Package 'plde'

October 14, 2022

Type Package

Title Penalized Log-Density Estimation Using Legendre Polynomials

Version 0.1.2

Date 2018-05-31

Author JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

Maintainer JungJun Lee <ljjoj@korea.ac.kr>

Description We present a penalized log-density estimation method using Legendre polynomials with lasso penalty to adjust estimate's smoothness. Re-expressing the logarithm of the density estimator via a linear combination of Legendre polynomials, we can estimate parameters by maximizing the penalized log-likelihood function. Besides, we proposed an implementation strategy that builds on the coordinate decent algorithm, together with the Bayesian information criterion (BIC).

License GPL (>= 2)

Encoding UTF-8

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2018-07-01 13:30:23 UTC

R topics documented:

`basic_values` *Compute basic values*

Description

Compute basic values

Usage

```
basic_values(sm)
```

Arguments

`sm`	List of plde fit

Details

`basic_values` function computes transformed variable (`sm$X_transform`), rectangular node points (`sm$nodes`) and weights (`sm$weights`) for numerical integrations, coefficient vector (`sm$coefficients`), basis matrix at node and data points (`sm$B_mat`, `sm$X_mat`), and basis mean (`sm$B_mean`).

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

See Also

`legendre_polynomial`

compute_fitted *compute_fitted*

Description

compute_fitted function gives the fitted values over the input grid points for the fixed tuning parameter λ.

Usage

```
compute_fitted(x, sm)
```

Arguments

x	grid points
sm	List of plde fit

Details

compute_fitted function computes fitted values of estimates having support for the given data by scaling back and change of variable technique. For more details, see Section 3.2 of the reference.

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

See Also

legendre_polynomial

`compute_lambdas` *Compute lambda sequence*

Description

`compute_lambdas` function gives the entire decreasing tuning parameter sequence (`sm$lambda`) on the log-scale.

Usage

```
compute_lambdas(sm)
```

Arguments

`sm`	List of plde fit

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

`fit_plde` *Fit plde for a fixed tuning parameter*

Description

`fit_plde` gives the plde fit for a fixed tuning parameter

Usage

```
fit_plde(sm)
```

Arguments

`sm`	List of plde fit

Details

This is the coordinate descent algorithm for computing $\hat{\theta}^{\lambda}$ when the penalty parameter λ is fixed. See Algorithm 1 in the reference for more details.

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

See Also

`fit_plde_sub`, `min_q_lambda`

`fit_plde_sub` *Fit plde for a fixed tuning parameter*

Description

`fit_plde_sub` function computes the updated normalizng constant (`sm$c_coefficients`), Legendre density function estimator (`sm$f`) and the negative of penalized log-likelihood function (`sm$pen_loglik`) for each iteration.

Usage

```
fit_plde_sub(sm)
```

Arguments

`sm`	List of plde fit

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

legendre_polynomial *legendre_polynomial*

Description

`legendre_polynomial` gives the Legendre polynomial design matrix over the input node points.

Usage

```
legendre_polynomial(x, sm)
```

Arguments

`x`	input node points
`sm`	List of plde fit

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

Examples

```
# clean up
rm(list = ls())
library(plde)
x = seq(-1, 1, length = 200)
L = legendre_polynomial(x, list(dimension = 10))
# Legendre polynomial basis for dimension 1 to 10
matplot(x, L, type = "l")
```

min_q_lambda *Minimization of the quadratic approximation to objective function*

Description

`min_q_lambda` function gives the coefficient vector (`sm$coefficients`) updated by the coordinate descent algorithm iteratively until the quadratic approximation to the objective function convergences.

Usage

```
min_q_lambda(sm)
```

Arguments

sm	List of plde fit

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

See Also

q_lambda, update

---

model_selection *Optimal model selection*

Description

`model_selection` function gives the optimal model over the whole plde fits based on information criterian (AIC, BIC). The optimal model is saved at `fit$optimal`.

Usage

```
model_selection(fit, method = "AIC")
```

Arguments

fit	Entire list of plde fit by all tuning parameters
method	model selection criteria. 'AIC' or 'BIC' is used. Default is 'AIC'.

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

plde *Penalized Log-density Estimation Using Legendre Polynomials*

Description

This function gives the penalized log-density estimation using Legendre polynomials.

Usage

```
plde(X, initial_dimension = 100, number_lambdas = 200,
     L = -0.9, U = 0.9, ic = 'AIC', epsilon = 1e-5, max_iterations = 1000,
     number_rectangular = 1000, verbose = FALSE)
```

Arguments

`X` Input vector, of dimension n.

`initial_dimension` Positive interger that decides initial dimension of Legendre polynomials. Default is 100.

`number_lambdas` The number of tuning parameter λ values. Default is 200.

`L` Lower bound of transformed data. Default is -0.9.

`U` Upper bound of transformed data. Default is +0.9.

`ic` Model selection criteria. 'AIC' or 'BIC' is used. Default is 'AIC'.

`epsilon` Positive real value that controls the iteration stopping criteria. In general, the smaller the value, convergence needs more iterations. Default is 1e-5.

`max_iterations` Positive integer value that decides the maximum number of iterations. Default is 1000.

