

Package ‘alabama’

February 14, 2012

Type Package

Title Constrained nonlinear optimization

Description Augmented Lagrangian Adaptive Barrier Minimization
Algorithm for optimizing smooth nonlinear objective functions
with constraints. Linear or nonlinear equality and inequality constraints are allowed.

Depends R (>= 2.10.1), numDeriv

Version 2011.9-1

Date 2011-09-15

Author Ravi Varadhan (with contributions from Gabor Grothendieck)

Maintainer Ravi Varadhan <rvaradhan@jhmi.edu>

URL http://www.jhsph.edu/agingandhealth/People/Faculty_personal_pages/Varadhan.html

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2011-09-16 05:51:57

R topics documented:

alabama	2
auglag	2
constrOptim.nl	6

Index	10
--------------	-----------

alabama

Constrained nonlinear optimization

Description

Augmented Lagrangian and Adaptive Barrier Minimization Algorithm for optimizing smooth nonlinear objective functions with constraints. Linear or nonlinear equality and inequality constraints are allowed.

Details

Package: alabama
Type: Package
Version: 1.0
Date: 2010-10-13
License: What license is it under?
LazyLoad: yes

Author(s)

Ravi Varadhan (with contributions from Gabor Grothendieck)

Ravi Varadhan <rvaradhan@jhmi.edu> URL:http://www.jhsph.edu/agingandhealth/People/Faculty_personal_pages/Varadhan

See Also

constrOptim, spg

auglag

Nonlinear optimization with constraints

Description

Augmented Lagrangian Minimization Algorithm for optimizing smooth nonlinear objective functions with constraints. Linear or nonlinear equality and inequality constraints are allowed.

Usage

```
auglag(par, fn, gr, hin, hin.jac, heq, heq.jac,  
control.outer=list(), control.optim = list(), ...)
```

Arguments

<code>par</code>	Starting vector of parameter values. Any initial vector, even those violating inequality constraints, may be specified. This is in contrast to <code>constrOptim.nl</code> which requires "feasible" initial values with respect to inequality constraints
<code>fn</code>	Nonlinear objective function that is to be optimized. A scalar function that takes a real vector as argument and returns a scalar that is the value of the function at that point (see details).
<code>gr</code>	The gradient of the objective function <code>fn</code> evaluated at the argument. This is a vector-function that takes a real vector as argument and returns a real vector of the same length. It defaults to "NULL", which means that gradient is evaluated numerically. Computations are dramatically faster in high-dimensional problems when the exact gradient is provided. See <i>*Example*</i> .
<code>hin</code>	a vector function specifying inequality constraints such that $hin[j] > 0$ for all j
<code>hin.jac</code>	Jacobian of <code>hin</code> . If unspecified, it will be computed using finite-difference, but computations will be faster if specified.
<code>heq</code>	a vector function specifying equality constraints such that $heq[j] = 0$ for all j
<code>heq.jac</code>	Jacobian of <code>heq</code> . If unspecified, it will be computed using finite-difference, but computations will be faster if specified.
<code>control.outer</code>	A list of control parameters to be used by the outer loop in <code>constrOptim.nl</code> . See <i>*Details*</i> for more information.
<code>control.optim</code>	A list of control parameters to be used by the unconstrained optimization algorithm in the inner loop. Identical to that used in <code>optim</code> .
<code>...</code>	Additional arguments passed to <code>fn</code> , <code>gr</code> , <code>hin</code> , <code>heq</code> . All of them must accept any specified arguments, either explicitly or by having a <code>...</code> argument, but they do not need to use them all.

Details

Argument `control.outer` is a list specifying any changes to default values of algorithm control parameters for the outer loop. Note that the names of these must be specified completely. Partial matching will not work. The list items are as follows:

`lam0`: Initial value for the Lagrangian parameter.

`sig0`: A scaling parameter for penalty term that is augmented to the Lagrangian.

`eps`: Tolerance for convergence of outer iterations of the barrier and/or augmented lagrangian algorithm

`itmax`: Maximum number of outer iterations.

`ilack.max`: Maximum number of outer iterations where no change in parameters is tolerated.

`trace`: A logical variable indicating whether information on outer iterations should be printed out. If TRUE, at each outer iteration information is displayed on: (i) how well the inequality and equalities are satisfied, (ii) current parameter values, and (iii) current objective function value.

`method`: Unconstrained optimization algorithm in `optim()` to be used; default is the "BFGS" variable metric method.

NMinit: A logical variable indicating whether "Nelder-Mead" algorithm should be used in optim() for the first outer iteration.

i.scale: A vector of length equal to number of inequalities that may be used to scale the inequalities or it can be a scalar in which case all the inequalities are scaled by the same value.

e.scale: A vector of length equal to number of equalities that may be used to scale the equalities or it can be a scalar in which case all the equalities are scaled by the same value.

