Package: dcsvm (via r-universe)

October 9, 2026

Type Package

Title Density Convoluted Support Vector Machines

Version 0.0.1

Date 2025-01-08

Description Implements an efficient algorithm for solving sparse-penalized support vector machines with kernel density convolution. This package is designed for high-dimensional classification tasks, supporting lasso (L1) and elastic-net penalties for sparse feature selection and providing options for tuning kernel bandwidth and penalty weights. The 'dcsvm' is applicable to fields such as bioinformatics, image analysis, and text classification, where high-dimensional data commonly arise. Learn more about the methodology and algorithm at Wang, Zhou, Gu, and Zou (2023) <doi:10.1109/TIT.2022.3222767>.

Depends Matrix

Imports grDevices, graphics, methods, stats

License GPL-2

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.3.2

Author Boxiang Wang [aut, cre], Le Zhou [aut], Yuwen Gu [aut], Hui Zou [aut]

Maintainer Boxiang Wang <boxiang-wang@uiowa.edu>

Repository https://boxiang-wang.r-universe.dev

Date/Publication 2025-01-10 21:20:02 UTC

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

RemoteRef HEAD

RemoteSha 5a439b70d0dbc4ef00bb398fa774552a0c49b049

Contents

dcsvm-package . . . 2
coef.cv.dcsvm . . . 3
coef.dcsvm . . . 4
colon . . . 5
cv.dcsvm . . . 6
dcsvm . . . 8
plot.cv.dcsvm . . . 10
plot.dcsvm . . . 11
predict.cv.dcsvm . . . 12
predict.dcsvm . . . 13
print.dcsvm . . . 14

Index **16**

`dcsvm-package` *Density-Convoluted Support Vector Machines*

Description

This package provides tools to perform density-convoluted support vector machine (DCSVM) modeling for high-dimensional data classification.

Details

This package implements the density-convoluted SVM for high-dimensional classification.

Package:	dcsvm
Type:	Package
Version:	0.0.1
Date:	2025-01-08
License:	GPL-2

The package `dcsvm` contains five main functions:

- `dcsvm`
- `cv.dcsvm`
- `coef.dcsvm`
- `plot.dcsvm`
- `plot.cv.dcsvm`

Author(s)

Boxiang Wang, Le Zhou, Yuwen Gu, and Hui Zou
Maintainer: Boxiang Wang <boxiang-wang@uiowa.edu>

References

Wang, B., Zhou, L., Gu, Y., and Zou, H. (2023) *Density-Convoluted Support Vector Machines for High-Dimensional Classification, IEEE Transactions on Information Theory, Vol. 69(4), 2523-2536*,

---

`coef.cv.dcsvm` *Compute Coefficients from a "cv.dcsvm" Object*

Description

Computes the coefficients at specified `lambda` values for a `cv.dcsvm` object.

Usage

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

Arguments

`object`	A fitted `cv.dcsvm` object, obtained by conducting cross-validation on the sparse density-convoluted SVM model.
`s`	Value(s) of the L1 tuning parameter `lambda` for computing coefficients. Default is `"lambda.1se"`, the largest `lambda` value achieving a cross-validation error within one standard error of the minimum. Alternatively, `"lambda.min"` corresponds to the `lambda` incurring the least cross-validation error. `s` can also be numeric, specifying the value(s) to use.
`...`	Other arguments that can be passed to `dcsvm`.

Details

Compute Coefficients from a "cv.dcsvm" Object

Computes coefficients at chosen values of `lambda` from the `cv.dcsvm` object.

This function computes the coefficients for `lambda` values suggested by cross-validation.

Value

The returned object depends on the choice of `s` and any additional arguments passed to the `dcsvm` method.

See Also

`cv.dcsvm` and `predict.cv.dcsvm` methods.

Examples

```
data(colon)
colon$x <- colon$x[ ,1:100] # Use only the first 100 columns for this example
set.seed(1)
cv <- cv.dcsvm(colon$x, colon$y, lam2=1, nfolds=5)
c1 <- coef(cv, s="lambda.1se")
```

`coef.dcsvm` *Compute Coefficients for Sparse Density-Convoluted SVM*

Description

Computes the coefficients or indices of nonzero coefficients at specified `lambda` values from a fitted `dcsvm` model.

