Package 'sparseSVM'

July 23, 2025

Type Package

Title Solution Paths of Sparse High-Dimensional Support Vector Machine with Lasso or Elastic-Net Regularization

Version 1.1-7

Date 2024-09-26

Description Offers a fast algorithm for fitting solution paths of sparse SVM models with lasso or elastic-net regularization. Reference: Congrui Yi and Jian Huang (2017) <doi:10.1080/10618600.2016.1256816>.

License GPL-3

NeedsCompilation yes

URL https://github.com/CY-dev/sparseSVM

Imports parallel

Author Congrui Yi [aut, cre], Yaohui Zeng [aut]

Maintainer Congrui Yi <eric.ycr@gmail.com>

Repository CRAN

Date/Publication 2024-09-26 20:50:02 UTC

Contents

sparseSVM-package 2
cv.sparseSVM 3
plot.cv.sparseSVM 4
plot.sparseSVM 5
predict.cv.sparseSVM 6
predict.sparseSVM 8
sparseSVM 9

Index **12**

`sparseSVM-package` *Solution Paths for Sparse High-dimensional Support Vector Machine with Lasso or Elastic-Net Regularization*

Description

Fast algorithm for fitting solution paths for sparse SVM regularized by lasso or elastic-net that generate sparse solutions.

Details

Package:	sparseSVM
Type:	Package
Version:	1.1-7
Date:	2024-09-23
License:	GPL-3

Accepts `X`, `y` data for binary classification and produces the solution path over a grid of values of the regularization parameter `lambda`. Also provides functions for plotting, prediction and parallelized cross-validation.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

References

Yi, C. and Huang, J. (2017) *Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, doi: 10.1080/10618600.2016.1256816 *Journal of Computational and Graphical Statistics*

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

fit = sparseSVM(X, y)
coef(fit, 0.05)
predict(fit, X[1:5,], lambda = c(0.2, 0.1))
plot(fit)
```

```
cv.fit <- cv.sparseSVM(X, y, ncores = 2, seed = 1234)
predict(cv.fit, X)
coef(cv.fit)
plot(cv.fit)
```

`cv.sparseSVM` *Cross validation for sparseSVM*

Description

Perform k-fold cross validation for sparse linear SVM regularized by lasso or elastic-net over a sequence of lambda values and find an optimal lambda.

Usage

```
cv.sparseSVM(X, y, ..., ncores = 1, eval.metric = c("me"),
             nfolds = 10, fold.id, seed, trace = FALSE)
```

Arguments

`X`	Input matrix.
`y`	Response vector.
`...`	Additional arguments to `sparseSVM`.
`ncores`	`cv.sparseSVM` can be run in parallel across a cluster using the `parallel` package. If `ncores` > 1,a cluster is created to run `cv.sparseSVM` in parallel. The code is run in series if `ncores` = 1 (the default). An error occurs if `ncores` is larger than the total number of available cores.
`eval.metric`	The metric used to choose optimial `lambda`. Current version only supports "me": misclassification error.
`nfolds`	The number of cross-validation folds. Default is 10.
`seed`	The seed of the random number generator in order to obtain reproducible results.
`fold.id`	Which fold each observation belongs to. By default the observations are randomly assigned by `cv.sparseSVM`.
`trace`	If set to TRUE, cv.sparseSVM will inform the user of its progress by announcing the beginning of each CV fold. Default is FALSE. (No trace output when running in parallel even if `trace=TRUE`.)

Details

The function randomly partitions the data in nfolds. It calls `sparseSVM` nfolds+1 times, the first to obtain the lambda sequence, and the remainder to fit with each of the folds left out once for validation. The cross-validation error is the average of validation errors for the nfolds fits.

Note by default, the cross-validation fold assignments are balanced across the two classes, so that each fold has the same class proportion (or as close to the same proportion as it is possible to achieve if cases do not divide evenly).

Value

The function returns an object of S3 class "cv.sparseSVM", which is a list containing:

`cve`	The validation error for each value of `lambda`, averaged across the cross-validation folds.
`cvse`	The estimated standard error associated with each value of `cve`.
`lambda`	The values of lambda used in the cross-validation fits.
`fit`	The fitted `sparseSVM` object for the whole data.
`min`	The index of `lambda` corresponding to `lambda.min`.
`lambda.min`	The value of `lambda` with the minimum cross-validation error in terms of `eval.metric`.
`eval.metric`	The metric used in selecting optimal `lambda`.
`fold.id`	The same as above.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

See Also

`sparseSVM`, `predict.cv.sparseSVM`, `plot.cv.sparseSVM`

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

cv.fit1 <- cv.sparseSVM(X, y, nfolds = 5, ncores = 2, seed = 1234)
cv.fit2 <- cv.sparseSVM(X, y, nfolds = 5, seed = 1234)
stopifnot(all.equal(cv.fit1, cv.fit2))
```

