Package 'sparseHessianFD'

September 24, 2021

Type Package

Title Numerical Estimation of Sparse Hessians

Version 0.3.3.5

Date 2021-09-22

Maintainer Michael Braun <braunm@smu.edu>

URL https://braunm.github.io/sparseHessianFD/,
https://github.com/braunm/sparseHessianFD/

BugReports https://github.com/braunm/trustOptim/issues/

Description Estimates Hessian of a scalar-valued function, and returns it in a sparse Matrix format. The sparsity pattern must be known in advance. The algorithm is especially efficient for hierarchical models with a large number of heterogeneous units. See Braun, M. (2017) <doi:10.18637/jss.v082.i10>.

License MPL (== 2.0)

LazyData true

Depends R (>= 3.4.0)

Imports Matrix (>= 1.3), methods, Rcpp (>= 0.12.13)

Suggests testthat, numDeriv, scales, knitr, xtable, dplyr

LinkingTo Rcpp, RcppEigen (>= 0.3.3.3.0)

Encoding UTF-8

VignetteBuilder knitr

SystemRequirements C++11

RoxygenNote 7.1.2

NeedsCompilation yes

Author Michael Braun [aut, cre, cph] (<https://orcid.org/0000-0003-4774-2119>)

Repository CRAN

Date/Publication 2021-09-24 04:30:04 UTC

R topics documented:

`sparseHessianFD-package`

Estimate sparse Hessians using finite differences of gradients.

Description

Estimate sparse Hessians using finite differences of gradients.

Details

The Hessian is returned as a sparse Matrix (dgCMatrix-class). The user supplies the objective function, the gradient, and the row and column indices of the non-zero elements of the lower triangle of the Hessian (i.e., the sparsity structure must be known in advance).

In a typical case, you should only need to use the sparseHessianFD initializer, and the fn, gr and hessian methods of the sparseHessian class, and the Matrix.to.Coord utility function.

References

Braun, Michael. 2017. sparseHessianFD: An R Package for Estimating Sparse Hessian Matrices. Journal of Statistical Software 82 (10): 1-22.

Coleman, Thomas F, and Jin-Yi Cai. 1986. The Cyclic Coloring Problem and Estimation of Sparse Hessian Matrices. SIAM Journal on Algebraic Discrete Methods 7 (2): 221-235

Coleman, Thomas F, Burton S Garbow, and Jorge J More. 1985. Software for Estimating Sparse Hessian Matrices. ACM Transaction on Mathematical Software 11 (4) (December): 363-377.

Coleman, Thomas F and Jorge J More. 1984. Estimation of Sparse Hessian Matrices and Graph Coloring Problems. Mathematical Programming 28 (3) (October): 243-270

Powell, M J D and Ph L Toint. 1979. On the Estimation of Sparse Hessian Matrices. SIAM Journal on Numerical Analysis 16 (6) (December): 1060-1074.

binary *Binary choice example*

Description

Functions for binary choice example in the vignette.

Usage

```
binary.f(P, data, priors, order.row = FALSE)

binary.grad(P, data, priors, order.row = FALSE)

binary.hess(P, data, priors, order.row = FALSE)
```

Arguments

P	Numeric vector of length $(N+1)k$. First Nk elements are heterogeneous coefficients. The remaining k elements are population parameters.
data	Named list of data matrices Y and X, and choice count integer T
priors	Named list of matrices inv.Omega and inv.Sigma
order.row	Determines order of heterogeneous coefficients in parameter vector. If TRUE, heterogeneous coefficients are ordered by unit. If FALSE, they are ordered by covariate.

Details

These functions are used by the heterogeneous binary choice example in the vignette. There are N heterogeneous units, each making T binary choices. The choice probabilities depend on k covariates.

Value

Log posterior density, gradient and Hessian. The Hessian is a dgCMatrix object.

binary-data *Sample simulated data for binary choice example in vignette*

Description

Sample datasets for vignette

Details

The package provides four sample datasets for the hierarchical binary choice model described in the vignette. These datasets are:

binary $N = 50, k = 3$

binary_small $N = 20, k = 2$

binary_large $N = 800, k = 3$

binary_super $N = 1200, k = 3$

N is the number of heterogeneous units. k is the number of covariates.

The datasets were generated using the code in data-raw/binary_sim.R.

`coloring` *Triangular partitioning of variables*

Description

cyclic coloring from a lower triangular pattern matrix

Usage

```
coloring(L)
```

Arguments

`L`	sparsity pattern of the Hessian as a lower triangular pattern matrix

Details

For internal use. Exported in order for replication files for JSS article to run.

Value

Integer vector of length nvars with color assignments for each variable.

`Coord.to.Pointers` *Convert a matrix defined by row and column indices to one defined by a row- or column-oriented compression scheme.*

Description

Returns indices and pointers that define a sparse Hessian in compressed format. Inputs are the row and column indices.

