

Package: VBel (via r-universe)

December 5, 2024

Type Package

Title Variational Bayes for Fast and Accurate Empirical Likelihood Inference

Version 1.1.0

Date 2024-12-05

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](https://doi.org/10.1080/01621459.2023.2169701)>.

License GPL (>= 3)

Imports Rcpp (>= 1.0.12), stats

LinkingTo Rcpp, RcppEigen

Encoding UTF-8

RoxygenNote 7.3.2

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

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

Suggests mvtnorm, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2024-12-05 05:20:02 UTC

Contents

| | |
|---------------------------|---|
| VBel-package | 2 |
| compute_AEL | 3 |
| compute_GVA | 5 |
| diagnostic_plot | 7 |

| | |
|--------------|---|
| VBel-package | <i>Variational Bayes for Fast and Accurate Empirical Likelihood Inference</i> |
|--------------|---|

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:

```
Package:          VBel
Type:             Package
Title:            Variational Bayes for Fast and Accurate Empirical Likelihood Inference
Version:          1.1.0
Date:             2024-12-05
Authors@R:        c( person("Weichang", "Yu", , "weichang.yu@unimelb.edu.au", role = c("aut"), comment = c(ORCID
Description:      Computes the Gaussian variational approximation of the Bayesian empirical likelihood posterior. Th
License:          GPL (>= 3)
Imports:          Rcpp (>= 1.0.12), stats
LinkingTo:        Rcpp, RcppEigen
Encoding:         UTF-8
Roxygen:          list(markdown = TRUE)
RoxygenNote:      7.3.2
URL:              https://github.com/jlimrasc/VBel
BugReports:       https://github.com/jlimrasc/VBel/issues
Suggests:         mvtnorm, testthat (>= 3.0.0)
Config/testthat/edition: 3
Author:           Weichang Yu [aut] (<https://orcid.org/0000-0002-0399-3779>), Jeremy Lim [cre, aut]
Maintainer:       Jeremy Lim <jeremy.lim@unimelb.edu.au>
Archs:            x64
```

Index of help topics:

| | |
|-----------------|--|
| VBel-package | Variational Bayes for Fast and Accurate Empirical Likelihood Inference |
| compute_AEL | Compute the Adjusted Empirical Likelihood |
| compute_GVA | Compute the Full-Covariance Gaussian VB Empirical Likelihood Posterior |
| diagnostic_plot | Check the convergence of a data set computed by 'compute_GVA' |

Author(s)

Weichang 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 Log-Adjusted Empirical Likelihood (AEL) (Chen, Variyath, and Abraham 2008) for a given data set, moment conditions and parameter values. The AEL function is formulated as

$$\log \text{AEL}(\boldsymbol{\theta}) = \max_{\mathbf{w}'} \sum_{i=1}^{n+1} \log(w'_i),$$

where \mathbf{z}_{n+1} is a pseudo-observation that satisfies

$$h(\mathbf{z}_{n+1}, \boldsymbol{\theta}) = -\frac{a_n}{n} \sum_{i=1}^n h(\mathbf{z}_i, \boldsymbol{\theta})$$

for some constant $a_n > 0$ that may (but not necessarily) depend on n , and $\mathbf{w}' = (w'_1, \dots, w'_n, w'_{n+1})$ is a vector of probability weights that define a discrete distribution on $\{\mathbf{z}_1, \dots, \mathbf{z}_n, \mathbf{z}_{n+1}\}$, and are subject to the constraints

$$\sum_{i=1}^{n+1} w'_i h(\mathbf{z}_i, \boldsymbol{\theta}) = 0, \quad \text{and} \quad \sum_{i=1}^{n+1} w'_i = 1.$$

