Package: LowRankQP (via r-universe)

September 20, 2024

Version 1.0.6

Date 2023-08-10

Title Low Rank Quadratic Programming

Maintainer John T. Ormerod <john.ormerod@sydney.edu.au>

Description Solves quadratic programming problems where the Hessian is represented as the product of two matrices. Thanks to Greg Hunt for helping getting this version back on CRAN. The methods in this package are described in: Ormerod, Wand and Koch (2008) ``Penalised spline support vector classifiers: computational issues'' <doi:10.1007/s00180-007-0102-8>.

License GPL (>= 2)

NeedsCompilation yes

Author John T. Ormerod [aut, cre], Matt P. Wand [aut]

Repository CRAN

Date/Publication 2023-08-10 07:00:11 UTC

Contents

`LowRankQP` *Solve Low Rank Quadratic Programming Problems*

Description

This routine implements a primal-dual interior point method solving quadratic programming problems of the form

$$\begin{array}{ll} \text{min} & d^T alpha + 1/2 alpha^T H alpha \\ \text{such that} & A alpha = b \end{array}$$

$$0 <= alpha <= u$$

with dual

$$\begin{array}{ll} \min & 1/2alpha^T Halpha + beta^T b + xi^T u \\ \text{such that} & Halpha + c + A^T beta - zeta + xi = 0 \\ & xi, zeta >= 0 \end{array}$$

where $H = V$ if V is square and $H = VV^T$ otherwise.

Usage

```
LowRankQP(Vmat,dvec,Amat,bvec,uvec,method="PFCF",verbose=FALSE,niter=200,epsterm=1.0E-8)
```

Arguments

`Vmat`	matrix appearing in the quadratic function to be minimized.
`dvec`	vector appearing in the quadratic function to be minimized.
`Amat`	matrix defining the constraints under which we want to minimize the quadratic function.
`bvec`	vector holding the values of b (defaults to zero).
`uvec`	vector holding the values of u.
`method`	Method used for inverting H+D where D is full rank diagonal. If V is square: • 'LU': Use LU factorization. (More stable) • 'CHOL': Use Cholesky factorization. (Faster) If V is not square: • 'SMW': Use Sherman-Morrison-Woodbury (Faster) • 'PFCF': Use Product Form Cholesky Factorization (More stable)
`verbose`	Display iterations and termination statistics.
`niter`	Number of iteration to perform.
`epsterm`	Termination tolerance. See equation (12) of Ormerod et al (2008).

Value

a list with the following components:

`alpha`	vector containing the solution of the quadratic programming problem.
`beta`	vector containing the solution of the dual of quadratic programming problem.
`xi`	vector containing the solution of the dual quadratic programming problem.
`zeta`	vector containing the solution of the dual quadratic programming problem.

References

Ormerod, J.T., Wand, M.P. and Koch, I. (2008). Penalised spline support vector classifiers: computational issues, Computational Statistics, 23, 623-641.

Boyd, S. and Vandenberghe, L. (2004). Convex Optimization. Cambridge University Press.

Ferris, M. C. and Munson, T. S. (2003). Interior point methods for massive support vector machines. SIAM Journal on Optimization, 13, 783-804.

Fine, S. and Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. Journal of Machine Learning Research, 2, 243-264.

B. Sch\"olkopf and A. J. Smola. (2002). Learning with Kernels. The MIT Press, Cambridge, Massachusetts.

Examples

```
library(LowRankQP)

# Assume we want to minimize: (0 -5 0 0 0 0) %*% alpha + 1/2 alpha[1:3]^T alpha[1:3]
# under the constraints:      A^T alpha = b
# with b = (-8,  2,  0 )^T
# and      (-4   2   0 )
#      A = (-3   1  -2 )
#          ( 0   0   1 )
#          (-1   0   0 )
#          ( 0  -1   0 )
#          ( 0   0  -1 )
#  alpha >= 0
#
# (Same example as used in quadprog)
#
# we can use LowRankQP as follows:

Vmat          <- matrix(0,6,6)
diag(Vmat)    <- c(1, 1,1,0,0,0)
dvec          <- c(0,-5,0,0,0,0)
Amat          <- matrix(c(-4,-3,0,-1,0,0,2,1,0,0,-1,0,0,-2,1,0,0,-1),6,3)
bvec          <- c(-8,2,0)
uvec          <- c(100,100,100,100,100,100)
LowRankQP(Vmat,dvec,t(Amat),bvec,uvec,method="CHOL")

# Now solve the same problem except use low-rank V

Vmat          <- matrix(c(1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0),6,3)
dvec          <- c(0,-5,0,0,0,0)
Amat          <- matrix(c(-4,-3,0,-1,0,0,2,1,0,0,-1,0,0,-2,1,0,0,-1),6,3)
bvec          <- c(-8,2,0)
uvec          <- c(100,100,100,100,100,100)
LowRankQP(Vmat,dvec,t(Amat),bvec,uvec,method="SMW")
```

Index