Usage

```
Coord.to.Pointers(
  rows,
  cols,
  dims,
  triangle = TRUE,
  lower = TRUE,
  symmetric = FALSE,
  order = c("column", "row", "triplet"),
  index1 = TRUE
)
```

Arguments

`rows, cols`	row and column indices of non-zero elements
`dims`	2-element vector for number of rows and columns.
`triangle`	Is input intended to be a triangular (TRUE) or full (FALSE) matrix. See details for how this argument is interpreted for different values of `order`.
`lower`	If `triangular` is true, this argument identifies the input matrix as lower- or upper-triangular. This argument is ignored if `triangle` is FALSE.
`symmetric`	If TRUE, and matrix is triangular, then the matrix will be treated as symmetric, with only the triangular elements provided. If matrix is neither triangular nor symmetric, then symmetric=TRUE will probably trigger an error.
`order`	Determines the indexing/compression scheme for the output matrix. Use "triplet" to get row and column indices. Defaults to the same class as M.
`index1`	TRUE if using 1-based indexing. FALSE for 0-based indexing.

Details

`triangle` and `order` have the following interpretation:

triangle=TRUE Input `rows` and cols represent lower or upper triangle of a matrix. If `order="symmetric"`, then the output list will be for a full, symmetric matrix. Otherwise, the output list will be for only the lower or upper triangle. Any elements outside of the specified triangle will trigger an error.

triangle=FALSE Input `rows` and cols represent a full matrix. If that matrix is not symmetric, then `order=="symmetric"` will trigger an error. If `symmetric=FALSE` and `order='triplet'`, the output list should contain the same row and column indices as the input list.

Value

A list. See Matrix.to.Pointers (no values are included in return list).

See Also

Matrix.to.Pointers

`deprecated` *Deprecated functions*

Description

These functions were in earlier versions, but will no longer be maintained, and are not even guaranteed to work now.

Build sparse matrix from data in CSC (column compressed) format.

Converts row and column indices to a pattern Matrix object of Matrix class

This function is deprecated. Use `sparseHessianFD` instead.

Usage

```
Sym.CSC.to.Matrix(H, nvars)

Coord.to.Sym.Pattern.Matrix(H, nvars)

Coord.to.Pattern.Matrix(
  rows,
  cols,
  dims,
  compressed = TRUE,
  symmetric = FALSE,
  index1 = TRUE
)

new.sparse.hessian.obj(
  x,
  fn,
  gr,
  hs,
  fd.method = 0L,
  eps = sqrt(.Machine$double.eps),
  ...
```

```
)

sparseHessianFD.new(
  x,
  fn,
  gr,
  rows,
  cols,
  direct = NULL,
  eps = sqrt(.Machine$double.eps),
  ...
)
```

Arguments

H	a list containing Hessian data. See details.
nvars	the number of rows (and columns) in the matrix.
rows, cols	row and column indices of non-zero elements
dims	2-element vector for number of rows and columns in matrix
compressed	If TRUE, returns a matrix is compressed column (default=TRUE)
symmetric	If TRUE, matrix will be symmetric, and only the lower triangular elements need to be provided (default=FALSE)
index1	TRUE if input row and col use 1-based indexing, and FALSE for 0-based indexing.
x	variable vector for initialization
fn	R function that returns function value
gr	R function that returns the gradient of the function
hs	list of two vectors: row and column indices of non-zero elements of lower triangle of Hessian. See details.
fd.method	If TRUE, use direct method for computatation. Otherwise, use indirect/substitution method. See references.
eps	The perturbation amount for finite differencing of the gradient to compute the Hessian. Defaults to sqrt(.Machine$double.eps).
...	Other parameters to be passed to fn and gr.
direct	If TRUE, use direct method for computatation. Otherwise, use indirect/substitution method. See references.

Details

Use Matrix::sparseMatrix instead of Sym.CSC.to.Matrix.

Use Coord.to.Pattern.Matrix with symmetric=TRUE instead of Coord.to.Sym.Pattern.Matrix.

This function is useful to prototype a sparsity pattern. No assumptions are made about symmetry.

hs is a list of two elements:

iRow Integer vector of row indices of non-zero elements in lower triangle of Hessian.

jCol Integer vector of column indices of non-zero elements in lower triangle of Hessian.

This function is deprecated. Use `sparseHessianFD` instead.

Value

An object of Matrix class.

A sparse pattern matrix

An object of class sparseHessianFD

`get_colors` *Vertex coloring of a sparse undirected graph*

Description

Generate proper vertex coloring of a sparse undirected graph.