Usage

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

Arguments

`object`	A fitted `dcsvm` object.
`s`	Value(s) of the L1 tuning parameter `lambda` for computing coefficients. Default is the entire `lambda` sequence obtained by `dcsvm`.
`type`	`"coefficients"` or `"nonzero"`? `"coefficients"` computes the coefficients at given values for `s`; `"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

Compute Coefficients for Sparse Density-Convoluted SVM

Computes the coefficients or returns the indices of nonzero coefficients at chosen values of `lambda` from a fitted `dcsvm` object.

`s` is the vector of `lambda` values at which predictions are requested. If `s` is not in the lambda sequence used for fitting the model, the `coef` function uses linear interpolation. The new values are interpolated using a fraction of coefficients from both left and right `lambda` indices.

Value

Either the coefficients at the requested values of `lambda`, or a list of the indices of the nonzero coefficients for each `lambda`.

See Also

predict.dcsvm

Examples

```
data(colon)
fit <- dcsvm(colon$x, colon$y, lam2=1)
c1 <- coef(fit, type="coefficients", s=c(0.1, 0.005))
c2 <- coef(fit, type="nonzero")
```

---

colon *Simplified Gene Expression Data from Alon et al. (1999)*

Description

This dataset contains 62 colon tissue samples with 2000 gene expression levels. Among these samples, 40 are tumor tissues (coded as 1) and 22 are normal tissues (coded as -1).

Usage

```
data(colon)
```

Details

Simplified Gene Expression Data from Alon et al. (1999)

Gene expression data (2000 genes for 62 samples) from a DNA microarray experiment of colon tissue samples (Alon et al., 1999).

Value

A list with the following elements:

x	A matrix of 62 rows and 2000 columns representing the gene expression levels of 62 colon tissue samples. Each row corresponds to a sample, and each column corresponds to a gene.
y	A numeric vector of length 62 representing the tissue type (1 for tumor; -1 for normal).

Source

The data were introduced in Alon et al. (1999).

References

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., and Levine, A.J. (1999). "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, **96**(12), 6745–6750.

Examples

```
# Load the dcsvm library
library(dcsvm)

# Load the dataset
data(colon)

# Check the dimensions of the data
dim(colon$x)

# Count the number of samples in each class
sum(colon$y == -1)
sum(colon$y == 1)
```

`cv.dcsvm` *Cross-Validation for Sparse Density-Convoluted SVM*

Description

Performs cross-validation for the sparse density-convoluted SVM to estimate the optimal tuning parameter `lambda`.

Usage

```
cv.dcsvm(x, y, lambda = NULL, hval = 1,
  pred.loss = c("misclass", "loss"), nfolds = 5, foldid, ...)
```

Arguments

`x`	A matrix of predictors, i.e., the `x` matrix used in `dcsvm`.
`y`	A vector of binary class labels, i.e., the `y` used in `dcsvm`.
`lambda`	Default is `NULL`, and the sequence generated by `dcsvm` is used. User can also provide a new `lambda` sequence for cross-validation.
`hval`	The bandwidth parameter for kernel smoothing. Default is 1.
`pred.loss`	`"misclass"` for classification error, `"loss"` for the density-convoluted SVM loss.
`nfolds`	The number of folds. Default is 5. The allowable range is from 3 to the sample size. Larger `nfolds` increases computational time.
`foldid`	An optional vector with values between 1 and `nfold`, representing the fold indices for each observation. If supplied, `nfolds` can be missing.
`...`	Other arguments that can be passed to `dcsvm`.

Details

Cross-Validation for Sparse Density-Convoluted SVM

Conducts a k-fold cross-validation for dcsvm and returns the suggested values of the L1 parameter lambda.

This function runs dcsvm on the sparse density-convoluted SVM by excluding each fold in turn, then computes the mean cross-validation error and standard deviation. It is adapted from the cv functions in the gcdnet and glmnet packages.

