Package 'hgm'

April 7, 2022

Type Package

Depends R (>= 2.6.0), deSolve

Title Holonomic Gradient Method and Gradient Descent

Version 1.20

Date 2022-04-07

Author Nobuki Takayama, Tamio Koyama, Tomonari Sei, Hiromasa Nakayama, Kenta Nishiyama

Maintainer Nobuki Takayama <takayama@math.kobe-u.ac.jp>

Description The holonomic gradient method (HGM, hgm) gives a way to evaluate normalization constants of unnormalized probability distributions by utilizing holonomic systems of differential or difference equations. The holonomic gradient descent (HGD, hgd) gives a method to find maximal likelihood estimates by utilizing the HGM.

License GPL-2

LazyLoad yes

URL http://www.openxm.org

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-04-07 07:42:33 UTC

R topics documented:

hgm-package *HGM*

Description

The holonomic gradient method (HGM, hgm) gives a way to evaluate normalizing constants of unnormalized probability distributions by utilizing holonomic systems of differential or difference equations. The holonomic gradient descent (HGD, hgd) gives a method to find maximal likelihood estimates by utilizing the HGM.

Details

Package:	hgm
Type:	Package
License:	GPL-2
LazyLoad:	yes

The HGM and HGD are proposed in the paper below. This method based on the fact that a broad class of normalizing constants of unnormalized probability distributions belongs to the class of holonomic functions, which are solutions of holonomic systems of linear partial differential equations.

Note

This package includes a small subset of the Gnu scientific library codes (http://www.gnu.org/software/gsl/). Then, it might cause a conflict with the package gsl.

References

- (N3OST2) Hiromasa Nakayama, Kenta Nishiyama, Masayuki Noro, Katsuyoshi Ohara, Tomonari Sei, Nobuki Takayama, Akimichi Takemura, Holonomic Gradient Descent and its Application to Fisher-Bingham Integral, Advances in Applied Mathematics 47 (2011), 639–658, doi: 10.1016/j.aam.2011.03.001
- (dojo) Edited by T.Hibi, Groebner Bases: Statistics and Software Systems, Springer, 2013, doi: 10.1007/9784431545743
- http://www.openxm.org

See Also

hgm.ncBingham, hgm.ncorthant, hgm.ncso3, hgm.pwishart, hgm.Rhgm hgm.p2wishart,

Examples

```
## Not run:
example(hgm.ncBingham)
```

```
example(hgm.ncorthant)
example(hgm.ncso3)
example(hgm.pwishart)
example(hgm.Rhgm)
example(hgm.p2wishart)

## End(Not run)
```

`hgm.ncBingham` *The function hgm.ncBingham performs the holonomic gradient method (HGM) for Bingham distributions.*

Description

The function hgm.ncBingham performs the holonomic gradient method (HGM) for Bingham distributions with the deSolve package in R.

Usage

```
hgm.ncBingham(th, d=rep(1,length(th)+1), logarithm=FALSE,
              ini.method="power", times=NULL, withvol=FALSE, ...)
```

Arguments

`th`	A (p-1)-dimensional vector which specifies the first (p-1) components of the parameter vector of the Bingham distribution on the (p-1)-dim sphere. The p-th parameter is assumed to be zero.
`d`	A p-dimensional vector which specifies the multiplicity of the parameter. The default is all-one vector.
`logarithm`	If 'logarithm' is TRUE, then the result is log of the normalizing constant.
`ini.method`	The method for computing the initial value. Only "power" is implemented now.
`times`	a vector; times in [0,1] at which explicit estimates for G are desired. If time = NULL, the set 0,1 is used, and only the final value is returned.
`withvol`	If 'withvol' is TRUE, then the normalizing constant with volume of sphere is returned. Otherwise that without volume is returned. Therefore, if 'withvol' is FALSE and the parameter is zero, then the normalizing constant becomes 1.
`...`	Additional parameters for computing initial values. Details are omitted.

Details

The function hgm.ncBingham computes the normalizing constant of the Bingham distribution and its derivatives at any specified point. The initial value is computed by the power series expansion.

Value

The output is p-dimensional vector G. The first element of G is the normalizing constant and the following (p-1)-elements are partial derivative of the normalizing constant with respect to the first (p-1) components of the parameter 'th'.

