

Package ‘ccvsvm’

May 19, 2022

Type Package

Title Consolidated Cross-Validation for Support Vector Machines

Version 1.0.0

Date 2022-05-19

Author Anonymous <anonymous@gmail.com>

Maintainer Anonymous <anonymous@gmail.com>

Description Consolidated cross-validation algorithm for the kernel support vector machines.

Depends methods

Imports stats, utils

License GPL-2

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.1.2

R topics documented:

| | |
|---------------|----------|
| BUPA | 1 |
| kernel-class | 2 |
| predict.scsvm | 3 |
| scsvm | 4 |
| sigest | 6 |
| Index | 8 |

BUPA

BUPA's liver disorders data

Description

BUPA's liver disorders data: 345 male individuals' blood test result and liver disorder status.

Format

Data objects used to demonstrate the ccvsvm package.

Details

This data set consists of 345 observations and 6 predictors representing the blood test result liver disorder status of 345 patients. The three predictors are mean corpuscular volume (MCV), alkaline phosphatase (ALP), aspartate aminotransferase (SGPT), aspartate aminotransferase (SGOT), gamma-glutamyl transpeptidase (GGT), and the number of alcoholic beverage drinks per day (DRINKS).

Value

A list with the following elements:

- | | |
|---|---|
| X | A numerical matrix for predictors: 345 rows and 6 columns; each row corresponds to a patient. |
| y | A numeric vector of length 305 representing the liver disorder status. |

Source

The data set is available for download from UCI machine learning repository.

Examples

```
# load data set
data(BUPA)
# the number of samples predictors
dim(BUPA$X)
# the number of samples for each class
sum(BUPA$y == -1)
sum(BUPA$y == 1)
```

kernel-class

Kernel Functions

Description

Kernel functions provided in the R package `scsvm`. Details can be seen in the reference below.

The Gaussian RBF kernel $k(x, x') = \exp(-\sigma \|x - x'\|^2)$

The Polynomial kernel $k(x, x') = (scale < x, x' > + offset)^{degree}$

The Linear kernel $k(x, x') = \langle x, x' \rangle$

The Laplacian kernel $k(x, x') = \exp(-\sigma \|x - x'\|)$

The Bessel kernel $k(x, x') = (-\text{Bessel}_{\nu+1}^n \sigma \|x - x'\|^2)$

The ANOVA RBF kernel $k(x, x') = \sum_{1 \leq i_1, \dots, i_D \leq N} \prod_{d=1}^D k(x_{i_d}, x'_{i_d})$ where $k(x, x')$ is a Gaussian RBF kernel.

The Spline kernel $\prod_{d=1}^D 1 + x_i x_j + x_i x_j \min(x_i, x_j) - \frac{x_i + x_j}{2} \min(x_i, x_j)^2 + \frac{\min(x_i, x_j)^3}{3}$. The parameter σ used in `rbfdot` can be selected by `sigest()`.

Arguments

- | | |
|--------|--|
| sigma | The inverse kernel width used by the Gaussian, the Laplacian, the Bessel, and the ANOVA kernel. |
| degree | The degree of the polynomial, bessel or ANOVA kernel function. This has to be an positive integer. |

| | |
|--------|--|
| scale | The scaling parameter of the polynomial kernel function. |
| offset | The offset used in a polynomial kernel. |
| order | The order of the Bessel function to be used as a kernel. |

Details

These R functions and descriptions are directly duplicated and/or adapted from the R package kernlab.

References

Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004) “*kernlab – An S4 Package for Kernel Methods in R*”, *Journal of Statistical Software*, Vol. 11(9), 1-20, <https://www.jstatsoft.org/v11/i09/paper>.

Examples

```
data(BUPA)
# generate a linear kernel
kfun <- vanilladot()

# generate a Laplacian kernel function with sigma = 1
kfun <- laplacedot(sigma=1)

# set kern=kfun when fitting an scsvm object
data(BUPA)
BUPA$X <- scale(BUPA$X, center=TRUE, scale=TRUE)
lambda <- 10^(seq(-3, 3, length.out=10))
m1 <- scsvm(BUPA$X, BUPA$y, kern=kfun, lambda=lambda, eps=1e-5, maxit=1e5)
```

| | |
|---------------|--|
| predict.scsvm | <i>predict class labels for new observations</i> |
|---------------|--|

Description

Predict the binary class labels or the fitted values for a scsvm object.

