Package: asm (via r-universe)

August 26, 2024

Type Package

Title Optimal Convex M-Estimation for Linear Regression via Antitonic Score Matching

Version 0.2.0

License GPL (>= 3)

Description Performs linear regression with respect to a data-driven convex loss function that is chosen to minimize the asymptotic covariance of the resulting M-estimator. The convex loss function is estimated in 5 steps: (1) form an initial OLS (ordinary least squares) or LAD (least absolute deviation) estimate of the regression coefficients; (2) use the resulting residuals to obtain a kernel estimator of the error density; (3) estimate the score function of the errors by differentiating the logarithm of the kernel density estimate; (4) compute the L2 projection of the estimated score function onto the set of decreasing functions; (5) take a negative antiderivative of the projected score function estimate. Newton's method (with Hessian modification) is then used to minimize the convex empirical risk function. Further details of the method are given in Feng et al. (2024) <doi:10.48550/arXiv.2403.16688>.

Encoding UTF-8

Depends R (>= 3.1)

Imports fdrtool, pracma, Iso, MASS, quantreg

RoxygenNote 7.3.1

NeedsCompilation no

Author Yu-Chun Kao [aut], Min Xu [aut, cre], Oliver Y. Feng [aut], Richard J. Samworth [aut]

Maintainer Min Xu <min.cut@gmail.com>

Date/Publication 2024-05-11 02:23:05 UTC

Repository https://nineisprime.r-universe.dev

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

RemoteRef HEAD

RemoteSha 1dbe469e17a6416595be413f85ec0bdbf45d5b29

Contents

asm 2
asm.fit 3
coef.asm 4
confint.asm 5
plot.asm 6
predict.asm 7
print.asm 8
print.summary.asm 8
residuals.asm 9
summary.asm 10

Index **11**

`asm` *Linear regression via antitonic score matching*

Description

Performs linear regression with a data-driven convex loss function

Usage

```
asm(formula, data = NULL, ...)
```

Arguments

`formula`	regression formula
`data`	input data frame
`...`	additional arguments for asm.fit

Value

`asm` class object containing the following components:

`betahat`: vector of estimated coefficients

`std_errs`: vector of standard errors of the estimated coefficients

`fitted.values`: fitted values

`residuals`: residuals

`zvals`: z-values

`sig_vals`: p-values

`info_asm`: antitonic information

`I_mat`: estimated antitonic information matrix

`Cov_mat`: covariance matrix of the estimated coefficients

`psi`: estimated antitonic score function

Examples

```
asm(mpg ~ cyl + hp + disp, data=mtcars)
```

`asm.fit` *Fit a linear regression model via antitonic score matching*

Description

Performs linear regression via M-estimation with respect to a data-driven convex loss function

Usage

```
asm.fit(
  X,
  Y,
  betapilot = "OLS",
  alt_iter = 1,
  intercept.selection = "mean",
  k = 3000,
  max_iter = 65,
  kernel_pts = 2^15,
  bw = "nrd0",
  kernel = "gaussian",
  verbose = FALSE,
  ...
)
```

Arguments

`X`	design matrix
`Y`	response vector
`betapilot`	initial estimate of the regression coefficients: can be "LAD", "OLS" or a vector of coefficients
`alt_iter`	number of iterations of the alternating procedure: when alt_iter == 1, this function is equivalent to asm_regression
`intercept.selection`	mean or median of the residuals if intercept.selection == "median", then the standard error of the intercept estimate is set to NA
`k`	the density quantile function is evaluated at (0, 1/k, 2/k, ..., 1)

max_iter	maximum number of iterations for the damped Newton–Raphson algorithm when minimizing the convex loss function
kernel_pts	number of points at which the kernel density estimate is evaluated, i.e. the parameter "n" in density()
bw	bandwidth for kernel density estimation i.e. the parameter "bw" in density()
kernel	kernel for kernel density estimation i.e. the parameter "kernel" in density()
verbose	logical; if TRUE, print optimization progress
...	additional arguments to ensure compatibility with generic functions

Value

asm class object containing the following components:

betahat: vector of estimated coefficients

std_errs: vector of standard errors of the estimated coefficients

fitted.values: fitted values

residuals: residuals

zvals: z-values

sig_vals: p-values

info_asm: antitonic information

I_mat: estimated antitonic information matrix

Cov_mat: asymptotic covariance matrix of the estimated coefficients

psi: estimated antitonic score function

Examples

```
n <- 1000 ; d <- 2
X <- matrix(rnorm(n * d), n, d)
Y <- X %*% c(2, 3) + rnorm(n) # no intercept!
asm.fit(X,Y)
```

---

coef.asm *Coefficients of an* asm *regression model*

Description

Outputs the coefficients of a fitted asm regression model

Usage

```
## S3 method for class 'asm'
coef(object, ...)
```

Arguments

object	asm object
...	additional arguments to ensure compatibility with the generic function coef()