Usage

```
get_colors(pntr, idx, nvars)
```

Arguments

`pntr, idx`	row pointers and column indices of the adjacency matrix, in compressed column-oriented format. Must use zero-based indexing.
`nvars`	Number of vertices.

Details

For internal use. You should not have to call this function directly.

Value

An integer vector of length nvars, where each element represents the color of the corresponding vertex. Indices are zero-based.

`Matrix.to.Coord` *Row and column indices from sparse matrix.*

Description

Utility function to extract row and column indices of the non-zero elements of a sparse matrix.

Usage

```
Matrix.to.Coord(M, index1 = TRUE)
```

Arguments

`M`	A matrix that is a subclass of sparseMatrix, as defined in the Matrix package.
`index1`	TRUE if the index of the first element should be 1, and FALSE if 0.

Details

A wrapper to Matrix.to.Pointers for `order='triplet'` and `values=FALSE`, for extracting the row and column indices of a sparsity pattern from a matrix that has that same pattern.

Value

A list with two named elements.

rows Integer vector containing row indices of non-zero elements

cols Integer vector containing column indices of non-zero elements

Examples

```
M1 <- as(kronecker(diag(3), matrix(TRUE,2,2)),"sparseMatrix")
C <- Matrix.to.Coord(M1)
M2 <- Matrix::sparseMatrix(i=C$rows, j=C$cols)
all.equal(M1,M2)
```

`Matrix.to.Pointers` *Extract row and column indices, pointers and values from a sparse matrix.*

Description

Returns a list of row indices, column indices, pointers, and/or values of a sparse Hessian.

Usage

```
Matrix.to.Pointers(
  M,
  as.symmetric = Matrix::isSymmetric(M),
  values = !is(M, "nMatrix"),
  order = NULL,
  index1 = TRUE
)
```

Arguments

M	A sparse Matrix, as defined in the Matrix package.
as.symmetric	Defaults to isSymmetric(M). If M is symmetric, and as.symmetric is FALSE, then index/pointer elements in the output list will be labeled according to order. If M is not symmetric, and as.symmetric is TRUE, then an error will be triggered.
values	If TRUE, values are returned in list as 'x'. Defaults to TRUE for numeric and logical matrices, and FALSE for pattern matrices. If M is a pattern matrix, values=TRUE will trigger a warning.
order	Determines the indexing/compression scheme for the output matrix. Use 'triplet" to get row and column indices. Defaults to the same class as M.
index1	TRUE (default) if return indices and pointers should use 1-based indexing. FALSE for 0-based indexing.

Details

This function is included primarily for debugging purposes. It is used internally, but would not ordinarily be called by an end user.

Value

A list with the following elements. If order=='row',

jCol Integer vector containing column indices of non-zero elements

ipntr Integer vector containing pointers to elements of jCol at which the next row begins.

If order=='column'

iRow Integer vector containing row indices of non-zero elements

jpntr Integer vector containing pointers to elements of iRow at which the next column begins.

If order=='triplet'

rows Row indices of non-zero elements

cols Column indices of non-zero elements

If as.symmetric is TRUE, then the row/column orientation does not matter.

idx Integer vector containing indices of non-zero elements

pntr Integer vector containing pointers to elements of idx at which the next row or column begins.

If values=TRUE, the return list includes x, the values of the non-zero elements. The 'class' element is the name of the sparse matrix class to which the output corresponds (identifies numeric type, pattern, and indexing/compression scheme).

See Also

Matrix.to.Coord

`sparseHessianFD` *sparseHessianFD*

Description

A reference class for computing sparse Hessians

Details

The sparseHessianFD function calls the initializer for the sparseHessianFD class, and returns a sparseHessianFD object.

```
sparseHessianFD(x, fn, gr, rows, cols, delta, index1, complex, ...)
```

The function, gradient and sparsity pattern are declared as part of the initialization.

Once initialized, the $hessian method will evaluate the Hessian at x.

```
obj <- sparseHessian(x, fn, gr, rows, cols, ...)
obj$hessian(x)
```

For convenience, the class provides wrapper methods to the `fn` and `gr` functions that were specified in the initializer.

```
obj$fn(P) ## wrapper to objective function
obj$gr(P) ## wrapper to gradient
obj$fngr(P) ## list of obj function and gradient
obj$fngrhs(P) ## list of obj function, gradient and Hessian.
```

Arguments to initializer:

x an vector at which the function, gradient and Hessian are initialized and tested.

fn, gr R functions that return the function value and gradient, evaluated at x.

rows, cols Numeric vectors with row and column indices of the non-zero elements in the lower triangle (including diagonal) of the Hessian.

delta The perturbation amount for finite difference (or complex step) of the gradient to compute the Hessian. Defaults to 1e-07.

index1 TRUE if rows and cols use 1-based (R format) indexing (FALSE for 0-based (C format) indexing.

complex TRUE if Hessian will be computed using the complex step method, and FALSE (default) if using finite differences. If TRUE, both fn and gr must accept complex arguments and return complex values.