Author(s)

Tomonari Sei

References

http://www.math.kobe-u.ac.jp/OpenXM/Math/hgm/ref-hgm.html

Examples

```
# Example 1.
hgm.ncBingham(c(1,3,5))
```

`hgm.ncorthant` *The function hgm.ncorthant evaluates the orthant probability.*

Description

The function hgm.ncorthant evaluates the orthant probability, which is the normalization constant of the multivariate normal distribution restrcted to the first orthant.

Usage

```
hgm.ncorthant(x,y,rk_step_size=1e-3)
```

Arguments

`x`	See the description of y.
`y`	This function evaluates the orthant probability for the m dimensional multivariate normal distribution whose m by m covariance matrix and the mean vector of size m are x and y respectively.
`rk_step_size`	The step size for the Runge-Kutta method to apply the HGM.

Details

The function hgm.ncorthant evaluates the orthant probability, which is the normalization constant of the m-dimensional multivariate normal distribution restrcted to the first orthant. It uses the holonomic gradient method (HGM) to evalute it. The rank of the system of differential equations for the HGM is 2^m.

Value

The output is the orthant probalibity.

Author(s)

Tamio Koyama

References

Tamio Koyama, Akimichi Takemura, Calculation of orthant probabilities by the holonomic gradient method, https://arxiv.org/abs/1211.6822.

Examples

```
## ====================================================
## Example 1. Computing the orthant probability
## ====================================================
x <- matrix(c(15,26,23,19,
      26,47,46,35,
      23,46,65,38,
      19,35,38,33), nrow =4)
y <- c(1,2,3,4)
hgm.ncorthant(x,y)
```

`hgm.ncso3` *The function hgm.ncso3 evaluates the normalization constant for the Fisher distribution on SO(3).*

Description

The function hgm.ncso3 evaluates the normalization constant for the Fisher distribution on SO(3).

Usage

```
hgm.ncso3(a,b,c,t0=0.0,q=1,deg=0,log=0)
```

Arguments

a	See the description of c.
b	See the description of c.
c	This function evaluates the normalization constant for the parameter Theta=diag(theta_ii) of the Fisher distribution on SO(3). The variables a,b,c stand for the parameters theta_11, theta_22, theta_33 respectively.
t0	It is the initial point to evaluate the series. If it is set to 0.0, a default value is used.
q	If it is 1, then the program works in a quiet mode.
deg	It gives the approximation degree of the power series approximation of the normalization constant near the origin. If it is 0, a default value is used.
log	If it is 1, then the function returns the log of the normalizing constant.

Details

The normalization constant c(Theta) of the Fisher distribution on SO(3) is defined by integral(exp(trace(transpose(Theta) X))) where X is the integration variable and runs over S0(3) and Theta is a 3 x 3 matrix parameter. A general HGM algorithm to evaluate the normalization constant is given in the reference below. We use the Corollary 1 and the series expansion in 3.2 for the evaluation.

Value

The output is an array of c(Theta) and its derivatives with respect to Theta_11,Theta_22,Theta_33. It is the vector C of the reference below. When log=1, the output is an array of log of them.

Author(s)

Nobuki Takayama

References

Tomonari Sei, Hiroki Shibata, Akimichi Takemura, Katsuyoshi Ohara, Nobuki Takayama, Properties and applications of Fisher distribution on the rotation group, Journal of Multivariate Analysis, 116 (2013), 440–455, doi: 10.1016/j.jmva.2013.01.010

Examples

```
## =====================================================
## Example 1. Computing normalization constant of the Fisher distribution on SO(3)
## =====================================================
hgm.ncso3(1,2,3)[1]

## =====================================================
## Example 2. Asteroid data in the paper
## =====================================================
hgm.ncso3(19.6,0.831,-0.671)[1]
```

`hgm.p2wishart` *The function hgm.p2wishart evaluates the cumulative distribution function of the largest eigenvalues of W1*inverse(W2).*

Description

The function hgm.p2wishart evaluates the cumulative distribution function of the largest eigenvalues of W1*inverse(W2) where W1 and W2 are Wishart matrices of size m*m of the freedom n1 and n2 respectively.