---

plot.cv.sparseSVM *Plot the cross-validation curve for a "cv.sparseSVM" object*

Description

Plot the cross-validation curve for a "cv.sparseSVM" object against the `lambda` values used, along with standard error bars.

Usage

```
## S3 method for class 'cv.sparseSVM'
plot(x, log.l = TRUE, nvars = TRUE, ...)
```

Arguments

x	A "cv.sparseSVM" object.
log.l	Should log(lambda) be used instead of lambda for the X-axis? Default is TRUE.
nvars	If TRUE (the default), places an axis on top of the plot denoting the number of variables with nonzero coefficients at each lambda.
...	Other graphical parameters to plot

Details

Produces a plot of mean cv errors at each lambda along with upper and lower standard error bars.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

See Also

sparseSVM, cv.sparseSVM

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

cv.fit <- cv.sparseSVM(X, y, ncores = 2, seed = 1234)
plot(cv.fit)
plot(cv.fit, log.l = FALSE)
```

---

plot.sparseSVM *Plot coefficients from a "sparseSVM" object*

Description

Produce a plot of the coefficient paths for a fitted "sparseSVM" object.

Usage

```
## S3 method for class 'sparseSVM'
plot(x, xvar = c("lambda", "norm"), log.l = TRUE, nvars = TRUE,
    alpha = 1, ...)
```

Arguments

x	A sparseSVM object.
xvar	What is on the X-axis. "lambda" plots against the lambda sequence, "norm" against the L1-norm of the coefficients. Default is "lambda".
log.l	Should log(lambda) be used instead of lambda when xvar = "lambda"? Default is TRUE. It has no effect on "norm".
nvars	If TRUE (the default), places an axis on top of the plot denoting the number of variables with nonzero coefficients at each lambda.
alpha	A value between 0 and 1 for alpha transparency channel(0 means transparent and 1 means opaque), helpful when the number of variables is large.
...	Other graphical parameters to plot.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

See Also

sparseSVM

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

fit = sparseSVM(X, y)
par(mfrow = c(2,2))
plot(fit)
plot(fit, nvars = FALSE, alpha = 0.5)
plot(fit, log.l = FALSE)
plot(fit, xvar = "norm")
```

---

predict.cv.sparseSVM *Model predictions based on "cv.sparseSVM" object.*

Description

This function returns fitted values, coefficients and more from a fitted "cv.sparseSVM" object.

Usage

```
## S3 method for class 'cv.sparseSVM'
predict(object, X, lambda = object$lambda.min,
        type = c("class","coefficients","nvars"), exact = FALSE, ...)
## S3 method for class 'cv.sparseSVM'
coef(object, lambda = object$lambda.min, exact = FALSE, ...)
```

Arguments

object	Fitted "cv.sparseSVM" model object.
X	Matrix of values at which predictions are to be made. Used only for type = "class".
lambda	Values of the regularization parameter lambda at which predictions are requested. Default is the one corresponding to the minimum cross-validation error.
type	Type of prediction. "class" returns the class labels; "coefficients" returns the coefficients; "nvars" returns the number of nonzero coefficients at each value of lambda.
exact	If exact=FALSE (default), then the function uses linear interpolation to make predictions for values of lambda that do not coincide with those used to fit the model. If exact=TRUE, and predictions are requested at values of lambda not included in the original fit, the model is refit on a lambda sequence consisting object$lambda and the new ones before predictions are made.
...	Not used. Other arguments to predict.