Value

A list with the following components:

par	Parameters that optimize the nonlinear objective function, satisfying constraints, if convergence is successful.
value	The value of the objective function at termination.
counts	A vector of length 2 denoting the number of times the objective fn and the gr were evaluated, respectively.
convergence	An integer code indicating type of convergence. 0 indicates successful convergence. Positive integer codes indicate failure to converge.
outer.iterations	Number of outer iterations
lambda	Values of the Lagrangian parameter. This is a vector of same length as the total number of inequalities and equalities. It must be zero for inactive inequalities; non-negative for active inequalities; and can have any sign for equalities.
sigma	Value of augmented penalty parameter for the quadratic term
gradient	Gradient of the augmented Lagrangian function at convergence. It should be small.
hessian	Hessian of the augmented Lagrangian function at convergence. It should be positive (negative) definite for minimization (maximization)
ineq	Values of inequality constraints at convergence. All of them must be non-negative
equal	Values of equality constraints at convergence. All of them must be close to zero.
kkt1	A logical variable indicating whether or not the first-order KKT conditions were satisfied.
kkt2	A logical variable indicating whether or not the second-order KKT conditions were satisfied.

Author(s)

Ravi Varadhan, Center on Aging and Health, Johns Hopkins University.

References

Lange K, *Optimization*, 2004, Springer.

Madsen K, Nielsen HB, Tingleff O, *Optimization With Constraints*, 2004, IMM, Technical University of Denmark.

See Also

See Also [constrOptim.nl](#).

Examples

```
fn <- function(x) (x[1] + 3*x[2] + x[3])^2 + 4 * (x[1] - x[2])^2

gr <- function(x) {
  g <- rep(NA, 3)
  g[1] <- 2*(x[1] + 3*x[2] + x[3]) + 8*(x[1] - x[2])
  g[2] <- 6*(x[1] + 3*x[2] + x[3]) - 8*(x[1] - x[2])
  g[3] <- 2*(x[1] + 3*x[2] + x[3])
  g
}

heq <- function(x) {
  h <- rep(NA, 1)
  h[1] <- x[1] + x[2] + x[3] - 1
  h
}

heq.jac <- function(x) {
  j <- matrix(NA, 1, length(x))
  j[1, ] <- c(1, 1, 1)
  j
}

hin <- function(x) {
  h <- rep(NA, 1)
  h[1] <- 6*x[2] + 4*x[3] - x[1]^3 - 3
  h[2] <- x[1]
  h[3] <- x[2]
  h[4] <- x[3]
  h
}

hin.jac <- function(x) {
  j <- matrix(NA, 4, length(x))
  j[1, ] <- c(-3*x[1]^2, 6, 4)
  j[2, ] <- c(1, 0, 0)
  j[3, ] <- c(0, 1, 0)
  j[4, ] <- c(0, 0, 1)
  j
}

# Note: 'auglag' accepts infeasible starting values
#
p0 <- runif(3)
ans <- auglag(par=p0, fn=fn, gr=gr, heq=heq, heq.jac=heq.jac, hin=hin, hin.jac=hin.jac)
```

```
# Not specifying the gradient and the Jacobians
set.seed(12)
p0 <- runif(3)
ans2 <- auglag(par=p0, fn=fn, heq=heq, hin=hin)
```

constrOptim.nl

Nonlinear optimization with constraints

Description

Augmented Lagrangian Adaptive Barrier Minimization Algorithm for optimizing smooth nonlinear objective functions with constraints. Linear or nonlinear equality and inequality constraints are allowed.

Usage

```
constrOptim.nl(par, fn, gr = NULL,
hin = NULL, hin.jac = NULL, heq = NULL, heq.jac = NULL,
control.outer=list(), control.optim = list(), ...)
```

Arguments

par	starting vector of parameter values; initial vector must be "feasible"
fn	Nonlinear objective function that is to be optimized. A scalar function that takes a real vector as argument and returns a scalar that is the value of the function at that point (see details).
gr	The gradient of the objective function fn evaluated at the argument. This is a vector-function that takes a real vector as argument and returns a real vector of the same length. It defaults to "NULL", which means that gradient is evaluated numerically. Computations are dramatically faster in high-dimensional problems when the exact gradient is provided. See <i>*Example*</i> .
hin	a vector function specifying inequality constraints such that $hin[j] > 0$ for all j
hin.jac	Jacobian of hin. If unspecified, it will be computed using finite-difference, but computations will be faster if specified.
heq	a vector function specifying equality constraints such that $heq[j] = 0$ for all j
heq.jac	Jacobian of heq. If unspecified, it will be computed using finite-difference, but computations will be faster if specified.
control.outer	A list of control parameters to be used by the outer loop in constrOptim.nl. See <i>*Details*</i> for more information.
control.optim	A list of control parameters to be used by the unconstrained optimization algorithm in the inner loop. Identical to that used in optim.
...	Additional arguments passed to fn, gr, hin, heq. All of them must accept any specified arguments, either explicitly or by having a ... argument, but they do not need to use them all.