Value

A cv.dcsvm object is returned, which includes the cross-validation fit:

lambda	The lambda sequence used in dcsvm.
cvm	A vector of length length(lambda) for the mean cross-validated error.
cvsd	A vector of length length(lambda) for estimates of standard error of cvm.
cvupper	The upper curve: cvm + cvsd.
cvlower	The lower curve: cvm - cvsd.
nzero	Number of non-zero coefficients at each lambda.
name	"Mis-classification error", for plotting purposes.
dcsvm.fit	A fitted dcsvm object using the full data.
lambda.min	The lambda incurring the minimum cross-validation error cvm.
lambda.1se	The largest value of lambda such that error is within one standard error of the minimum.
cv.min	The minimum cross-validation error.
cv.1se	The cross-validation error associated with lambda.1se.

See Also

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

Examples

```
data(colon)
colon$x <- colon$x[ ,1:100] # Use only the first 100 columns for this example
n <- nrow(colon$x)
set.seed(1)
id <- sample(n, trunc(n / 3))
cvfit <- cv.dcsvm(colon$x[-id, ], colon$y[-id], lam2=1, nfolds=5)
plot(cvfit)
predict(cvfit, newx=colon$x[id, ], s="lambda.min")
```

dcsvm *Density-Convoluted Support Vector Machine*

Description

Fits the density-convoluted support vector machine (DCSVM) through kernel density convolutions.

Usage

```
dcsvm(
  x,
  y,
  nlambda = 100,
  lambda.factor = ifelse(nobs < nvars, 0.01, 1e-04),
  lambda = NULL,
  lam2 = 0,
  kern = c("gaussian", "uniform", "epanechnikov"),
  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,
  istrong = TRUE
)
```

Arguments

x	A numeric matrix with N rows and p columns representing predictors. Each row corresponds to an observation, and each column corresponds to a variable.
y	A numeric vector of length N representing binary responses. Elements must be either -1 or 1.
nlambda	Number of lambda values in the sequence. Default is 100.
lambda.factor	Ratio of the smallest to the largest lambda in the sequence: lambda.factor = min(lambda) / max(lambda). The default value is 0.0001 if $N >= p$ or 0.01 if $N < p$. Takes no effect if a lambda sequence is specified.
lambda	An optional user-specified sequence of lambda values. If lambda = NULL (default), the sequence is computed based on nlambda and lambda.factor. The program automatically sorts user-defined lambda sequences in decreasing order.
lam2	Users may tune λ_2, which controls the L2 regularization strength. Default is 0 (lasso).
kern	Type of kernel method for smoothing. Options are "gaussian", "uniform", and "epanechnikov". Default is "epanechnikov".

hval	The bandwidth parameter for kernel smoothing. Default is 1.
pf	A numeric vector of length p representing the L1 penalty weights for each coefficient. A common choice is $(\beta + 1/n)^{-1}$, where n is the sample size and β is obtained from L1 DCSVM or enet DCSVM. Default is 1 for all predictors.
pf2	A numeric vector of length p representing the L2 penalty weights for each coefficient. A value of 0 indicates no L2 shrinkage. Default is 1 for all predictors.
exclude	Indices of predictors to exclude from the model. Equivalent to assigning an infinite penalty factor. Default is none.
dfmax	Maximum number of nonzero coefficients allowed in the model. Default is $p+1$. Useful for large p when a partial path is acceptable.
pmax	Maximum number of variables allowed to ever be nonzero during the computation. Default is `min(dfmax * 1.2, p)`.
standardize	Logical indicating whether predictors should be standardized to unit variance. Default is `TRUE`. Note that predictors are always centered.
eps	Convergence threshold. The algorithm stops when $4 \max_j(\beta_j^{new} - \beta_j^{old})^2$ is less than `eps`. Default is `1e-8`.
maxit	Maximum number of iterations allowed. Default is `1e6`. Consider increasing `maxit` if the algorithm does not converge.
istrong	Logical indicating whether to use the strong rule for faster computation. Default is `TRUE`.