Value

The object returned depends on type.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

See Also

sparseSVM, cv.sparseSVM

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

cv.fit <- cv.sparseSVM(X, y, ncores = 2, seed = 1234)
predict(cv.fit, X)
predict(cv.fit, type = 'nvars')
```

```
predict(cv.fit, type = 'coef')
coef(cv.fit)
```

`predict.sparseSVM` *Model predictions based on "sparseSVM" object.*

Description

This function returns fitted values, coefficients and more from a fitted `"sparseSVM"` object.

Usage

```
## S3 method for class 'sparseSVM'
predict(object, X, lambda, type = c("class","coefficients","nvars"),
    exact = FALSE, ...)
## S3 method for class 'sparseSVM'
coef(object, lambda, exact = FALSE, ...)
```

Arguments

`object`	Fitted `"sparseSVM"` model object.
`X`	Matrix of values at which predictions are to be made. Used only for `type = "class"`.
`lambda`	Values of the regularization parameter `lambda` at which predictions are requested. Default is the entire sequence used to create the model.
`type`	Type of prediction. `"class"` returns the class labels; `"coefficients"` returns the coefficients; `"nvars"` returns the number of nonzero coefficients at each value of `lambda`.
`exact`	If `exact=FALSE` (default), then the function uses linear interpolation to make predictions for values of `lambda` that do not coincide with those used to fit the model. If `exact=TRUE`, and predictions are requested at values of `lambda` not included in the original fit, the model is refit on a lambda sequence consisting `object$lambda` and the new ones before predictions are made.
`...`	Not used. Other arguments to predict.

Value

The object returned depends on type.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

See Also

sparseSVM

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

fit = sparseSVM(X, y)
predict(fit, X[1:5,], lambda = c(0.05, 0.03))
predict(fit, X[1:5,], lambda = 0.05, exact = TRUE)
predict(fit, type = "nvars")
coef(fit, lambda = 0.05)
```

---

sparseSVM *Fit sparse linear SVM with lasso or elasti-net regularization*

Description

Fit solution paths for sparse linear SVM regularized by lasso or elastic-net over a grid of values for the regularization parameter lambda.

Usage

```
sparseSVM(X, y, alpha = 1, gamma = 0.1, nlambda=100,
  lambda.min = ifelse(nrow(X)>ncol(X), 0.01, 0.05),
          lambda, preprocess = c("standardize", "rescale", "none"),
          screen = c("ASR", "SR", "none"), max.iter = 1000, eps = 1e-5,
          dfmax = ncol(X)+1, penalty.factor=rep(1, ncol(X)), message = FALSE)
```

Arguments

X	Input matrix.
y	Output vector. Currently the function only supports binary output and converts the output into +1/-1 coding internally.
alpha	The elastic-net mixing parameter that controls the relative contribution from the lasso and the ridge penalty. It must be a number between 0 and 1. alpha=1 is the lasso penalty and alpha=0 the ridge penalty.
gamma	The tuning parameter for huberization smoothing of hinge loss. Default is 0.1.
nlambda	The number of lambda values. Default is 100.
lambda.min	The smallest value for lambda, as a fraction of lambda.max, the data derived entry value. Default is 0.01 if the number of observations is larger than the number of variables and 0.05 otherwise.

