Package: expectile (via r-universe)

October 7, 2024

Version 0.3.2

Depends R (>= 2.3.0)

Imports R.methodsS3 (>= 1.2.2)

Title Modelling of Expectiles

Author Pratyaksha Wirapati, Henrik Bengtsson, Mark Robinson

Maintainer Henrik Bengtsson <henrikb@braju.com>

Description Methods for fitting a simplex or polyhedral cone to multivariate data, for doing expectile regression and skyline/baseline estimation.

License LGPL (== 2.1)

LazyLoad yes

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

RemoteUrl https://github.com/HenrikBengtsson/expectile

RemoteRef master

RemoteSha f85c066661802c3a50d5bd05802afedad5ba4614

Contents

`sfit2.matrix` *Fit a simplex or polyhedral cone to multivariate data*

Description

Fit a simplex or polyhedral cone to multivariate data by decomposing data PxN `matrix` $Y = XB + E$, where X is a PxM `matrix`, B is a "mostly non-negative" MxN `matrix`, and E is a PxN `matrix` of noise, all with $M - 1 \leq P$.

Usage

```
## S3 method for class 'matrix'
sfit2(y, M=dim(y)[1] + 1, w=rep(1, dim(y)[2]), lambda=2, alpha=0.05,
 family=c("biweight", "huber", "normal"), robustConst=4.685, tol=0.001, maxIter=60,
 Rtol=1e-07, priorX=NULL, priorW=NULL, initX=NULL, fitCone=FALSE, verbose=FALSE, ...)
```

Arguments

y	A PxN matrix (or data.frame) containing P variables and N observations in R^N.
M	Number of vertices, M-1 <= P.
.	
w	An optional vector in [0,1] of length N specifying weight for each observation.
lambda	A scalar vertex assigment parameter in [1,Inf).
alpha	A double in [0,1] specifying the desired expectile.
family	A character string specifying the ....
robustConst	A double constant multiplier of MAR scale estimate.
tol	A positive double tolerance for expectile estimation.
maxIter	The maximum number of iterations in estimation step.
Rtol	A postive double tolerance in linear solve, before a vertex is ignored.
priorX, priorW	(Optional) Prior simplex PxM matrix and M vertex weights. An Inf weight corresponds to a fixed vertex. If NULL, no priors are used.
initX	(Optional) An initial simplex PxM matrix ('X'). If NULL, the initial simplex is estimated automatically.
fitCone	If TRUE, the first vertex is treated as an apex and the opposite face has its own residual scale estimator.
verbose	if TRUE, iteration progress is printed to standard error.
...	Not used.

Details

Given multidimensional data matrix Y with P rows (variables) and N columns (observations), decompose Y into two matrices, X (P-by-M) and B (M-by-N) as $Y = XB + E$, where P may be larger than M-1.

In simplex fitting mode, B_j for each observation sums to one, and mostly non-negative. The columns of X are the estimated vertices of the simplex enclosing most points.

In cone fitting mode, the first column of X is apex of the cone, while the others are directions of the rays emanating from the apex, with the vector norms standardized to one. The first row of B is always equal to one, and the remaining rows are mostly non-negative. They don't necessarily sum to one.

Value

Returns a named list structure elements:

X	the fitted simplex, as a PxM matrix.
B	Affine coefficients, as an MxN matrix.

Author(s)

Algorithm and native code by Pratyaksha (Asa) Wirapati. R interface by Henrik Bengtsson.

References

[1] P. Wirapati, & T. Speed, *Fitting polyhedrial cones and simplices to multivariate data points*, Walter and Eliza Hall Institute of Medical Research, December 30, 2001.
[2] P. Wirapati and T. Speed, *An algorithm to fit a simplex to a set of multidimensional points*, Walter and Eliza Hall Institute of Medical Research, January 15, 2002.

Examples

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Example with simulated data
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Number of observations
n <- 20000

# Offset and cross talk
a0 <- c(50,300)
A <- matrix(c(1,0.2,0.5,1), nrow=2, ncol=2)  # cross-talk

# the true signal is joint gamma
z <- matrix(rgamma(2*n, shape=0.25, scale=100), nrow=2, ncol=n)

# Observed signal plus Gaussian error
eps <- matrix(rnorm(2*n, mean=0, sd=10), nrow=2, ncol=n)
y <- A %*% z + a0 + eps


layout(matrix(1:4, nrow=2, byrow=TRUE))
par(mar=c(5,4,2,2)+0.1)
lim <- c(0,1000)
xlab <- expression(y[1])
ylab <- expression(y[2])

for (withPrior in c(FALSE, TRUE)) {
  if (withPrior) {
    priorX <- matrix(c(a0, 0,0, 0,0), nrow=2, ncol=3)
    priorW <- c(Inf,0,0)
    priorW <- c(+100,0,0)
    # Fit cone
    fit <- fitCone(y, priorX=priorX, priorW=priorW)
```

```
##    stopifnot(identical(fit$X[,1], a0))
  } else {
    # Fit cone
    fit <- fitCone(y)
    fit0 <- fit
  }

  cat("Estimated cone:\n")
  print(fit$X)

  plot(t(y), pch=".", xlim=lim, ylim=lim, xlab=xlab, ylab=ylab)
  points(fit, pch=19, cex=1.5, col="#aaaaaa")
  radials(fit, col="#aaaaaa", lwd=2)
  drawApex(fit, pch=19, cex=1, col="tomato")
  lines(fit, col="tomato", lwd=2)


  # The rectified data points
  xlab <- expression(hat(x)[1])
  ylab <- expression(hat(x)[2])
  plot(t(fit$Beta[2:3,]), pch=".", xlab=xlab, ylab=ylab)
  points(0,0, pch=19, cex=1.5, col="tomato") # the apex
  lines(c(0,0,lim[2]), c(lim[2],0,0), lwd=2, col="tomato")
}
```

Index