Package: VBel (via r-universe)

October 28, 2024

Type Package

Title Variational Bayes for Fast and Accurate Empirical Likelihood Inference

Version 1.0.1

Date 2024-05-28

Description Computes the Gaussian variational approximation of the Bayesian empirical likelihood posterior. This is an implementation of the function found in Yu, W., & Bondell, H. D. (2023) <doi:10.1080/01621459.2023.2169701>.

License GPL (>= 3)

Imports Rcpp (>= 1.0.12), stats

LinkingTo Rcpp, RcppEigen

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/jlimrasc/VBel

BugReports https://github.com/jlimrasc/VBel/issues

Suggests mvtnorm, testthat (>= 3.0.0)

Config/testthat/edition 3

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

RemoteUrl https://github.com/jlimrasc/vbel

RemoteRef HEAD

RemoteSha bf6f67f2952e5e5ce3f58ad4f5f17d3ab5cfc217

Contents

VBel-package *Variational Bayes for Fast and Accurate Empirical Likelihood Inference*

Description

Computes the Gaussian variational approximation of the Bayesian empirical likelihood posterior. This is an implementation of the function found in Yu, W., & Bondell, H. D. (2023) <doi:10.1080/01621459.2023.2169701>.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Wei-Chang Yu [aut] (<https://orcid.org/0000-0002-0399-3779>), Jeremy Lim [cre, aut]

Maintainer: Jeremy Lim <jeremy.lim@unimelb.edu.au>

References

https://www.tandfonline.com/doi/abs/10.1080/01621459.2023.2169701

See Also

compute_AEL() for choice of R and/or C++ computation of AEL

compute_GVA() for choice of R and/or C++ computation of GVA

diagnostic_plot() for verifying convergence of computed GVA data

Examples

```
#ansGVARcppPure <- compute_GVA(mu, C_0, h, delthh, delth_logpi, z, lam0, rho,
#elip, a, iters, iters2, fullCpp = TRUE)
#diagnostic_plot(ansGVARcppPure)
```

compute_AEL *Compute the Adjusted Empirical Likelihood*

Description

Evaluates the AEL for a given data set, moment conditions and parameter values

Usage

```
compute_AEL(th, h, lam0, a, z, iters, useR_forz, returnH)
```

Arguments

`th`	Vector or scalar theta
`h`	User-defined function, outputs array
`lam0`	Initial vector for lambda
`a`	Scalar constant
`z`	n-1 by d matrix
`iters`	Number of iterations using Newton-Raphson for estimation of lambda (default: 500)
`useR_forz`	Bool whether to calculate the function first in R (True) or call the function in C (False) (default: True)
`returnH`	Whether to return calculated values of h, H matrix and lambda

Value

A numeric value for the Adjusted Empirical Likelihood function computed evaluated at a given theta value

Author(s)

Wei Chang Yu, Jeremy Lim

References

Yu, W., & Bondell, H. D. (2023). Variational Bayes for Fast and Accurate Empirical Likelihood Inference. Journal of the American Statistical Association, 1–13. doi:10.1080/01621459.2023.2169701

Examples

```
# Generate toy variables
set.seed(1)
x     <- runif(30, min = -5, max = 5)
elip  <- rnorm(30, mean = 0, sd = 1)
y     <- 0.75 - x + elip
```

```
# Set initial values for AEL computation
lam0  <- matrix(c(0,0), nrow = 2)
th    <- matrix(c(0.8277, -1.0050), nrow = 2)
a     <- 0.00001
iters <- 10

# Define Dataset and h-function
z <- cbind(x, y)
h <- function(z, th) {
    xi      <- z[1]
    yi      <- z[2]
    h_zith  <- c(yi - th[1] - th[2] * xi, xi*(yi - th[1] - th[2] * xi))
    matrix(h_zith, nrow = 2)
}
ansAELRcpp <- compute_AEL(th, h, lam0, a, z, iters, useR_forz = TRUE)
```

compute_GVA — *Compute the Full-Covariance Gaussian VB Empirical Likelihood Posterior*

Description

Requires a given data set, moment conditions and parameter values and returns a list of the final mean and variance-covariance along with an array of the in-between calculations at each iteration for analysis of convergence

Usage

```
compute_GVA(
  mu,
  C,
  h,
  delthh,
  delth_logpi,
  z,
  lam0,
  rho,
  elip,
  a,
  iters,
  iters2,
  fullCpp,
  verbosity
)
```

Arguments

`mu`	Column vector, initial value of Gaussian VB mean

C	Lower Triangular Matrix, initial value of Gaussian VB Cholesky
h	User-defined moment-condition function, outputs a k x 1 matrix containing the kth row of h. Function must take two arguments: zi and theta for h(zi,th)
delthh	User defined function, outputs k x p Jacobian matrix of h(zi,th) with respect to theta
delth_logpi	User-defined function, outputs p x 1 matrix, derivative of log prior function
z	Data matrix, n-1 x d matrix
lam0	Initial guess for lambda, k x 1 matrix
rho	Scalar (between 0 to 1, reccomended to be close to 1) ADADELTA accumulation constant
elip	Scalar numeric stability constant. Should be specified with a small value
a	Scalar AEL constant, must be >0, small values recommended
iters	Number of iterations for GVA (default:10,000)
iters2	Number of iterations for Log AEL (default:500)
fullCpp	Bool whether to calculate the main section in cpp (TRUE) or only partially (FALSE, doing all the AEL calculations in R before handing values to cpp) (default: TRUE)
verbosity	Integer for how often to print updates on current iteration number (default:500)

