Package 'cossonet'

May 8, 2026

Title Sparse Nonparametric Regression for High-Dimensional Data

Version 1.0

Description Estimation of sparse nonlinear functions in nonparametric regression using component selection and smoothing. Designed for the analysis of high-dimensional data, the models support various data types, including exponential family models and Cox proportional hazards models. The methodology is based on Lin and Zhang (2006) <doi:10.1214/009053606000000722>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Imports cosso, survival, stats, MASS, glmnet, graphics

Suggests knitr, rmarkdown, testthat (>= 3.0.0), usethis (>= 2.1.5), devtools

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Jieun Shin [aut, cre]

Maintainer Jieun Shin <jieunstat@uos.ac.kr>

Repository CRAN

Date/Publication 2025-03-13 12:10:06 UTC

Contents

cossonet *Load a matrix from a file*

Description

The cossonet function implements a nonparametric regression model that estimates nonlinear components. This function can be applied to continuous, count, binary, and survival responses. To use this function, the user must specify a family, kernel function, etc. For cross-validation, the sequence vectors `lambda0` and `lambda_theta` appropriate for the input data must also be specified.

Usage

```
cossonet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "Cox"),
  wt = rep(1, ncol(x)),
  scale = TRUE,
  nbasis,
  basis.id,
  kernel = c("linear", "gaussian", "poly", "spline"),
  effect = c("main", "interaction"),
  nfold = 5,
  kparam = 1,
  lambda0 = exp(seq(log(2^{
     -10
 }), log(2^{
     10
 }), length.out = 20)),
  lambda_theta = exp(seq(log(2^{
     -10
 }), log(2^{
     10
 }), length.out = 20)),
  gamma = 0.95,
  one.std = TRUE
)
```

Arguments

x	Input matrix or data frame of n by p. x must have at least two columns ($p>1$).
y	A response vector with a continuous, binary, or count type. For survival responses, this should be a two-column matrix (or data frame) with columns called 'time' and 'status'.

family	A distribution corresponding to the response type. family="gaussian" for continuous responses, family="binomial" for binary responses, family="poisson" for count responses, and family="cox" for survival responses.
wt	The weights assigned to the explanatory variables. The default is rep(1,ncol(x)).
scale	Boolean for whether to scale continuous explanatory variables to values between 0 and 1.
nbasis	The number of "knots". If basis.id is provided, it is set to the length of basis.id.
basis.id	The index of the "knot" to select.
kernel	TThe kernel function. One of four types of linear (default), gaussian, poly, and spline.
effect	The effect of the component. main (default) is the main effect, and interaction is the two-way interaction.
nfold	The number of folds to use in cross-validation is used to determine how many subsets to divide the data into for the training and validation sets.
kparam	Parameters for Gaussian and polynomial kernel functions
lambda0	A vector of lambda0 sequences. The default is a grid of 20 values [2^{-10}, \dots, 2^{10}] on an equally spaced logarithmic scale. This may need to be adjusted based on the input data. Do not set \lambda0 as a single value.
lambda_theta	A vector of lambda sequences. The default is a grid of 20 values [2^{-10}, \dots, 2^{10}] on an equally spaced logarithmic scale. This may need to be adjusted based on the input data. Do not set lambda as a single value.
gamma	Elastic-net mixing parameter 0 \leq \gamma \leq 1. If gamma = 1, the LASSO penalty is applied, and if gamma = 0, the Ridge penalty is applied. The default is gamma = 0.95.
one.std	A logical value indicating whether to apply the "1-standard error rule." When set to TRUE, it applies to both the c-step and theta-step, selecting the simplest model within one standard error of the best model.

Value

A list containing information about the fitted model.

Examples

```
# Generate example data
set.seed(20250101)
tr = data_generation(n = 200, p = 20, SNR = 9, response = "continuous")
tr_x = tr$x
tr_y = tr$y

te = data_generation(n = 1000, p = 20, SNR = 9, response = "continuous")
te_x = te$x
te_y = te$y

# Fit the model
```

```
fit = cossonet(tr_x, tr_y, family = 'gaussian', gamma = 0.95, kernel = "spline", scale = TRUE,
      lambda0 = exp(seq(log(2^{-4}), log(2^{0}), length.out = 20)),
      lambda_theta = exp(seq(log(2^{-8}), log(2^{-6}), length.out = 20))
      )
```

`cossonet.predict` *The function* `cossonet.predict` *predicts predictive values for new data based on an object from the* `cossonet` *function.*

Description

The function `cossonet.predict` predicts predictive values for new data based on an object from the `cossonet` function.

Usage

```
cossonet.predict(model, testx)
```

Arguments

`model`	The fitted cossonet object.
`testx`	The new data set to be predicted.

Value

A list of predicted values for the new data set.

Examples

```
set.seed(20250101)
tr = data_generation(n = 200, p = 20, SNR = 9, response = "continuous")
tr_x = tr$x
tr_y = tr$y

te = data_generation(n = 1000, p = 20, SNR = 9, response = "continuous")
te_x = te$x
te_y = te$y

# Fit the model
fit = cossonet(tr_x, tr_y, family = 'gaussian', gamma = 0.95, kernel = "spline", scale = TRUE,
      lambda0 = exp(seq(log(2^{-4}), log(2^{0}), length.out = 20)),
      lambda_theta = exp(seq(log(2^{-8}), log(2^{-6}), length.out = 20))
      )

# Predict new dataset
pred = cossonet.predict(fit, te_x)
```

data_generation *The function data_generation generates an example dataset for applying the cossonet function.*

Description

The function data_generation generates an example dataset for applying the cossonet function.

Usage

```
data_generation(
  n,
  p,
  rho,
  SNR,
  response = c("continuous", "binary", "count", "survival")
)
```

Arguments

n	observation size.
p	dimension.
rho	a positive integer indicating the correlation strength for the first four informative variables.
SNR	signal-to-noise ratio.
response	the type of the response variable.

Value

a list of explanatory variables, response variables, and true functions.

Examples

```
# Generate example data
set.seed(20250101)
tr = data_generation(n = 200, p = 20, SNR = 9, response = "continuous")
tr_x = tr$x
tr_y = tr$y

te = data_generation(n = 1000, p = 20, SNR = 9, response = "continuous")
te_x = te$x
te_y = te$y
```

`metric` *The function* `metric` *provides a contingency table for the predicted class and the true class for binary classes.*

Description

The function `metric` provides a contingency table for the predicted class and the true class for binary classes.

Usage

```
metric(true, est)
```

Arguments

`true`	binary true class.
`est`	binary predicted class.

Value

a contingency table for the predicted results of binary class responses.

Examples

```
set.seed(20250101)
tr = data_generation(n = 200, p = 20, SNR = 9, response = "continuous")
tr_x = tr$x
tr_y = tr$y

te = data_generation(n = 1000, p = 20, SNR = 9, response = "continuous")
te_x = te$x
te_y = te$y

# Fit the model
fit = cossonet(tr_x, tr_y, family = 'gaussian', gamma = 0.95, kernel = "spline", scale = TRUE,
      lambda0 = exp(seq(log(2^{-4}), log(2^{0}), length.out = 20)),
      lambda_theta = exp(seq(log(2^{-8}), log(2^{-6}), length.out = 20))
      )

# Predict new dataset
pred = cossonet.predict(fit, te_x)

# Calculate the contingency table for binary class
true_var = c(rep(1, 4), rep(0, 20-4))
est_var = ifelse(fit$theta_step$theta.new > 0, 1, 0)
metric(true_var, est_var)
```

Index