Value

vector of coefficients of the asm regression model

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
coef(model)
```

---

confint.asm *Confidence intervals for coefficients in an* asm *regression model*

Description

Computes confidence intervals for individual regression coefficients based on a fitted asm regression model

Usage

```
## S3 method for class 'asm'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	asm object
parm	parameters to calculate confidence intervals
level	confidence level
...	additional arguments to ensure compatibility with the generic function confint()

Value

matrix of confidence intervals for the regression coefficients

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
confint(model)
```

plot.asm *Generate diagnostic plots for an* asm *regression model*

Description

Generates plots of residuals vs fitted values, and the estimated convex loss and antitonic score functions based on a fitted asm regression model

Usage

```
## S3 method for class 'asm'
plot(
  x,
  which = c(1, 2, 3),
  caption = list("Residuals vs fitted", "Convex loss function",
    "Antitonic score function"),
  extend.ylim.f = 0.08,
  id.n = 3,
  labels.id = rownames(x$residuals),
  label.pos = c(4, 2),
  ext.xlim.f = 0.08,
  grid.length.f = 10,
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  ...
)
```

Arguments

x	asm object
which	a subset of the plots to be displayed
caption	a list of captions for the plots
extend.ylim.f	factor to extend the y-axis limits for the residuals vs fitted plot
id.n	number of residuals to label in the residuals vs fitted plot
labels.id	labels for the residuals in the residuals vs fitted plot
label.pos	position of the labels in the residuals vs fitted plot
ext.xlim.f	factor to extend the x-axis limits for the convex loss and antitonic score function plots
grid.length.f	the number of grid points for the convex loss plot is defined as grid.length.f * length(x$residuals)
ask	logical; if TRUE, the user is asked before each plot
...	additional arguments to ensure compatibility with the generic function plot()

Value

No return value

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
plot(model)
```

`predict.asm` *Predict new responses using an* `asm` *regression model.*

Description

Outputs predictions on new test data based on a fitted `asm` regression model. Also returns a confidence interval around the conditional mean (if interval = "confidence") or predicted response (if interval = "prediction").

Usage

```
## S3 method for class 'asm'
predict(object, newdata = NULL, interval = "none", level = 0.95, ...)
```

Arguments

`object`	asm object
`newdata`	new data frame
`interval`	type of interval calculation, either "none", "confidence" or "prediction". Default is "none".
`level`	confidence level
`...`	additional arguments to ensure compatibility with the generic function predict()

Value

matrix of predicted values * if interval = "none", the matrix has one column of predicted values * if interval = "confidence" or "prediction", the matrix has three columns: predicted value, lower bound, and upper bound

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
predict(model, newdata = data.frame(cyl = 4, hp = 121, disp = 80), interval = "prediction")

n <- 1000
X <- rnorm(n)
beta <- 2
Y <- beta*X + rt(n,df=3)
Mod <- asm(Y ~ X)
predict(Mod, newdata = data.frame(X = 1), interval = "prediction")
```

print.asm *Short description of a fitted* asm *regression model*

Description

Outputs estimated coefficients and standard errors

Usage

```
## S3 method for class 'asm'
print(x, ...)
```

Arguments

x	asm object
...	additional arguments to ensure compatibility with the generic function print()

Value

No return value, called for its side effect

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
print(model)
```

print.summary.asm *Print summary of the* asm *regression model*

Description

Prints the summary of a fitted asm regression model

Usage

```
## S3 method for class 'summary.asm'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  concise = FALSE,
  ...
)
```

Arguments

x	summary.asm object
digits	number of digits to print
signif.stars	logical; if TRUE, 'significance stars' are printed
concise	logical; if TRUE, the output is concise
...	additional arguments to ensure compatibility with the generic function print()

Value

No return value

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
print(summary(model))
```

---

residuals.asm *Residuals from an* asm *regression model*

Description

Outputs the residuals (on the training data) from a fitted asm regression model

Usage

```
## S3 method for class 'asm'
residuals(object, ...)
```

Arguments

object	asm object
...	additional arguments to ensure compatibility with the generic function residuals()

Value

vector of residuals from the asm regression model

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
residuals(model)
```

summary.asm *Summary of an* asm *regression model*

Description

Outputs estimated coefficients, standard errors and p-values based on a fitted asm regression model

Usage

```
## S3 method for class 'asm'
summary(object, ...)
```

Arguments

object	asm object
...	additional arguments to ensure compatibility with the generic function summary()

Value

summary.asm class object containing the following components:

coefficients: estimated coefficients, standard errors, z-values and p-values

residuals: residuals of the fitted model

call: call to the asm function

Examples

```
model = asm(mpg ~ cyl + hp + disp, data=mtcars)
summary(model)
```

Index

asm, 2
asm.fit, 3

coef.asm, 4
confint.asm, 5

plot.asm, 6
predict.asm, 7
print.asm, 8
print.summary.asm, 8

residuals.asm, 9

summary.asm, 10