Here, the maximizer $\tilde{\mathbf{w}}$ is of the form

$$\tilde{w}_i = \frac{1}{n+1} \frac{1}{1 + \lambda_{\text{AEL}}^\top h(\mathbf{z}_i, \boldsymbol{\theta})},$$

where λ_{AEL} satisfies the constraints

$$\frac{1}{n+1} \sum_{i=1}^{n+1} \frac{h(\mathbf{z}_i, \boldsymbol{\theta})}{1 + \lambda_{\text{AEL}}^\top h(\mathbf{z}_i, \boldsymbol{\theta})} = 0, \quad \text{and} \quad \frac{1}{n+1} \sum_{i=1}^{n+1} \frac{1}{1 + \lambda_{\text{AEL}}^\top h(\mathbf{z}_i, \boldsymbol{\theta})} = 1.$$

Usage

```
compute_AEL(th, h, lam0, a, z, iters = 500, returnH = FALSE)
```

Arguments

| | |
|---------|--|
| th | p x 1 parameter vector to evaluate the AEL function at |
| h | User-defined moment-condition function. Note that output should be an (n-1) x K matrix where K is necessarily \geq p |
| lam0 | Initial vector for Lagrange multiplier lambda |
| a | Positive scalar adjustment constant |
| z | (n-1) x d data matrix. Note that $\{z_i\}_{i=1}^{n-1}$ is a sequence of d-dimensional data vectors |
| iters | Number of iterations using Newton-Raphson for estimation of lambda. Default: 500 |
| returnH | Whether to return calculated values of h, H matrix and lambda. Default: 'FALSE' |

Details

Note that theta (th) is a p-dimensional vector, h is a K-dimensional vector and $K \geq p$

Value

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

Author(s)

Weichang Yu, Jeremy Lim

References

Chen, J., Variyath, A. M., and Abraham, B. (2008), "Adjusted Empirical Likelihood and its Properties," *Journal of Computational and Graphical Statistics*, 17, 426–443. Pages 2,3,4,5,6,7 [doi:10.1198/106186008X321068](https://doi.org/10.1198/106186008X321068)

Examples

```
# Generating 30 data points from a simple linear-regression model
set.seed(1)
x   <- runif(30, min = -5, max = 5)
vari <- rnorm(30, mean = 0, sd = 1)
y   <- 0.75 - x + vari
```

```

z <- cbind(x, y)

lam0 <- matrix(c(0,0), nrow = 2)
th <- matrix(c(0.8277, -1.0050), nrow = 2)

# Specify AEL constant and Newton-Rhapson iteration
a <- 0.00001
iters <- 10

# Specify moment condition functions for linear regression
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)
}
result <- compute_AEL(th, h, lam0, a, z, iters)

```

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(
  mu0,
  C0,
  h,
  delthh,
  delth_logpi,
  z,
  lam0,
  rho,
  epsil,
  a,
  SDG_iters = 10000,
  AEL_iters = 500,
  verbosity = 500
)

```

Arguments

| | |
|----------------|---|
| μ_0 | $p \times 1$ initial vector of Gaussian VB mean |
| C_0 | $p \times p$ initial lower triangular matrix of Gaussian VB Cholesky |
| h | User-defined moment-condition function. Note that output should be an $(n-1) \times K$ matrix where K is necessarily $\geq p$ |
| $delthh$ | User-defined first-order derivative of moment-condition function. Note that output should be a $K \times p$ matrix of $h(z_i, \theta)$ with respect to θ |
| $delth_logpi$ | User-defined first-order derivative of log-prior function. Note that output should be a $p \times 1$ vector |
| z | Data matrix, $n-1 \times d$ matrix |
| λ_0 | Initial vector for Lagrange multiplier λ |
| ρ | Scalar numeric beteen 0 to 1. ADADELTA accumulation constant |
| ϵ | Positive numeric scalar stability constant. Should be specified with a small value |
| a | Positive scalar adjustment constant. For more accurate calculations, small values are recommended |
| SDG_iters | Number of Stochastic Gradient-Descent iterations for optimising μ and C . Default: 10,000 |
| AEL_iters | Number of iterations using Newton-Raphson for optimising AEL λ . Default: 500 |
| $verbosity$ | Integer for how often to print updates on current iteration number. Default:500 |

Value

A list containing:

1. μ_FC : VB Posterior Mean at final iteration. A vector of size $p \times 1$
2. C_FC : VB Posterior Variance-Covariance (Cholesky) at final iteration. A lower-triangular matrix of size $p \times p$
3. μ_FC_arr : VB Posterior Mean for each iteration. A matrix of size $p \times (SDG_iters + 1)$
4. C_FC_arr : VB Posterior Variance-Covariance (Cholesky) for each iteration. An array of size $p \times p \times (SDG_iters + 1)$

Author(s)

Weichang 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

