1 for all variables (and implicitly infinity for variables in the exclude list).
pf2	L2 penalty factor of length p used for adaptive elastic net. Separate L2 penalty weights can be applied to each coefficient to allow different L2 shrinkage. Can be 0 for some variables, which imposes no 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	The maximum number of variables allowed in the model. Useful for very large p when a partial path is desired. Default is $p + 1$.
pmax	Maximum count of non-zero coefficients across the solution path.
standardize	Logical flag for variable standardization, prior to fitting the model sequence. The coefficients are always returned to the original scale. Default is TRUE.
eps	Convergence criterion for stopping the algorithm.
maxit	Maximum number of iterations permitted.
sigma	Penalty parameter in the quadratic term of the augmented Lagrangian.
is_exact	If TRUE, solutions are computed exactly; otherwise, approximations are used.

Details

The function utilizes the hinge loss function combined with elastic net penalization:

$$1'[\max\{1 - y_i(\beta_0 + X_i^\top \beta), 0\}]/N + \lambda_1 \cdot |pf_1 \circ \beta|_1 + 0.5 \cdot \lambda_2 \cdot (\sqrt{pf_2} \circ \beta)^2,$$

where $\circ$ denotes the Hadamard product.

For faster computation, if the algorithm is not converging or running slow, consider increasing eps, increasing sigma, decreasing nlambda, or increasing lambda.factor before increasing maxit.

Value

An object with S3 class hdsvm consisting of

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.
npasses	the number of iterations for every lambda value
jerr	error flag, for warnings and errors, 0 if no error.

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
fit <- hdsvm(x, y, lam2=lam2)
```

---

nc.hdsvm *Solve the Penalized SVM with Nonconvex Penalties*

Description

This function fits the penalized SVM using nonconvex penalties such as SCAD or MCP. It allows for flexible control over the regularization parameters and offers advanced options for initializing and optimizing the fit.

Usage

```
nc.hdsvm(
  x,
  y,
  lambda,
  pen = "scad",
  aval = NULL,
  lam2 = 1,
  ini_beta = NULL,
  lla_step = 3,
  ...
)
```

Arguments

x	Matrix of predictors, with dimensions (nobs * nvars); each row represents an observation.
y	Response variable, with length n.

lambda	Optional user-supplied sequence of lambda values. If unspecified, the program calculates its own sequence based on nlambda and lambda.factor. Supplying a decreasing sequence of lambda values is advisable to leverage the warm-start optimization.
pen	Specifies the type of nonconvex penalty: "SCAD" or "MCP".
aval	The parameter value for the SCAD or MCP penalty. Default is 3.7 for SCAD and 2 for MCP.
lam2	Regularization parameter lambda2 for the quadratic penalty on the coefficients. Only one value of lambda2 is used per fit.
ini_beta	Optional initial coefficients to start the fitting process.
lla_step	Number of Local Linear Approximation (LLA) steps. Default is 3.
...	Additional arguments passed to hdsvm.

Value

An object with S3 class nc.hdsvm consisting of

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.
npasses	the number of iterations for every lambda value
jerr	error flag, for warnings and errors, 0 if no error.

#'

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out = 30))
nc.fit <- nc.hdsvm(x = x, y = y, lambda = lambda, lam2 = lam2, pen = "scad")
```

predict.cv.hdsvm *Make Predictions from a 'cv.hdsvm' Object*

Description

Generates predictions using a fitted 'cv.hdsvm()' object. This function utilizes the stored 'hdsvm.fit' object and an optimal value of 'lambda' determined during the cross-validation process.

Usage

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

Arguments

object	A fitted 'cv.hdsvm()' object from which predictions are to be made.
newx	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
s	Specifies the value(s) of the penalty parameter 'lambda' at which predictions are desired. The default is 's = "lambda.1se"', representing the largest value of 'lambda' such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, 's = "lambda.min"' can be used, corresponding to the minimum of the cross-validation error estimate. If 's' is numeric, these are taken as the actual values of 'lambda' to use for predictions.
type	Type of prediction required. Type '"class"' produces the predicted binary class labels and type '"loss"' returns the fitted values. Default is "class".
...	Not used.

Value

Returns a matrix or vector of predicted values corresponding to the specified 'lambda' values.