Usage

```
hgm.p2wishart(m,n1,n2,beta,q0,approxdeg,h,dp,q,mode,method,
            err,automatic,assigned_series_error,verbose,autoplot)
```

Arguments

m	The dimension of the Wishart matrix.
n1	The degree of freedome of the Wishart distribution S1
n2	The degree of freedome of the Wishart distribution S2
beta	The eigenvalues of inverse(S2)*S1 where S1 and S2 are covariant matrices of W1 and W2 respectively.
q0	The point to evaluate the matrix hypergeometric series. q0>0
approxdeg	Zonal polynomials upto the approxdeg are calculated to evaluate values near the origin. A zonal polynomial is determined by a given partition (k1,...,km). We call the sum k1+...+km the degree.
h	A (small) step size for the Runge-Kutta method. h>0.
dp	Sampling interval of solutions by the Runge-Kutta method. When autoplot=1 or dp is negative, it is automatically set. if it is 0, no sample is stored.
q	The second value y[0] of this function is the Prob(L1 < q) where L1 is the first eigenvalue of the Wishart matrix.
mode	When mode=c(1,0,0), it returns the evaluation of the matrix hypergeometric series and its derivatives at q0. When mode=c(1,1,(2^m+1)*p), intermediate values of P(L1 < x) with respect to p-steps of x are also returned. Sampling interval is controled by dp. When autoplot=1, mode is automatically set.
method	a-rk4 is the default value. When method="a-rk4", the adaptive Runge-Kutta method is used. Steps are automatically adjusted by err.
err	When err=c(e1,e2), e1 is the absolute error and e2 is the relative error. This parameter controls the adative Runge-Kutta method. If the output is absurd, you may get a correct answer by setting, e.g., err=c(1e-(xy+5), 1e-10) or by increasing q0 when initial value at q0 is very small as 1e-xy.
automatic	automatic=1 is the default value. If it is 1, the degree of the series approximation will be increased until |(F(i)-F(i-1))/F(i-1)| < assigned_series_error where F(i) is the degree i approximation of the hypergeometric series with matrix argument. Step sizes for the Runge-Kutta method are also set automatically from the assigned_series_error if it is 1.
assigned_series_error	assigned_series_error=0.00001 is the default value.
verbose	verbose=0 is the default value. If it is 1, then steps of automatic degree updates and several parameters are output to stdout and stderr.
autoplot	autoplot=0 is the default value. If it is 1, then this function outputs an input for plot (which is equivalent to setting the 3rd argument of the mode parameter properly). When ans is the output, ans[1,] is c(q,prob at q,...), ans[2,] is c(q0,prob at q0,...), and ans[3,] is c(q0+q/100,prob at q/100,...), ... When the adaptive Runge-Kutta method is used, the step size h may change automatically, which makes the sampling period change, in other words, the sampling points q0+q/100, q0+2*q/100, q0+3*q/100, ... may change. In this case, the output matrix may contain zero rows in the tail or overfull. In case of the overful, use the mode option to get the all result.

Details

It is evaluated by the Koev-Edelman algorithm when x is near the origin and by the HGM when x is far from the origin. We can obtain more accurate result when the variables h is smaller, q0 is relevant value (not very big, not very small), and the approxdeg is more larger. A heuristic method to set parameters q0, h, approxdeg properly is to make x larger and to check if the y[0] approaches to 1.

Value

The output is x, y[0], ..., y[2^m] in the default mode, y[0] is the value of the cumulative distribution function P(L1 < x) at x. y[1],...,y[2^m] are some derivatives. See the reference below.

Note

This function does not work well under the following cases: 1. The beta (the set of eigenvalues) is degenerated or is almost degenerated. 2. The beta is very skew, in other words, there is a big eigenvalue and there is also a small eigenvalue. The error control is done by a heuristic method. The obtained value is not validated automatically.

Author(s)

Nobuki Takayama

References

H.Hashiguchi, N.Takayama, A.Takemura, Distribution of ratio of two Wishart matrices and evaluation of cumulative probability by holonomic gradient method.

Examples

```
## =====================================================
## Example 1.
## =====================================================
hgm.p2wishart(m=3,n1=5,n2=10,beta=c(1,2,4),q=4)
## =====================================================
## Example 2.
## =====================================================
b<-hgm.p2wishart(m=3,n1=5,n2=10,beta=c(1,2,4),q0=0.3,q=20,approxdeg=20,mode=c(1,1,(8+1)*1000));
c<-matrix(b,ncol=8+1,byrow=1);
#plot(c)
## =====================================================
## Example 3.
## =====================================================
c<-hgm.p2wishart(m=3,n1=5,n2=10,beta=c(1,2,4),q0=0.3,q=20,approxdeg=20,autoplot=1);
#plot(c)
```

hgm.pwishart *The function hgm.pwishart evaluates the cumulative distribution function of random wishart matrices.*

Description

The function hgm.pwishart evaluates the cumulative distribution function of random wishart matrices of size m times m.