```

# -----
# Initialise Inputs
# -----
# Generating 30 data points from a simple linear-regression model
set.seed(1)
x  <- runif(30, min = -5, max = 5)
vari <- rnorm(30, mean = 0, sd = 1)
y  <- 0.75 - x + vari
lam0 <- matrix(c(0,0), nrow = 2)
z  <- cbind(x, y)

# Specify moment condition functions for linear regression and its corresponding derivative
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)
}

# Specify derivative of log prior
delth_logpi <- function(theta) { -0.0001 * mu0 }

# Specify AEL constant and Newton-Rhapson iteration
a          <- 0.00001
AEL_iters <- 10

# Specify initial values for GVA mean vector and Cholesky
reslm <- lm(y ~ x)
mu0  <- matrix(unnamed(reslm$coefficients),2,1)
C0   <- unname(t(chol(vcov(reslm))))

# Specify details for ADADELTA (Stochastic Gradient-Descent)
SDG_iters <- 50
epsil    <- 10^-5
rho      <- 0.9

# -----
# Main
# -----
result <- compute_GVA(mu0, C0, h, delthh, delth_logpi, z, lam0,
rho, epsil, a, SDG_iters, AEL_iters)

```

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

```
# -----
# Initialise Inputs
# -----
# Generating 30 data points from a simple linear-regression model
seedNum <- 100
set.seed(seedNum)
n       <- 100
p       <- 10
lam0    <- matrix(0, nrow = p)
mean    <- rep(1, p)
sigStar <- matrix(0.4, p, p) + diag(0.6, p)
z       <- rmvnorm::rmvnorm(n = n-1, mean = mean, sigma = sigStar)

# Specify moment condition functions for linear regression and its corresponding derivative
h       <- function(zi, th) { matrix(zi - th, nrow = 10) }
delthh  <- function(z, th) { -diag(p) }

# Specify derivative of log prior
delth_logpi <- function(theta) {-0.0001 * theta}

# Specify AEL constant and Newton-Rhapson iteration
AEL_iters <- 5 # Number of iterations for AEL
a         <- 0.00001

# Specify initial values for GVA mean vector and Cholesky
zbar     <- 1/(n-1) * matrix(colSums(z), nrow = p)
mu_0     <- matrix(zbar, p, 1)

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
C_0     <- 1/sqrt(n) * t(chol(sigHat))

# Specify details for ADADELTA (Stochastic Gradient-Descent)
SDG_iters <- 5 # Number of iterations for GVA
epsil     <- 10^-5
rho       <- 0.9

# -----
# Main
# -----
# Compute GVA
ansGVA <- compute_GVA(mu_0, C_0, h, delthh, delth_logpi, z, lam0, rho, epsil,
a, SDG_iters, AEL_iters)

# Plot graphs
diagnostic_plot(ansGVA) # Plot all graphs
diagnostic_plot(ansGVA, muList = c(1,4)) # Limit number of graphs
diagnostic_plot(ansGVA, cList = matrix(c(1,1, 5,6, 3,3), ncol = 2)) # Limit number of graphs
```

Index

* package

VBel-package, [2](#)

compute_AEL, [3](#)

compute_AEL(), [3](#)

compute_GVA, [5](#), [8](#)

compute_GVA(), [3](#)

diagnostic_plot, [7](#)

diagnostic_plot(), [3](#)

VBel (VBel-package), [2](#)

VBel-package, [2](#)