`number_rectangular` Number of node points for numerical integration

`verbose` verbose

Details

The basic idea of implementation is to approximate the negative log-likelihood function by a quadratic function and then to solve penalized quadratic optimization problem using a coordinate descent algorithm. For a clear exposition of coordinate-wise updating scheme, we briefly explain a penalized univariate quadratic problem and its solution expressed as soft-thresholding operator `soft_thresholding`. We use this univariate case algorithm to update parameter vector coordinatewisely to find a minimizer.

Value

A list contains the whole fits of all tuning parameter λ sequence. For example, `fit$sm[[k]]` indicates the fit of k th lambda.

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

Source

This package is built on R version 3.4.2.

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." Journal of statistical software 33.1 (2010): 1.

See Also

basic_values, compute_lambdas, fit_plde, model_selection

Examples

```
# clean up
rm(list = ls())
library(plde)
Eruption = faithful$eruptions
Waiting = faithful$waiting
n = length(Eruption)
# fit PLDE
fit_Eruption = plde(Eruption, initial_dimension = 30, number_lambdas = 50)
fit_Waiting = plde(Waiting, initial_dimension = 30, number_lambdas = 50)
x_Eruption = seq(min(Eruption), max(Eruption), length = 100)
x_Waiting = seq(min(Waiting), max(Waiting), length = 100)
fhat_Eruption = compute_fitted(x_Eruption, fit_Eruption$sm[[fit_Eruption$number_lambdas]])
fhat_Waiting = compute_fitted(x_Waiting, fit_Waiting$sm[[fit_Waiting$number_lambdas]])
# display layout
par(mfrow = c(2, 2), oma=c(0,0,2,0), mar = c(4.5, 2.5, 2, 2))
#========================================
# Eruption
#========================================
col_index = rainbow(fit_Eruption$number_lambdas)
plot(x_Eruption, fhat_Eruption, type = "n", xlab = "Eruption", ylab = "", main = "")
# all fit plot
for(i in 1 : fit_Eruption$number_lambdas)
{
   fhat = compute_fitted(x_Eruption, fit_Eruption$sm[[i]])
   lines(x_Eruption, fhat, lwd = 0.5, col = col_index[i])
}
k_Eruption = density(Eruption, bw = 0.03)
lines(k_Eruption$x, k_Eruption$y / 2, lty = 2)

# optimal model
```

```
hist_col = rgb(0.8,0.8,0.8, alpha = 0.6)
hist(Eruption, nclass = 20, freq = FALSE, xlim = c(1.1, 5.9),
     col = hist_col, ylab = "", main = "", ylim = c(0, 1.2))
fhat_optimal_Eruption = compute_fitted(x_Eruption, fit_Eruption$optimal)
lines(x_Eruption, fhat_optimal_Eruption, col = "black", lwd = 2)
#========================================
# Waiting
#========================================
col_index = rainbow(fit_Waiting$number_lambdas)
plot(x_Waiting, fhat_Waiting, type = "n", xlab = "Waiting", ylab = "", main = "")
# all fit plot
for(i in 1 : fit_Waiting$number_lambdas)
{
   fhat = compute_fitted(x_Waiting, fit_Waiting$sm[[i]])
   lines(x_Waiting, fhat, lwd = 0.5, col = col_index[i])
}
k_Waiting = density(Waiting, bw = 1)
lines(k_Waiting$x, k_Waiting$y / 2, lty = 2)

# optimal model
hist_col = rgb(0.8,0.8,0.8, alpha = 0.6)
hist(Waiting, nclass = 20, freq = FALSE, xlim = c(40, 100),
     col = hist_col, ylab = "", main = "", ylim = c(0, 0.055))
fhat_optimal_Waiting = compute_fitted(x_Waiting, fit_Waiting$optimal)
lines(x_Waiting, fhat_optimal_Waiting, col = "black", lwd = 2)
```

---

q_lambda *Compute quadratic approximation objective function*

Description

q_lambda function computes quadratic approximation of the objective function.

Usage

```
q_lambda(sm)
```

Arguments

sm	List of plde fit

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

soft_thresholding *Soft thresholding operator*

Description

soft_thresholding gives the soft threshold value of y given the threshold. When threshold increasing, y shrinks to zero.

Usage

```
soft_thresholding(y, threshold)
```

Arguments

y	input real value
threshold	threshold value

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

Examples

```
# clean up
rm(list = ls())
library(plde)
# soft thresholding operater
soft_thresholding(3, 1)
soft_thresholding(-3, 1)
# if the threshold value is large enough, it shrinks to zero
soft_thresholding(-3, 4)
soft_thresholding(3, 4)
# Plot of the soft thresholding operater
y = seq(-3, 3, length = 100)
st = NULL
for (i in 1 : length(y))
   st[i] = soft_thresholding(y[i], 1)
plot(y, y, col = "gray", type = "l", ylab = "ST")
lines(y, st, col = "blue")
```

update *Update the Legendre polynomial coefficient vector*

Description

`update` function finds the minimizer of an univariate quadratic approximation objective function for each coefficient coordinate-wise.

Usage

```
update(sm)
```

Arguments

`sm`	List of plde fit

Author(s)

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim, Ja-yong Koo

References

JungJun Lee, Jae-Hwan Jhong, Young-Rae Cho, SungHwan Kim and Ja-Yong Koo. "Penalized Log-density Estimation Using Legendre Polynomials." Submitted to Communications in Statistics - Simulation and Computation (2017), in revision.

Index