Usage

```
## S3 method for class 'scsvm'
predict(
  object,
  kern,
  x,
  newx,
  s = object$lambda,
  type = c("class", "loss"),
  exact = FALSE,
  ...
)
```

Arguments

| | |
|--------|--|
| object | A fitted scsvm object. |
| kern | A kernel function used when fitting the scsvm object. |
| x | The predictor matrix, i.e., the x matrix used when fitting the scsvm object. |
| newx | A matrix of new values for x at which predictions are to be made. Note that newx must be of a matrix form, predict function does not accept a vector or other formats of newx. |
| s | A lambda value. |
| type | "class" or "loss"? "class" produces the predicted binary class labels and "loss" returns the fitted values. Default is "class". |
| exact | Whether to refit the scsvm object to get the exact coefficients at the requested s value or use linear interpolation to compute approximated coefficients. Default is FALSE, say, the approximated coefficients. |
| ... | Not used. |

Details

If type="class", the function returns the predicted class labels. If type="loss", the result is $\beta_0 + K'_i \alpha$ where β_0 and α are from the scsvm object and K_i is the ith row of the kernel matrix.

Value

Returns either the predicted class labels or the fitted values, depending on the choice of type.

Examples

```
data(BUPA)
# standardize the predictors
BUPA$X <- scale(BUPA$X, center=TRUE, scale=TRUE)
# Gaussian kernel
kern <- rbfdot(sigma=sigest(BUPA$X))
# a grid of tuning parameters
lambda = 10^(seq(3, -3, length.out=10))
fit <- scsvm(BUPA$X, BUPA$y, kern, lambda=lambda)
predict(fit, kern, BUPA$X, tail(BUPA$X))
```

scsvm

efficiently the kernel support vector machines with an efficient data screening algorithm.

Description

Trains the kernel large-margin classifiers and carries out an integrated leave-one-out cross-validation procedure to estimate the cross-validation error and find the optimal value of the tuning parameter lambda. The algorithm is very efficient in performing the cross-validation due to a data screening algorithm.

Usage

```
scsvm(
  x,
  y,
  kern,
  lambda,
  pred.loss = NULL,
  isdr = TRUE,
  intcpt = TRUE,
  maxit = 1e+05,
  eps = 1e-05,
  maxup = length(lambda)
)
```

Arguments

| | |
|------------------------|---|
| <code>x</code> | A numerical input matrix. The dimension is n rows and p columns. |
| <code>y</code> | Response variable. The length is n and every element is either -1 or 1. |
| <code>kern</code> | A kernel function. |
| <code>lambda</code> | A user-supplied lambda sequence. |
| <code>pred.loss</code> | Whether to use "misclass", the mis-classification error, or "loss", the margin loss value, to compute the cross-validation error. |
| <code>isdr</code> | Whether to use the data screening algorithm. Default is TRUE. |
| <code>intcpt</code> | Whether to include an intercept term. Default is TRUE. |
| <code>maxit</code> | Maximum number of iterates. |
| <code>eps</code> | Convergence threshold. |
| <code>maxup</code> | The largest number of consequent lambda value associated with increasing cross-validation error before the algorithm terminates. |

Details

The function efficiently computes the average cross-validation error and reports the standard error. This function solves the cross-validated kernel SVM with a data screening algorithm.

