Package ‘sparseSVM’

June 2, 2018

Type  Package
Title  Solution Paths of Sparse High-Dimensional Support Vector Machine with Lasso or Elastic-Net Regularization
Version  1.1-6
Date  2018-06-01
Author  Congrui Yi and Yaohui Zeng
Maintainer  Congrui Yi <eric.ycr@gmail.com>
Description  Fast algorithm for fitting solution paths of sparse SVM models with lasso or elastic-net regularization.
License  GPL-3
NeedsCompilation  yes
Imports  parallel
Repository  CRAN
Date/Publication  2018-06-02 12:27:22 UTC

R topics documented:

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

Index  12
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: 1.1-6
Date: 2018-06-01
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

Journal of Computational and Graphical Statistics

Examples

\[
X = \text{matrix}(\text{rnorm}(1000 \times 100), 1000, 100) \\
b = 3 \\
w = 5 \times \text{rnorm}(10) \\
\text{eps} = \text{rnorm}(1000) \\
y = \text{sign}(b + \text{drop}(X[,1:10] \times w + \text{eps}))
\]

\[
\text{fit} = \text{sparseSVM}(X, y) \\
\text{coef}(\text{fit}, 0.05) \\
\text{predict}(\text{fit}, X[1:5,], \lambda = c(0.2, 0.1)) \\
\text{plot}(\text{fit})
\]
cv.sparseSVM

```
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**

```r
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 optimal 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

```r
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*runif(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

```r
## 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

```r
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)
```

Description

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

Usage

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

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

```r
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")
```

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

```r
X = matrix(rnorm(1000*100), 1000, 100)
b = 3
w = 5*runnorm(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')
```
This function returns fitted values, coefficients and more from a fitted "sparseSVM" object.

Usage

```r
## 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
sparseSVM

Examples

```r
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)
```
To get coefficients for a single lambda, use coef or predict instead after fitting the solution path with \texttt{sparsesvm}.

\textbf{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.

\textbf{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.

\textbf{max.iter}  
Maximum number of iterations. Default is 1000.

\textbf{eps}  
Convergence threshold. The algorithms continue until the maximum change in the objective after any coefficient update is less than \texttt{eps} times the null deviance. Default is 1E-7.

\textbf{dfmax}  
Upper bound for the number of nonzero coefficients. The algorithm exits and returns a partial path if \texttt{dfmax} is reached. Useful for very large dimensions.

\textbf{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.

\textbf{message}  
If set to \texttt{TRUE}, \texttt{sparsesvm} will inform the user of its progress. This argument is kept for debugging. Default is \texttt{FALSE}.

**Details**

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

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

where

$$
\text{hingeLoss}(t) = \max(0, 1 - t)
$$

and the intercept \texttt{b} is unpenalized.

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

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

for \texttt{preprocess = "rescale"},

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

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

```r
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

*Topic **SVM**
  cv.sparseSVM, 3
  plot.cv.sparseSVM, 4
  plot.sparseSVM, 5
  predict.cv.sparseSVM, 6
  predict.sparseSVM, 8
  sparseSVM, 9
  sparseSVM-package, 2

*Topic **classification**
  cv.sparseSVM, 3
  plot.cv.sparseSVM, 4
  plot.sparseSVM, 5
  predict.cv.sparseSVM, 6
  predict.sparseSVM, 8
  sparseSVM, 9
  sparseSVM-package, 2

*Topic **machine learning**
  cv.sparseSVM, 3
  plot.cv.sparseSVM, 4
  plot.sparseSVM, 5
  predict.cv.sparseSVM, 6
  predict.sparseSVM, 8
  sparseSVM, 9
  sparseSVM-package, 2

*Topic **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
  sparseSVM, 3, 5, 7, 11

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