Package ‘Dykstra’

January 20, 2025

Type Package

Title Quadratic Programming using Cyclic Projections

Version 1.0-0

Date 2018-02-09

Author Nathaniel E. Helwig <helwig@umn.edu>

Maintainer Nathaniel E. Helwig <helwig@umn.edu>

Description Solves quadratic programming problems using Richard L. Dykstra's cyclic projection algorithm. Routine allows for a combination of equality and inequality constraints. See Dykstra (1983) <doi:10.1080/01621459.1983.10477029> for details.

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-09 18:34:56 UTC

Contents

dykstra 1

Index **5**

`dykstra` *Solve a Quadratic Programming Problem via Dykstra’s Algorithm*

Description

This function uses Dykstra’s cyclic projection algorithm to solve quadratic programming problems of the form

$$-d^T x + (1/2)x^T Dx$$

subject to $A^T x >= b$ where D is a positive definite (or positive semidefinite) matrix.

Usage

```
dykstra(Dmat, dvec, Amat, bvec, meq = 0, factorized = FALSE,
        maxit = NULL, eps = NULL)
```

Arguments

`Dmat`	Quadratic program matrix D of order $n \times n$.
`dvec`	Quadratic program vector d of length n.
`Amat`	Constraint matrix A of order $n \times r$.
`bvec`	Constraint vector b of length r. Defaults to vector of zeros.
`meq`	First `meq` constraints are equality constraints (remaining are inequality constraints). Defaults to zero.
`factorized`	If `TRUE`, argument `Dmat` is R^{-1} where $R^T R = D$.
`maxit`	Maximum number of iterations (cycles). Defaults to $30n$.
`eps`	Numeric tolerance. Defaults to `n * .Machine$double.eps`.

Details

Arguments 1-6 of the `dykstra` function are inspired by (and identical to) the corresponding arguments of the `solve.QP` function in the **quadprog** package.

Value

`solution`	Vector x that minimizes quadratic function subject to constraints.
`value`	Value of quadratic function at `solution`. Will be `NA` if `factorized = TRUE`.
`unconstrained`	Vector $x_0 = D^{-1}d$ that minimizes quadratic function ignoring constraints.
`iterations`	Number of iterations (cycles) of the algorithm.
`converged`	`TRUE` if algorithm converged. `FALSE` if iteration limit exceeded.

Note

For positive semidefinite D, a small constant is added to each eigenvalue of D before solving the quadratic programming problem.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Dykstra, Richard L. (1983). An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, Volume 78, Issue 384, 837-842. doi: 10.1080/01621459.1983.10477029

Examples

```
###  EXAMPLE 1: Generic Quadratic Programming Problem  ###

# constraint 1 (equality): coefficients sum to 1
# constraints 2-4 (inequality): coefficients non-negative

# define QP problem
Dmat <- diag(3)
dvec <- c(1, 1.5, 1)
Amat <- cbind(rep(1, 3), diag(3))
bvec <- c(1, 0, 0, 0)

# solve QP problem
dykstra(Dmat, dvec, Amat, bvec, meq = 1)

# solve QP problem (factorized = TRUE)
dykstra(Dmat, dvec, Amat, bvec, meq = 1, factorized = TRUE)



###  EXAMPLE 2: Regression with Non-Negative Coefficients  ###

# generate regression data
set.seed(1)
nobs <- 100
nvar <- 5
X <- matrix(rnorm(nobs*nvar), nobs, nvar)
beta <- c(0, 1, 0.3, 0.7, 0.1)
y <- X %*% beta + rnorm(nobs)

# define QP problem
Dmat <- crossprod(X)
dvec <- crossprod(X, y)
Amat <- diag(nvar)

# solve QP problem
dykstra(Dmat, dvec, Amat)

# solve QP problem (factorized = TRUE)
Rmat <- chol(Dmat)
Rinv <- solve(Rmat)
dykstra(Rinv, dvec, Amat, factorized = TRUE)



###  EXAMPLE 3: Isotonic Regression  ###

# generate regression data
set.seed(1)
n <- 50
x <- 1:n
y <- log(x) + rnorm(n)
```

```
# define QP problem
Dmat <- diag(n)
Amat <- Dmat[, 2:n] - Dmat[, 1:(n-1)]

# solve QP problem
dyk <- dykstra(Dmat, y, Amat)
dyk

# plot results
plot(x, y)
lines(x, dyk$solution)



###  EX 4: Large Non-Negative Quadratic Program  ###

# define QP problem
set.seed(1)
n <- 1000
Dmat <- Amat <- diag(n)
dvec <- runif(n, min = -2)

# solve QP problem with dykstra
dyk <- dykstra(Dmat, dvec, Amat)
dyk
```

Index

∗ **optimize**
    dykstra, 1

dykstra, 1, *2*

solve.QP, *2*