Usage

```
hgm.pwishart(m,n,beta,q0,approxdeg,h,dp,q,mode,method,
            err,automatic,assigned_series_error,verbose,autoplot)
```

Arguments

m	The dimension of the Wishart matrix.
n	The degree of freedome (a parameter of the Wishart distribution)
beta	The eigenvalues of the inverse of the covariant matrix /2 (a parameter of the Wishart distribution). The beta is equal to inverse(sigma)/2.
q0	The point to evaluate the matrix hypergeometric series. q0>0
approxdeg	Zonal polynomials upto the approxdeg are calculated to evaluate values near the origin. A zonal polynomial is determined by a given partition (k1,...,km). We call the sum k1+...+km the degree.
h	A (small) step size for the Runge-Kutta method. h>0.
dp	Sampling interval of solutions by the Runge-Kutta method. When autoplot=1 or dp is negative, it is automatically set. if it is 0, no sample is stored.
q	The second value y[0] of this function is the Prob(L1 < q) where L1 is the first eigenvalue of the Wishart matrix.
mode	When mode=c(1,0,0), it returns the evaluation of the matrix hypergeometric series and its derivatives at q0. When mode=c(1,1,(2^m+1)*p), intermediate values of P(L1 < x) with respect to p-steps of x are also returned. Sampling interval is controled by dp. When autoplot=1, it is automatically set.
method	a-rk4 is the default value. When method="a-rk4", the adaptive Runge-Kutta method is used. Steps are automatically adjusted by err.
err	When err=c(e1,e2), e1 is the absolute error and e2 is the relative error. This parameter controls the adative Runge-Kutta method. If the output is absurd, you may get a correct answer by setting, e.g., err=c(1e-(xy+5), 1e-10) or by increasing q0 when initial value at q0 is very small as 1e-xy.
automatic	automatic=1 is the default value. If it is 1, the degree of the series approximation will be increased until |(F(i)-F(i-1))/F(i-1)| < assigned_series_error where F(i) is the degree i approximation of the hypergeometric series with matrix argument. Step sizes for the Runge-Kutta method are also set automatically from the assigned_series_error if it is 1.

`assigned_series_error`
: assigned_series_error=0.00001 is the default value.

`verbose`
: verbose=0 is the default value. If it is 1, then steps of automatic degree updates and several parameters are output to stdout and stderr.

`autoplot`
: autoplot=0 is the default value. If it is 1, then this function outputs an input for plot (which is equivalent to setting the 3rd argument of the mode parameter properly). When ans is the output, ans[1,] is c(q,prob at q,...), ans[2,] is c(q0,prob at q0,...), and ans[3,] is c(q0+q/100,prob at q/100,...), ... When the adaptive Runge-Kutta method is used, the step size h may change automatically, which makes the sampling period change, in other words, the sampling points q0+q/100, q0+2*q/100, q0+3*q/100, ... may change. In this case, the output matrix may contain zero rows in the tail or overfull. In case of the overful, use the mode option to get the all result.

Details

It is evaluated by the Koev-Edelman algorithm when x is near the origin and by the HGM when x is far from the origin. We can obtain more accurate result when the variables h is smaller, q0 is relevant value (not very big, not very small), and the approxdeg is more larger. A heuristic method to set parameters q0, h, approxdeg properly is to make x larger and to check if the y[0] approaches to 1.

Value

The output is x, y[0], ..., y[2^m] in the default mode, y[0] is the value of the cumulative distribution function P(L1 < x) at x. y[1],...,y[2^m] are some derivatives. See the reference below.

Note

This function does not work well under the following cases: 1. The beta (the set of eigenvalues) is degenerated or is almost degenerated. 2. The beta is very skew, in other words, there is a big eigenvalue and there is also a small eigenvalue. The error control is done by a heuristic method. The obtained value is not validated automatically.

