Package ‘sparseHessianFD’

February 20, 2015

Type Package
Title Numerical Estimation of Sparse Hessians
Version 0.2.0
Date 2015-01-29
Author R interface code by Michael Braun
Original Fortran code by Thomas F. Coleman, Burton S. Garbow and
Jorge J. More.
Maintainer Michael Braun <braunm@smu.edu>
URL coxprofs.cox.smu.edu/braunm

Description Computes Hessian of a scalar-valued function, and returns it in
sparse Matrix format, using ACM TOMS Algorithm 636. The user must supply 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). The algorithm exploits this sparsity, so Hessians can be
computed quickly even when the number of arguments to the objective
functions is large. This package is intended to be useful for numeric
optimization (e.g., with the trustOptim package) or in simulation (e.g.,
the sparseMVN package). The underlying algorithm is ACM TOMS Algorithm 636,
written by Coleman, Garbow and More (ACM Transactions on Mathematical

License file LICENSE
LazyData true
Depends R (>= 3.1.2), Rcpp (>= 0.11.3), Matrix (>= 1.1.4), methods
Suggests testthat, numDeriv, knitr
RcppModules sparseHessianFD
LinkingTo Rcpp, RcppEigen (>= 0.3.2.3)
VignetteBuilder knitr
SystemRequirements C++11
NeedsCompilation yes
License_restricts_use yes
Estimate sparse Hessians using finite differences of gradients.

Description

Estimate sparse Hessians using finite differences of gradients.

Details

This package contains methods to return the Hessian of a function in sparse Matrix format. The user must supply 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). Sparse Hessians occur in many applications, such as log posterior densities of hierarchical models under conditional independence assumptions. This package is intended to be useful when optimizing objective functions with this structure, using optimizers than can exploit this sparsity, such as the trustOptim package.

License details are available in the LICENSE file in the source code.

References


**binary**  

*Binary choice example*

**Description**

Functions for binary choice example in the vignette.

**Usage**

```r
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 \((N+1)\times k\). First \(N\times k\) elements are heterogeneous coefficients. The remaining \(k\) elements are population parameters.
- `data` List of data matrices \(Y\) and \(X\), and choice count integer \(T\)
- `priors` List of named matrices \(\text{inv.Omega}\) and \(\text{inv.Sigma}\)

**Details**

Hessian is sparse, and returned as a dgeMatrix object

**Value**

Log posterior density, gradient and Hessian.

---

**binary-data**  

*Sample simulated data for binary choice model in vignette*

**Description**

Simulated data. See vignette. Generated from data-raw/binary.R
Coord.to.Pattern.Matrix

Pattern matrix from row and column indices.

Description

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

Usage

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

Arguments

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)

Details

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

Value

A sparse pattern matrix

Matrix.to.Coord

Row and column indices from sparse matrix.

Description

Returns list of row and column indices of the non-zero elements of a sparse matrix.

Usage

Matrix.to.Coord(M)

Arguments

M  A sparse Matrix, as defined in the Matrix package.
sparseHessianFD-class

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

Description

This class provides methods to estimate the Hessian of an objective function.

Details

Methods to estimate the Hessian of an objective function using finite differencing of gradients. An object is created with the names of R functions that return the value and the gradient, and initialized with the row and column indices of the non-zero elements in the lower triangle of the Hessian. The class contains the following methods. See vignette for usage.

- \$fn(x)\ signature(x="numeric")\: returns fn(x)
- \$gr(x)\ signature(x="numeric")\: returns gr(x)
- \$fngr(x)\ signature(x="numeric")\: returns list of fn(x) and gr(x)
- \$hessian(x)\ signature(x="numeric")\: returns sparse Hessian as dgCMatrix object
- \$hessian.init(rows, cols, direct, eps)\ Used internally to initialize Hessian with sparsity pattern
- \$nnz()\ Number of non-zero elements in lower triangle of Hessian, as provided by sparsity pattern
- \$nvars()\ Length of parameter vector that was provided to constructor

sparseHessianFD-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.
- Deprecated constructor
sparseHessianFD-deprecated

Usage

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

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

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

Arguments

H a list containing Hessian data. See details.
nvars the number of rows (and columns) in the matrix.
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 computation. 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.

Details

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


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.

Value

An object of Matrix class.
Create and initialize a new sparseHessianFD object

Description

Create and initialize a new sparseHessianFD object

Usage

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

Arguments

x A initial vector of variables at which to evaluate value, gradient and Hessian during initialization.
fn R function that returns function value
gr R function that returns the gradient of the function
rows Integer vector of row indices of non-zero elements of the lower triangle of the Hessian
cols Integer vector of column indices of non-zero elements of the lower triangle of the Hessian
direct If TRUE, use direct method for computation. 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.

Details

Indexing starts at 1, and must include the diagonal elements. Any upper triangle coordinates will be removed, and replaced with their lower triangle counterparts. The algorithms used for estimating sparse Hessians using finite differencing are described in the reference below.

This method involves a partitioning and permutation of the Hessian matrix to reduce the number of distinct finite differencing operations of the gradient. There are two methods for computing the partition and permutation: direct and indirect. The direct method tends to require more computation, but may be more accurate. We recommend the indirect method to start, so the default value is FALSE.

Here is the description of the two methods, as included in the original ACM TOMS Fortran code:

The direct method (method = 1) first determines a partition of the columns of symmetric matrix A such that two columns in a group have a non-zero element in row K only if column K is in an earlier group. Using this partition, the subroutine then computes a symmetric permutation of A consistent with the determination of A by a lower triangular substitution method.
The indirect method first computes a symmetric permutation of $A$ which minimizes the maximum number of non-zero elements in any row of $L$, where $L$ is the lower triangular part of the permuted matrix. The subroutine then partitions the columns of $L$ into groups such that columns of $L$ in a group do not have a non-zero in the same row position.

The function `new.sparse.hessian.obj` has been deprecated. Use `sparseHessianFD.new` instead.

**Value**

An object of class `sparseHessianFD`

**References**


**See Also**

`sparseHessianFD-class`
Index

*Topic package
  sparseHessianFD-package, 2

binary, 3
binary-data, 3

Coord.to.Pattern.Matrix, 4
Coord.to.Sym.Pattern.Matrix
  (sparseHessianFD-deprecated), 5

Matrix.to.Coord, 4

new.sparse.hessian.obj
  (sparseHessianFD-deprecated), 5

Rcpp_sparseHessianFD-class
  (sparseHessianFD-class), 5

sparseHessianFD
  (sparseHessianFD-class), 5
sparseHessianFD-class, 5
sparseHessianFD-deprecated, 5
sparseHessianFD-package, 2
sparseHessianFD.new, 7
Sym.CSC.to.Matrix
  (sparseHessianFD-deprecated), 5