Value

An object with S3 class `scsvm`

| | |
|---------------------|---|
| <code>alpha</code> | An $n + 1$ by L matrix of coefficients, where n is the number of observations and L is the number of tuning parameters. The first row of alpha contains the intercepts. |
| <code>npass</code> | The total number of iterates used to train the classifier. |
| <code>lnpass</code> | The total number of iterates including both training and cross-validation. |
| <code>jerr</code> | Warnings and errors; 0 if none. |
| <code>info</code> | A list includes some settings used to fit this object: <code>eps</code> , <code>maxit</code> , <code>intcpt</code> , and <code>kern</code> . |
| <code>lambda</code> | The lambda sequence that was actually used. |
| <code>anc</code> | The number of observations after screening. |

| | |
|------------|--|
| KKT | KKT criterion of the fitted SVM solutions. |
| KKTloo | KKT criterion of the leave-one-out cross-validated SVM solutions. |
| cvm | Mean cross-validation error. |
| cvsd | Estimates of standard error of cross-validation error. |
| cvup | The upper curve: $cvm + cvsd$. |
| cvlo | The lower curve: $cvm - cvsd$. |
| cvm.min | The cross-validation error at <code>lambda.min</code> . |
| cvm.1se | The cross-validation error at <code>lambda.1se</code> . |
| lambda.min | The lambda incurring the minimum cross-validation error. |
| lambda.1se | The largest lambda whose error is within one standard error of the minimum. |
| name | Whether the cross-validation error is computed based on "misclass", mis-classification error, or "loss", the SVM loss. |
| Kmat | kernel matrix. |
| Nobs | The sample size. |
| call | The call that produced this object. |

Examples

```
data(BUPA)
# standardize the predictors
BUPA$X <- scale(BUPA$X, center=TRUE, scale=TRUE)
# Gaussian kernel
kern <- rbfdot(sigma=sigest(BUPA$X))
# a grid of tuning parameters
lambda <- 10^(seq(3, -3, length.out=10))
fit <- scsvm(BUPA$X, BUPA$y, lambda=lambda)
```

| | |
|---------------------|--|
| <code>sigest</code> | <i>estimates the sigma parameter for the Gaussian kernel</i> |
|---------------------|--|

Description

Estimates the sigma parameter which is used for fitting kernel large-margin classifiers.

Usage

```
sigest(x, frac = 0.5)
```

Arguments

| | |
|-------------------|--|
| <code>x</code> | A numerical input matrix. The dimension is n rows and p columns. |
| <code>frac</code> | Fraction of data to use for estimation. Default is 0.5. |

Details

The functions gives a tuning-free sigma parameter of the Gaussian kernel for use with kernel large-margin classifiers. This function is modified from the `sigest` function from the `kernlab` package.

Value

A sigma parameter for the Gaussian kernel.

References

Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004) “*kernlab – An S4 Package for Kernel Methods in R*”, *Journal of Statistical Software*, Vol. 11(9), 1-20, <https://www.jstatsoft.org/v11/i09/paper>.

Examples

```
data(BUPA)
# standardize the predictors
BUPA$X <- scale(BUPA$X, center=TRUE, scale=TRUE)
# Gaussian kernel
kern <- rbfdot(sigma=sigest(BUPA$X))
# a grid of tuning parameters
lambda <- 10^(seq(3, -3, length.out=10))
fit <- scsvm(BUPA$X, BUPA$y, lambda=lambda)
```

Index

* **classification**

predict.scsvm, [3](#)

scsvm, [4](#)

* **datasets**

BUPA, [1](#)

* **kernel**

predict.scsvm, [3](#)

scsvm, [4](#)

anovadot (kernel-class), [2](#)

anovakernel-class (kernel-class), [2](#)

besseldot (kernel-class), [2](#)

besselkernel-class (kernel-class), [2](#)

BUPA, [1](#)

dots (kernel-class), [2](#)

kern (kernel-class), [2](#)

kernel-class, [2](#)

laplacedot (kernel-class), [2](#)

laplacekernel-class (kernel-class), [2](#)

polydot (kernel-class), [2](#)

polykernel-class (kernel-class), [2](#)

predict.scsvm, [3](#)

rbfdot (kernel-class), [2](#)

rbfkernel-class (kernel-class), [2](#)

scsvm, [4](#)

sigest, [6](#)

splinedot (kernel-class), [2](#)

splinekernel-class (kernel-class), [2](#)

vanilladot (kernel-class), [2](#)

vanillakernel-class (kernel-class), [2](#)