Author(s)

Nobuki Takayama

References

H.Hashiguchi, Y.Numata, N.Takayama, A.Takemura, Holonomic gradient method for the distribution function of the largest root of a Wishart matrix, Journal of Multivariate Analysis, 117, (2013) 296-312, doi: 10.1016/j.jmva.2013.03.011,

Examples

```
## =====================================================
## Example 1.
## =====================================================
hgm.pwishart(m=3,n=5,beta=c(1,2,3),q=10)
```

```
## ====================================================
## Example 2.
## ====================================================
b<-hgm.pwishart(m=4,n=10,beta=c(1,2,3,4),q0=1,q=10,approxdeg=20,mode=c(1,1,(16+1)*100));
c<-matrix(b,ncol=16+1,byrow=1);
#plot(c)
## ====================================================
## Example 3.
## ====================================================
c<-hgm.pwishart(m=4,n=10,beta=c(1,2,3,4),q0=1,q=10,approxdeg=20,autoplot=1);
#plot(c)
```

`hgm.Rhgm` *The function hgm.Rhgm performs the holonomic gradient method (HGM) for a given Pfaffian system and an initial value vector.*

Description

The function hgm.Rhgm performs the holonomic gradient method (HGM) for a given Pfaffian system and an initial value vector with the deSolve package in R.

Usage

```
hgm.Rhgm(th0, G0, th1, dG.fun, times=NULL, fn.params=NULL)
```

Arguments

`th0`	A d-dimensional vector which is an initial point of the parameter vector th (theta).
`G0`	A r-dimensional vector which is the initial value of the vector G of the normalizing constant and its derivatives.
`th1`	A d-dimensional vector which is the target point of th.
`dG.fun`	dG.fun is the "right hand sides" of the Pfaffian system. It is a d*r-dimensional array.
`times`	a vector; times in [0,1] at which explicit estimates for G are desired. If time = NULL, the set 0,1 is used, and only the final value is returned.
`fn.params`	fn.params: a list of parameters passed to the function dG.fun. If fn.params = NULL, no parameter is passed to dG.fun.

Details

The function hgm.Rhgm computes the value of a holonomic function at a given point, using HGM. This is a "Step 3" function (see the reference below), which can be used for an arbitrary input, in the HGM framework. Efficient "Step 3" functions are given for some distributions in this package.

The Pfaffian system assumed is d G_j / d th_i = (dG.fun(th, G))_i,j

The inputs of hgm.Rhgm are the initial point th0, initial value G0, final point th1, and Pfaffian system dG.fun. The output is the final value G1.

If the argument 'times' is specified, the function returns a matrix, where the first column denotes time, the following d-vector denotes th, and the remaining r-vector denotes G.

Value

The output is the value of G at th1. The first element of G is the normalizing constant.

Author(s)

Tomonari Sei

References

http://www.math.kobe-u.ac.jp/OpenXM/Math/hgm/ref-hgm.html

Examples

```
# Example 1.
# A demo program; von Mises--Fisher on S^{3-1}

G.exact = function(th){  # exact value by built-in function
  c( sinh(th[1])/th[1], cosh(th[1])/th[1] - sinh(th[1])/th[1]^2 )
}

dG.fun = function(th, G, fn.params=NULL){  # Pfaffian
  dG = array(0, c(1, 2))
  sh = G[1] * th[1]
  ch = G[2] * th[1] + G[1]
  dG[1,1] = G[2] # Pfaffian eq's
  dG[1,2] = sh/th[1] - 2*ch/th[1]^2 + 2*sh/th[1]^3
  dG
}

th0 = 0.5
th1 = 15

G0 = G.exact(th0)
G0

G1 = hgm.Rhgm(th0, G0, th1, dG.fun)  # HGM
G1

G1.exact = G.exact(th1)
G1.exact

#
# Example 2.
#
hgm.Rhgm.demo1()
```

`hgm.Rhgm.demo1` *The function hgm.Rhgm.demo1 performs a demonstration of the function hgm.Rhgm.*

Description

The function hgm.Rhgm.demo1 performs a demonstration of the function hgm.Rhgm.

Usage

```
hgm.Rhgm.demo1()
```

Details

The function hgm.Rhgm.demo1 evaluates the normalizing constant of the Von-Mises distribution by the HGM.

Value

The returned value is a dataframe. The column exact of the dataframe is the exact value of the normalizing constant. The column byHGM is the value obtained by the HGM. The column start is the initial value for the HGM.

Author(s)

Tomonari Sei

See Also

`hgm.Rhgm`

Index