# Package 'grassopt'

August 1, 2012

**Type** Package

**Title** Optimization over Grassmannian

**Version** 1.0

**Date** 2012-07-03

**Author** Kerstin Johnsson, Lund University

**Maintainer** Kerstin Johnsson <johnsson@maths.lth.se>

**Description** Provides functions to do optimization over a Grassmann manifold with Newton's method or the gradient descent method.

**License** GPL (>=2)

**URL** http://www.maths.lth.se/matematiklth/personal/johnsson/

**LazyLoad** yes

## R topics documented:



---

`grassopt-package` *Optimization over Grassmannian*

---

### Description

Provides functions to do optimization over a Grassmann manifold with Newton's method or the gradient descent method.

## Details

| | |
|---|---|
| Package: | grassopt |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2012-07-04 |
| License: | GPL (>= 2) |
| LazyLoad: | yes |

The function `optgr` can be used to minmize a function over a Grassmann manifold. The function `ntStep` computes the Newton step for Newton's method, and the function `geod` moves a point along a geodesic along the Grassmann manifold. For details on how the Newton step is computed, see the package vignette \"Optimization over Grassmann manifolds\".

## Author(s)

Kerstin Johnsson, Lund University

Maintainer: Kerstin Johnsson <johnsson@maths.lth.se>

## References

Edelman, A., Arias, T. A., and Smith S. T. (1998) The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, **20**(2), 303-353.

---

`geod` *Move Along a Geodesic on Grassmannian*

---

## Description

Move from a point on the Grassmann manifold Gr(n,p) represented by an n $\times$ p matrix `y` along the geodesic in the direction `v`.

## Usage

```
geod(y,v,r = 1)
```

## Arguments

| | |
|---|---|
| `y` | the point from where the geodesic start, an n $\times$ p matrix. |
| `v` | the direction of the geodesic, an n $\times$ p matrix. |
| `r` | the step size |

## Details

Each point on the Grassmannian is represented by a n $\times$ p matrix $y$. $y$ should be orthonormal, i.e. $y^T y = I$. The columns in $y$ together form an orthonormal basis for the p-dimensional subspace of $\mathbf{R}^n$ that is intended. $v$ should be in the tangent space of Gr(n,p) at $y$, i.e. $y^T v = 0$. If $v$ is not in the tangent space, it is projected onto it and a warning is given.

The update is done according to

$$y(r) = (yV\ U) \begin{pmatrix} \cos(\Sigma r) \\ \sin(\Sigma r) \end{pmatrix} V^T$$

where $v = U\Sigma V^T$ is the compact singular value decomposition of v.

## Author(s)

Kerstin Johnsson, Lund University.

## References

Edelman, A., Arias, T. A., and Smith S. T. (1998) The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, **20**(2), 303-353.

## Examples

```
y <- rbind(diag(1, 3), matrix(0, 2, 3))
v <- rbind(diag(0, 3), matrix(runif(6), 2, 3))
t(y) %*% v # This being zero shows v is a tangent
y2 <- geod(y, v, 5)
max(abs(t(y2) %*% y2 - diag(1, 3)))


v <- (diag(1,nrow(y2)) - y2%*%t(y2)) %*% matrix(runif(15), 5, 3)
y2 <- geod(y2, v, 4)
max(abs(t(y2) %*% y2 - diag(1, 3))) # This shows how much the result deviates
                                    # from being orthonormal
```

---

`ntStep` *Compute the Newton Step on Grassmannian*

---

## Description

Computes the Newton step for a function on the Grassmann manifold that should be minimized.

## Usage

```
ntStep(y,f,fy,fyy,ltol = 1e-3)
```

## Arguments

| | |
|---|---|
| y | starting point on Gr(n,p), represented by an n $\times$ p matrix. |
| f | the value of f at y. |
| fy | the gradient of f at y with respect to each element in the input matrix. fy should be an n $\times$ p matrix. |
| fyy | the Hessian of f at y with respect to each element in the input matrix. This means that fyy should be a np $\times$ np matrix where the element in place ((j-1)n + i,(l-1)n + k) is $$\frac{\partial^2 f}{\partial y_{ij} \partial y_{kl}}$$ |
| ltol | The smallest eigenvalue of the Hessian that is tolerated. If the Hessian has an eigenvalue smaller than this (if it is not positive definite), a multiple of the identity matrix is added to the Hessian so that the resulting matrix has ltol as its smallest eigenvalue. |

## Details

The Newton step is $v = \Pi(B^{-1}\nabla f)$, where $\Pi$ is the projection onto the tangent plane, $B = Hess(f) + \epsilon I$, $Hess(f)$ is a matrix interpretation of the Hessian of fun on Gr(n,p), $\epsilon$ is chosen such that the smallest eigenvalue of $B$ is ltol and $\nabla f$ is the gradient of fun along the Gr(n,p). For further information, see the reference and the package vignette.