... other arguments to be passed to fn and gr.

Other methods are described below. Do not access any of the fields directly. The internal structure is subject to change in future versions.

Fields

`fn1` A closure for calling fn(x, ...).

`gr1` A closure for calling gr(x, ...).

`iRow,jCol` Numeric vectors with row and column indices of the non-zero elements in the lower triangle (including diagonal) of the Hessian.

`delta` The perturbation amount for finite differencing of the gradient to compute the Hessian. Defaults to 1e-07.

`index1` TRUE if rows and cols use 1-based (R format) indexing (FALSE for 0-based (C format) indexing.

`complex` TRUE if Hessian will be computed using the complex step method, and FALSE (default) if using finite differences.

`D` raw finite differences (internal use only)

`nvars` Number of variables (length of x)

`nnz` Number of non-zero elements in the lower triangle of the Hessian.

`ready` TRUE if object has been initialized, and Hessian has been partitioned.

`idx,pntr` Column indices and row pointers for non-zero elements in lower triangle of the permuted Hessian. Row-oriented compressed storage.

`colors` A vector representation of the partitioning of the columns. There are nvars elements, one for each column of the permuted Hessian. The value corresponds to the "color" for that column.

`perm,invperm` Permutation vector and its inverse

Methods

`fn(x)` Return function value, evaluated at x: fn(x, ...)

`fngr(x)` Return list of function value and gradient, evaluated at x

`fngrhs(x)` Return list of function value, gradient, and Hessian, evaluated at x

`get_invperm()` Return integer vector of inverse of permutation used for computing Hessian

`get_nnz()` Return number of non-zero elements in lower triangle of Hessian

`get_nvars()` Return dimension (number of rows or columns) of Hessian

`get_pattern()` Return pattern matrix of lower triangle of Hessian

`get_perm()` Return integer vector of permutation used for computing Hessian

`get_perm_pattern()` Return pattern matrix of lower triangle of *permuted* Hessian

`gr(x)` Return gradient, evaluated at x: gr(x,...)

`hessian(x)` Return sparse Hessian, evaluated at x, as a dgCMatrix object.

`initialize( x, fn, gr, rows, cols, delta = 1e-07, index1 = TRUE, complex = FALSE, ... )` Initialize object with functions to compute the objective function and gradient (fn and gr), row and column indices of non-zero elements (rows and cols), an initial variable vector x at which fn and gr can be evaluated, a finite differencing parameter delta, flags for 0 or 1-based indexing (index1), whether the complex step method will be used, and other arguments (...) to be passed to fn and gr.

`partition()` Return the partitioning used to compute finite differences

`pointers(out.index1 = index1)` Return list with indices (idx) and pointers (pntr) for sparsity pattern of the compressed sparse Hessian. Since the Hessian is symmetric, the indices and pointers for row-oriented and column-oriented storage patterns are the same.

Examples

```
## Log posterior density of hierarchical binary choice model. See vignette.
set.seed(123)
data("binary_small")
N <- length(binary[["Y"]])
k <- NROW(binary[["X"]])
T <- binary[["T"]]
P <- rnorm((N+1)*k)
priors <- list(inv.Sigma = rWishart(1,k+5,diag(k))[,,1],
               inv.Omega = diag(k))
true.hess <- binary.hess(P, binary, priors)
pattern <- Matrix.to.Coord(Matrix::tril(true.hess))
str(pattern)
obj <- sparseHessianFD(P, fn=binary.f, gr=binary.grad,
       rows=pattern[["rows"]], cols=pattern[["cols"]],
                      data=binary, priors=priors)
hs <- obj$hessian(P)
all.equal(hs, true.hess)

f <- obj$fn(P) ## obj function
df <- obj$gr(P) ## gradient
fdf <- obj$fngr(P) ## list of obj function and gradient
fdfhs <- obj$fngrhs(P) ## list of obj function, gradient and Hessian.
```

`subst` *Estimate sparse Hessian*

Description

Estimate Hessian using triangular subsitution algorithm

Usage

```
subst(Y, colors, jCol, ipntr, delta, nvars, nnz)
```

Arguments

`Y`	Matrix of finite differences of gradients
`colors`	Vector of length nvars that identifies color of each variable
`jCol, ipntr`	Column indices and row pointers for non-zero elements of lower triangle of Hessian (row-oriented compressed format).
`delta`	Perturbation factor used to compute finite differences of gradients.
`nvars`	Dimension of Hessian (number of variables)
`nnz`	Number of non-zero elements in the lower triangle of the Hessian.

Details

For internal use. You should not have to call this function directly.

Value

A sparse Hessian of class dgCMatrix.

Index