Value

An object of class `dcsvm` containing the following components:

b0	Intercept values for each `lambda`.
beta	Sparse matrix of coefficients for each `lambda`. Use `as.matrix()` to convert.
df	Number of nonzero coefficients for each `lambda`.
dim	Dimensions of the coefficient matrix.
lambda	Sequence of `lambda` values used.
npasses	Total number of iterations across all `lambda` values.
jerr	Warnings and errors. 0 if no errors.
call	The matched call.

See Also

`print.dcsvm`, `predict.dcsvm`, `coef.dcsvm`, `plot.dcsvm`, and `cv.dcsvm`.

Examples

```
# Load the data
data(colon)
# Fit the elastic-net penalized DCSVM with lambda2 to be 1
fit <- dcsvm(colon$x, colon$y, lam2 = 1)
print(fit)
```

```
# Coefficients at some lambda value
c1 <- coef(fit, s = 0.005)
# Make predictions
predict(fit, newx = colon$x[1:10, ], s = c(0.01, 0.005))
```

`plot.cv.dcsvm` *Plot the Cross-Validation Curve of Sparse Density-Convoluted SVM*

Description

Depicts the cross-validation curves for the sparse density-convoluted SVM.

Usage

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

Arguments

`x`	A fitted `cv.dcsvm` object.
`sign.lambda`	Specifies whether to plot against `log(lambda)` (default) or its negative if `sign.lambda = -1`.
`...`	Other graphical parameters to `plot`.

Details

Plot the Cross-Validation Curve of Sparse Density-Convoluted SVM

Plots the cross-validation curve against a function of `lambda` values, including upper and lower standard deviation curves.

This function visualizes the cross-validation curves for a `cv.dcsvm` object, which plots the relationship between `lambda` values and cross-validation error.

Value

No return value, only called for plots.

See Also

`cv.dcsvm`.

Examples

```
data(colon)
colon$x <- colon$x[ ,1:100] # Use only the first 100 columns for this example
set.seed(1)
cv <- cv.dcsvm(colon$x, colon$y, lam2=1, nfolds=5)
plot(cv)
```

plot.dcsvm *Plot Coefficients for Sparse Density-Convoluted SVM*

Description

Plots the solution paths as a coefficient profile plot for a fitted dcsvm model.

Usage

```
## S3 method for class 'dcsvm'
plot(x, xvar = c("norm", "lambda"), color = FALSE, label = FALSE, ...)
```

Arguments

x	A fitted dcsvm model.
xvar	Specifies the X-axis. If xvar == "norm", plots against the L1-norm of the coefficients; if xvar == "lambda", plots against the log-lambda sequence.
color	If TRUE, plots the curves with rainbow colors; otherwise, with gray colors (default).
label	If TRUE, labels the curves with variable sequence numbers. Default is FALSE.
...	Other graphical parameters to plot.

Details

Plot Coefficients for Sparse Density-Convoluted SVM

Plots the solution paths for a fitted dcsvm object.

This function generates a coefficient profile plot showing the solution paths of the sparse density-convoluted SVM.

Value

No return value, only called for plots.

See Also

print.dcsvm, predict.dcsvm, coef.dcsvm, plot.dcsvm, and cv.dcsvm.

Examples

```
data(colon)
fit <- dcsvm(colon$x, colon$y)
oldpar <- par(mfrow = c(1,3)) #changes par() and stores original par()
# Plots against the L1-norm of the coefficients
plot(fit)
# Plots against the log-lambda sequence
plot(fit, xvar="lambda", label=TRUE)
# Plots with colors
```

```
plot(fit, color=TRUE)
# Reset to user's option
par(oldpar)
```

`predict.cv.dcsvm` *Make Predictions from a "cv.dcsvm" Object*

Description

Predicts class labels for new data based on the cross-validated `lambda` values from a `cv.dcsvm` object.

Usage

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

Arguments

`object`	A fitted `cv.dcsvm` object.
`newx`	A matrix of new values for `x` at which predictions are to be made. Must be a matrix. See documentation for `predict.dcsvm`.
`s`	Value(s) of the L1 tuning parameter `lambda` for making predictions. Default is `s = "lambda.1se"` saved in the `cv.dcsvm` object. An alternative choice is `s = "lambda.min"`. `s` can also be numeric, representing the specific value(s) to use.
`...`	Not used. Other arguments to `predict`.