## Value

An element v in the tangent space, i.e. an n $\times$ p matrix such that $y^T v = 0$

## Author(s)

Kerstin Johnsson, Lund University.

## References

Edelman, A., Arias, T. A., and Smith S. T. (1998) The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, **20**(2), 303-353.

## See Also

optgr

---

# optgr *Minimize a function over a Grassmann manifold*

---

## Description

Minimizing a function over a Gr(n,p), using Newton's method if a Hessian is provided, and the steepest descent method otherwise.

## Usage

```
optgr(y0, fun, grEu, hessEu = NULL, maxit = 50, tol = 1e-8, ltol = 1e-3,
      alpha = 0.2, beta = 0.5)
```

## Arguments

| | |
|---|---|
| `y0` | Starting point at the Grassmann manifold, an n $\times$ p matrix. |
| `fun` | The function to be minimized. `fun` should take an n $\times$ p matrix as input and return a scalar. |
| `grEu` | The gradient of `fun` in Euclidean space. `grEu` should take an n $\times$ p matrix as input and return an n $\times$ p matrix. |
| `hessEu` | The Hessian of `fun` in Euclidean space. `grEu` should take an n $\times$ p matrix as input and return an np $\times$ np matrix. |
| `maxit` | Maximum number of iterations. |
| `tol` | Tolerated value for $\nabla f^T B^{-1} \nabla f$ (Newton) or $\|\|\nabla f\|\|^2$ (steepest descent) at the optimal point. |
| `ltol` | Smallest eigenvalue of $B$. |
| `alpha` | Parameter for line search. |
| `beta` | Parameter for line search. |

## Details

The Newton step is $v = \Pi(B^{-1}\nabla f)$, where $\Pi$ is the projection onto the tangent plane, $B = Hess(f) + \epsilon I$, $Hess(f)$ is a matrix interpretation of the Hessian of `fun` on Gr(n,p), $\epsilon$ is chosen such that the smallest eigenvalue of $B$ is `ltol` and $\nabla f$ is the gradient of `fun` along the Gr(n,p). For further information, see the reference and the package vignette.

For the steepest descent method, the corresponding step is $-\nabla f$, i.e.\ the negative of the gradient along Gr(n,p).

For both methods, backtracking line search is used.

## Value

A list with components

| | |
|---|---|
| `value` | Value of `fun` at the found solution |
| `y` | A list with the points on the Grassmann manifold (matrices) that gone through during the iterations. |
| `lam2` | A list with the values of $\nabla f^T B^{-1} \nabla f$ (Newton) or $\|\|\nabla f\|\|^2$ (steepest descent) at the points in `y`. |

## Author(s)

Kerstin Johnsson, Lund University.

## References

Edelman, A., Arias, T. A., and Smith S. T. (1998) The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, **20**(2), 303-353.

## See Also

`ntStep`, `geod`

## Examples

```
n <- 20
p <- 4

A <- matrix(runif(n^2), n, n)
A <- (A + t(A))/2

f1 <- function(y) sum(diag(t(y) %*% A %*% y))/2
f1y <- function(y) (A + t(A)) %*% y/2
f1yy <- function(y) {
  B <- matrix(0, n*p, n*p)
  for (j in 1:p) {
    B[((j-1)*n + 1):(j*n),((j-1)*n + 1):(j*n)] <- A
  }
  return(B)
}

y0 <- rbind(diag(1, p), matrix(0, n-p, p))
opt.res <- optgr(y0, f1, f1y, f1yy)

par(mfrow = c(1, 2))
plot(sapply(opt.res$y, f1), type = 'l', ylab = 'F(Y)', xlab = 'Iteration')
optval <- sum(sort(eigen(A, only.values = TRUE)$values)[1:p])/2
abline(h = optval, lty = 2)
print(optval)
err <- abs(sapply(opt.res$y, f1) - optval)
plot(err, log = 'y', ylab = 'Absolute error', xlab = 'Iteration')
```

# Index