Value

A list containing:

1. mu_FC: VB Posterior Mean at final iteration. A vector of size p x 1
2. C_FC: VB Posterior Variance-Covariance (Cholesky) at final iteration. A lower-triangular matrix of size p x p
3. mu_FC_arr: VB Posterior Mean for each iteration. A matrix of size p x (iters + 1)
4. C_FC_arr: VB Posterior Variance-Covariance (Cholesky) for each iteration. An array of size p x p x (iters + 1)

Author(s)

Wei Chang Yu, Jeremy Lim

References

Yu, W., & Bondell, H. D. (2023). Variational Bayes for Fast and Accurate Empirical Likelihood Inference. Journal of the American Statistical Association, 1–13. doi:10.1080/01621459.2023.2169701

Examples

```
set.seed(1)
x    <- runif(30, min = -5, max = 5)
elip <- rnorm(30, mean = 0, sd = 1)
y    <- 0.75 - x + elip
```

```
lam0 <- matrix(c(0,0), nrow = 2)
th   <- matrix(c(0.8277, -1.0050), nrow = 2)
a    <- 0.00001
z    <- cbind(x, y)
h    <- function(z, th) {
    xi     <- z[1]
    yi     <- z[2]
    h_zith <- c(yi - th[1] - th[2] * xi, xi*(yi - th[1] - th[2] * xi))
    matrix(h_zith, nrow = 2)
}

delthh <- function(z, th) {
    xi <- z[1]
    matrix(c(-1, -xi, -xi, -xi^2), 2, 2)
}

n     <- 31
reslm <- lm(y ~ x)
mu    <- matrix(unname(reslm$coefficients),2,1)
C_0   <- unname(t(chol(vcov(reslm))))

delth_logpi <- function(theta) { -0.0001 * mu }
elip        <- 10^-5
iters       <- 10
iters2      <- 50
rho         <- 0.9

# -----------------------------
# Main
# -----------------------------
ansGVARcppHalf <-compute_GVA(mu, C_0, h, delthh, delth_logpi, z, lam0,
rho, elip, a, iters, iters2, fullCpp = FALSE)
ansGVARcppPure <-compute_GVA(mu, C_0, h, delthh, delth_logpi, z, lam0,
rho, elip, a, iters, iters2, fullCpp = TRUE)
```

`diagnostic_plot` *Check the convergence of a data set computed by* `compute_GVA`

Description

Plots mu and variance in a time series plot to check for convergence of the computed data (i.e. Full-Covariance Gaussian VB Empirical Likelihood Posterior)

Usage

```
diagnostic_plot(dataList, muList, cList)
```

Arguments

dataList	Named list of data generated from compute_GVA
muList	Array of indices of mu_arr to plot. (default:all)
cList	Matrix of indices of variance to plot, 2xn matrix, each row is (col,row) of variance matrix

Value

Matrix of variance of C_FC

Examples

```
# Generate toy variables
seedNum <- 100
set.seed(seedNum)
n       <- 100
p       <- 10
lam0    <- matrix(0, nrow = p)

# Calculate z
mean    <- rep(1, p)
sigStar <- matrix(0.4, p, p) + diag(0.6, p)
z       <- mvtnorm::rmvnorm(n = n-1, mean = mean, sigma = sigStar)

# Calculate intermediate variables
zbar    <- 1/(n-1) * matrix(colSums(z), nrow = p)
sumVal  <- matrix(0, nrow = p, ncol = p)
for (i in 1:p) {
zi      <- matrix(z[i,], nrow = p)
sumVal  <- sumVal + (zi - zbar) %*% matrix(zi - zbar, ncol = p)
}
sigHat  <- 1/(n-2) * sumVal

# Initial values for GVA
mu_0    <- matrix(zbar, p, 1)
C_0     <- 1/sqrt(n) * t(chol(sigHat))

# Define h-function
h       <- function(zi, th) { matrix(zi - th, nrow = 10) }

# Define h-gradient function
delthh  <- function(z, th) { -diag(p) }

# Set other initial values
delth_logpi <- function(theta) {-0.0001 * theta}
elip    <- 10^-5
T       <- 5 # Number of iterations for GVA
T2      <- 5 # Number of iterations for AEL
rho     <- 0.9
a       <- 0.00001
```

```
ansGVA <-compute_GVA(mu_0, C_0, h, delthh, delth_logpi, z, lam0, rho, elip,
a, T, T2, fullCpp = TRUE)

diagnostic_plot(ansGVA)
diagnostic_plot(ansGVA, muList = c(1,4))
diagnostic_plot(ansGVA, cList = matrix(c(1,1, 5,6, 3,3), ncol = 2))
```

Index