Details

Argument `control.outer` is a list specifying any changes to default values of algorithm control parameters for the outer loop. Note that the names of these must be specified completely. Partial matching will not work. The list items are as follows:

`mu0`: A scaling parameter for barrier penalty for inequality constraints.

`sig0`: A scaling parameter for augmented lagrangian for equality constraints

`eps`: Tolerance for convergence of outer iterations of the barrier and/or augmented lagrangian algorithm

`itmax`: Maximum number of outer iterations.

`trace`: A logical variable indicating whether information on outer iterations should be printed out. If TRUE, at each outer iteration information is displayed on: (i) how well the inequality and equalities are satisfied, (ii) current parameter values, and (iii) current objective function value.

`method`: Unconstrained optimization algorithm in `optim()` to be used; default is the "BFGS" variable metric method.

`NMini`: A logical variable indicating whether "Nelder-Mead" algorithm should be used in `optim()` for the first outer iteration.

Value

A list with the following components:

<code>par</code>	Parameters that optimize the nonlinear objective function, satisfying constraints, if convergence is successful.
<code>value</code>	The value of the objective function at termination.
<code>convergence</code>	An integer code indicating type of convergence. 0 indicates successful convergence. Positive integer codes indicate failure to converge.
<code>message</code>	Text message indicating the type of convergence or failure.
<code>outer.iterations</code>	Number of outer iterations
<code>lambda</code>	Value of augmented Lagrangian penalty parameter
<code>sigma</code>	Value of augmented Lagrangian penalty parameter for the quadratic term
<code>barrier.value</code>	Reduction in the value of the function from its initial value. This is negative in maximization.
<code>K</code>	Residual norm of equality constraints. Must be small at convergence.
<code>counts</code>	A vector of length 2 denoting the number of times the objective fn and the gr were evaluated, respectively.

Author(s)

Ravi Varadhan, Center on Aging and Health, Johns Hopkins University.

References

Lange K, *Optimization*, 2004, Springer.

Madsen K, Nielsen HB, Tingleff O, *Optimization With Constraints*, 2004, IMM, Technical University of Denmark.

See Also

See Also [auglag](#), [constrOptim](#).

Examples

```
fn <- function(x) (x[1] + 3*x[2] + x[3])^2 + 4 * (x[1] - x[2])^2
```

```
gr <- function(x) {
  g <- rep(NA, 3)
  g[1] <- 2*(x[1] + 3*x[2] + x[3]) + 8*(x[1] - x[2])
  g[2] <- 6*(x[1] + 3*x[2] + x[3]) - 8*(x[1] - x[2])
  g[3] <- 2*(x[1] + 3*x[2] + x[3])
  g
}
```

```
heq <- function(x) {
  h <- rep(NA, 1)
  h[1] <- x[1] + x[2] + x[3] - 1
  h
}
```

```
heq.jac <- function(x) {
  j <- matrix(NA, 1, length(x))
  j[1, ] <- c(1, 1, 1)
  j
}
```

```
hin <- function(x) {
  h <- rep(NA, 1)
  h[1] <- 6*x[2] + 4*x[3] - x[1]^3 - 3
  h[2] <- x[1]
  h[3] <- x[2]
  h[4] <- x[3]
  h
}
```

```
hin.jac <- function(x) {
  j <- matrix(NA, 4, length(x))
  j[1, ] <- c(-3*x[1]^2, 6, 4)
  j[2, ] <- c(1, 0, 0)
  j[3, ] <- c(0, 1, 0)
  j[4, ] <- c(0, 0, 1)
}
```

```
j
}

set.seed(12)
p0 <- runif(3)
ans <- constrOptim.nl(par=p0, fn=fn, gr=gr, heq=heq, heq.jac=heq.jac, hin=hin, hin.jac=hin.jac)

# Not specifying the gradient and the Jacobians
set.seed(12)
p0 <- runif(3)
ans2 <- constrOptim.nl(par=p0, fn=fn, heq=heq, hin=hin)
```

Index

*Topic **optimize**

alabama, [2](#)

auglag, [2](#)

constrOptim.nl, [6](#)

adpbar (constrOptim.nl), [6](#)

alabama, [2](#)

alabama (constrOptim.nl), [6](#)

auglag, [2](#), [8](#)

auglag1 (auglag), [2](#)

auglag2 (auglag), [2](#)

auglag3 (auglag), [2](#)

augpen (constrOptim.nl), [6](#)

constrOptim, [8](#)

constrOptim.nl, [5](#), [6](#)