See Also

cv.hdsvm, coef.cv.hdsvm

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
cv.fit <- cv.hdsvm(x, y, lam2 = 0.01)
predict(cv.fit, newx = x[50:60, ], s = "lambda.min")
```

`predict.cv.nc.hdsvm` *Make Predictions from a 'cv.nc.hdsvm' Object*

Description

Generates predictions using a fitted 'cv.nc.hdsvm()' object. This function utilizes the stored 'nchdsvm.fit' object and an optimal value of 'lambda' determined during the cross-validation process.

Usage

```
## S3 method for class 'cv.nc.hdsvm'
predict(
  object,
  newx,
  s = c("lambda.1se", "lambda.min"),
  type = c("class", "loss"),
  ...
)
```

Arguments

`object`	A fitted 'cv.nc.hdsvm()' object from which predictions are to be made.
`newx`	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
`s`	Specifies the value(s) of the penalty parameter 'lambda' at which predictions are desired. The default is 's = "lambda.1se"', representing the largest value of 'lambda' such that the cross-validation error estimate is within one standard error of the minimum. Alternatively, 's = "lambda.min"' can be used, corresponding to the minimum of the cross-validation error estimate. If 's' is numeric, these are taken as the actual values of 'lambda' to use for predictions.
`type`	Type of prediction required. Type '"class"' produces the predicted binary class labels and type '"loss"' returns the fitted values. Default is `"class"`.
`...`	Not used.

Value

Returns a matrix or vector of predicted values corresponding to the specified ‘lambda‘ values.

See Also

cv.nc.hdsvm, predict.cv.nc.hdsvm

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out = 30))
cv.nc.fit <- cv.nc.hdsvm(x = x, y = y, lambda = lambda, lam2 = lam2, pen = "scad")
predict(cv.nc.fit, newx = x[50:60, ], s = "lambda.min")
```

---

predict.hdsvm *Make Predictions from a ‘hdsvm‘ Object*

Description

Produces fitted values for new predictor data using a fitted ‘hdsvm()‘ object.

Usage

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

Arguments

object	Fitted ‘hdsvm()‘ object from which predictions are to be derived.
newx	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
s	Values of the penalty parameter ‘lambda‘ for which predictions are requested. Defaults to the entire sequence used during the model fit.
type	Type of prediction required. Type ‘"class"‘ produces the predicted binary class labels and type ‘"loss"‘ returns the fitted values. Default is "class".
...	Not used.

Details

This function generates predictions at specified ‘lambda‘ values from a fitted ‘hdsvm()‘ object. It is essential to provide a new matrix of predictor values (‘newx‘) at which these predictions are to be made.

Value

Returns a vector or matrix of predicted values corresponding to the specified ‘lambda‘ values.

See Also

hdsvm, coef.hdsvm

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
fit <- hdsvm(x, y, lam2=lam2)
preds <- predict(fit, newx = tail(x), s = fit$lambda[3:5])
```

---

predict.nc.hdsvm *Make Predictions from a ‘nc.hdsvm‘ Object*

Description

Produces fitted values for new predictor data using a fitted ‘nc.hdsvm()‘ object.

Usage

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

Arguments

object	Fitted ‘nc.hdsvm()‘ object from which predictions are to be derived.
newx	Matrix of new predictor values for which predictions are desired. This must be a matrix and is a required argument.
s	Values of the penalty parameter ‘lambda‘ for which predictions are requested. Defaults to the entire sequence used during the model fit.

type	Type of prediction required. Type ‘"class"‘ produces the predicted binary class labels and type ‘"loss"‘ returns the fitted values. Default is "class".
...	Not used.

Details

This function generates predictions at specified ‘lambda‘ values from a fitted ‘nc.hdsvm()‘ object. It is essential to provide a new matrix of predictor values (‘newx‘) at which these predictions are to be made.

Value

Returns a vector or matrix of predicted values corresponding to the specified ‘lambda‘ values.

See Also

nc.hdsvm, coef.nc.hdsvm

Examples

```
set.seed(315)
n <- 100
p <- 400
x1 <- matrix(rnorm(n / 2 * p, -0.25, 0.1), n / 2)
x2 <- matrix(rnorm(n / 2 * p, 0.25, 0.1), n / 2)
x <- rbind(x1, x2)
beta <- 0.1 * rnorm(p)
prob <- plogis(c(x %*% beta))
y <- 2 * rbinom(n, 1, prob) - 1
lam2 <- 0.01
lambda <- 10^(seq(1,-4, length.out = 30))
nc.fit <- nc.hdsvm(x = x, y = y, lambda = lambda, lam2 = lam2, pen = "scad")
nc.preds <- predict(nc.fit, newx = tail(x), s = nc.fit$lambda[3:5])
```

Index