lambda	A user-specified sequence of lambda values. Typical usage is to leave blank and have the program automatically compute a lambda sequence based on nlambda and lambda.min. Specifying lambda overrides this. This argument should be used with care and supplied with a decreasing sequence instead of a single value. To get coefficients for a single lambda, use coef or predict instead after fitting the solution path with sparseSVM.
preprocess	Preprocessing technique to be applied to the input. Either "standardize" (default), "rescale" or "none" (see Details). The coefficients are always returned on the original scale.
screen	Screening rule to be applied at each lambda that discards variables for speed. Either "ASR" (default), "SR" or "none". "SR" stands for the strong rule, and "ASR" for the adaptive strong rule. Using "ASR" typically requires fewer iterations to converge than "SR", but the computing time are generally close. Note that the option "none" is used mainly for debugging, which may lead to much longer computing time.
max.iter	Maximum number of iterations. Default is 1000.
eps	Convergence threshold. The algorithms continue until the maximum change in the objective after any coefficient update is less than eps times the null deviance. Default is 1E-7.
dfmax	Upper bound for the number of nonzero coefficients. The algorithm exits and returns a partial path if dfmax is reached. Useful for very large dimensions.
penalty.factor	A numeric vector of length equal to the number of variables. Each component multiplies lambda to allow differential penalization. Can be 0 for some variables, in which case the variable is always in the model without penalization. Default is 1 for all variables.
message	If set to TRUE, sparseSVM will inform the user of its progress. This argument is kept for debugging. Default is FALSE.

Details

The sequence of models indexed by the regularization parameter lambda is fitted using a semismooth Newton coordinate descent algorithm. The objective function is defined to be

$$\frac{1}{n}\sum hingeLoss(y_i(x_i'w+b)) + \lambda \text{penalty}(w).$$

where

$$hingeLoss(t) = max(0, 1-t)$$

and the intercept b is unpenalized.

The program supports different types of preprocessing techniques. They are applied to each column of the input matrix X. Let x be a column of X. For preprocess = "standardize", the formula is

$$x' = \frac{x - mean(x)}{sd(x)};$$

for preprocess = "rescale",

$$x' = \frac{x - min(x)}{max(x) - min(x)}.$$

The models are fit with preprocessed input, then the coefficients are transformed back to the original scale via some algebra.

Value

The function returns an object of S3 class "sparseSVM", which is a list containing:

call	The call that produced this object.
weights	The fitted matrix of coefficients. The number of rows is equal to the number of coefficients, and the number of columns is equal to nlambda. An intercept is included.
iter	A vector of length nlambda containing the number of iterations until convergence at each value of lambda.
saturated	A logical flag for whether the number of nonzero coefficients has reached dfmax.
lambda	The sequence of regularization parameter values in the path.
alpha	Same as above.
gamma	Same as above.
penalty.factor	Same as above.
levels	Levels of the output class labels.

Author(s)

Congrui Yi and Yaohui Zeng
Maintainer: Congrui Yi <eric.ycr@gmail.com>

See Also

plot.sparseSVM, cv.sparseSVM

Examples

```
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*rnorm(10)
eps = rnorm(1000)
y = sign(b + drop(X[,1:10] %*% w + eps))

fit = sparseSVM(X, y)
coef(fit, 0.05)
predict(fit, X[1:5,], lambda = c(0.2, 0.1))
```

Index

∗ **SVM**
cv.sparseSVM, 3
plot.cv.sparseSVM, 4
plot.sparseSVM, 5
predict.cv.sparseSVM, 6
predict.sparseSVM, 8
sparseSVM, 9
sparseSVM-package, 2
∗ **classification**
cv.sparseSVM, 3
plot.cv.sparseSVM, 4
plot.sparseSVM, 5
predict.cv.sparseSVM, 6
predict.sparseSVM, 8
sparseSVM, 9
sparseSVM-package, 2
∗ **machine learning**
cv.sparseSVM, 3
plot.cv.sparseSVM, 4
plot.sparseSVM, 5
predict.cv.sparseSVM, 6
predict.sparseSVM, 8
sparseSVM, 9
sparseSVM-package, 2
∗ **models**
cv.sparseSVM, 3
plot.cv.sparseSVM, 4
plot.sparseSVM, 5
predict.cv.sparseSVM, 6
predict.sparseSVM, 8
sparseSVM, 9
sparseSVM-package, 2

coef.cv.sparseSVM
*(*predict.cv.sparseSVM*)*, 6
coef.sparseSVM *(*predict.sparseSVM*)*, 8
cv.sparseSVM, 3, *5*, *7*, *11*

plot.cv.sparseSVM, *4*, 4
plot.sparseSVM, 5, *11*

predict.cv.sparseSVM, *4*, 6
predict.sparseSVM, 8

sparseSVM, *4–7*, *9*, 9
sparseSVM-package, 2