Details

Make Predictions from a "cv.dcsvm" Object

This function predicts the class labels of new observations using the sparse density-convoluted SVM at the `lambda` values suggested by `cv.dcsvm`.

This function uses the cross-validation results to make predictions. It is adapted from the `predict.cv` function in the `glmnet` and `gcdnet` packages.

Value

Predicted class labels or fitted values, depending on the choice of `s` and any arguments passed to the `dcsvm` method.

See Also

`cv.dcsvm`, and `coef.cv.dcsvm` methods.

Examples

```
data(colon)
colon$x <- colon$x[ , 1:100] # Use only the first 100 columns for this example
set.seed(1)
cv <- cv.dcsvm(colon$x, colon$y, lam2=1, nfolds=5)
predict(cv$dcsvm.fit, newx=colon$x[2:5, ],
  s=cv$lambda.1se, type="class")
```

`predict.dcsvm` *Make Predictions for Sparse Density-Convoluted SVM*

Description

Predicts binary class labels or fitted values for a `dcsvm` model using new data.

Usage

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

Arguments

`object`	A fitted `dcsvm` object.
`newx`	A matrix of new values for `x` at which predictions are to be made. Note that `newx` must be a matrix; vectors or other formats are not accepted.
`s`	Value(s) of the L1 tuning parameter `lambda` for computing coefficients. Default is the entire `lambda` sequence obtained by `dcsvm`.
`type`	`"class"` or `"link"`? `"class"` produces the predicted binary class labels, while `"link"` returns the fitted values. Default is `"class"`.
`...`	Not used. Other arguments to `predict`.

Details

Make Predictions for Sparse Density-Convoluted SVM

This function predicts the binary class labels or the fitted values of a `dcsvm` object.

`s` represents the new `lambda` values for making predictions. If `s` is not part of the original `lambda` sequence generated by `dcsvm`, `predict.dcsvm` uses linear interpolation to compute predictions by combining adjacent `lambda` values in the original sequence. This functionality is adapted from the `predict` methods in the `glmnet` and `gcdnet` packages.

Value

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

See Also

coef.dcsvm

Examples

```
data(colon)
fit <- dcsvm(colon$x, colon$y, lam2=1)
print(predict(fit, type="class", newx=colon$x[2:5, ]))
```

---

print.dcsvm *Print a DCSVM Object*

Description

Prints a summary of the dcsvm object, showing the solution paths.

Usage

```
## S3 method for class 'dcsvm'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	A fitted dcsvm object.
digits	Specifies the significant digits to use in the output. Default is max(3, getOption("digits") - 3).
...	Additional arguments to print.

Details

Print a DCSVM Object

Print a summary of the dcsvm solution paths.

This function prints a two-column matrix with columns Df and Lambda. The Df column shows the number of nonzero coefficients, and the Lambda column displays the corresponding lambda value. It is adapted from the print function in the gcdnet and glmnet packages.

Value

A two-column matrix with one column showing the number of nonzero coefficients and the other column showing the lambda values.

See Also

print.dcsvm, predict.dcsvm, coef.dcsvm, plot.dcsvm, and cv.dcsvm.

Examples

```
data(colon)
fit <- dcsvm(colon$x, colon$y)
print(fit)
```

Index

∗ **SVM**
dcsvm-package, 2
∗ **classification**
dcsvm-package, 2
∗ **data**
colon, 5
∗ **high-dimensional**
dcsvm-package, 2
∗ **set**
colon, 5

coef.cv.dcsvm, 3, *7*, *12*
coef.dcsvm, 4, *14*
colon, 5
cv.dcsvm, *3*, 6, *7*, *10*, *12*

dcsvm, *3, 4*, *6, 7*, 8, *11–14*
dcsvm-package, 2

plot.cv.dcsvm, *7*, 10
plot.dcsvm, 11
predict.cv.dcsvm, *3*, *7*, 12
predict.dcsvm, *5*, 13
print.dcsvm, 14