MAJORIZING A MULTIVARIATE POLYNOMIAL OVER THE UNIT SPHERE, WITH APPLICATIONS

JAN DE LEEUW AND PATRICK J.F. GROENEN

1. Problem

The problem studied in this paper is to minimize a polynomial $P : \mathbb{R}^m \Rightarrow \mathbb{R}$ over the unit sphere $\mathbb{S} = \{x \mid x'x = 1\}$. Clearly the problem is well-defined, because the minimum always exists.

We use the standard notation $P(x) = \sum_\alpha p_\alpha x^\alpha$ for multivariate polynomials, where α are vectors of m integers, and

$$x^\alpha = \prod_{j=1}^m x_j^{\alpha_j}.$$

2. Algorithm

We use quadratic majorization [Böhning and Lindsay, 1988; De Leeuw and Lange, 2009]. This requires us to find an upper bound for the quadratic term in the Taylor expansion of P. So, using g and H for the gradient and Hessian,

$$P(x) = P(y) + (x - y)'g(y) + \frac{1}{2}(x - y)'H(z)(x - y), \tag{1}$$

Date: Wednesday 19th October, 2011 — 14h 15min — Typeset in LUCIDA BRIGHT.

2000 *Mathematics Subject Classification.* 49M20.

Key words and phrases. Majorization, Polynomial Optimization.

where z is on the line between x and y. Thus

$$P(x) \leq P(y) + (x-y)'g(y) + \frac{1}{2}|(x-y)'H(z)(x-y)|. \tag{2}$$

Now, using $\rho(z)$ for the spectral norm of $H(z)$,

$$|(x-y)'H(z)(x-y)| \leq \rho(z)(x-y)'(x-y), \tag{3}$$

and for any submultiplicative matrix norm $\|H(z)\|$

$$|(x-y)'H(z)(x-y)| \leq \|H(z)\|(x-y)'(x-y) \tag{4}$$

because $\rho(z) \leq \|H(z)\|$.

Now each element of the gradient and the Hessian is, again, a multivariate polynomial. This can be used in finding upper bounds for some conveniently chosen matrix norms. The easiest choice, perhaps, is the ℓ_1 norm, for which

$$\|H(z)\|_1 = \sum_{i=1}^m \sum_{j=1}^m |h_{ij}(z)| = \sum_{i=1}^m \sum_{j=1}^m |\sum_\alpha p_{\alpha ij} z^\alpha| \leq \sum_{i=1}^m \sum_{j=1}^m \sum_\alpha |p_{\alpha ij}|, \tag{5}$$

because if x and y are on the sphere we also have z in the sphere, and thus $|z^\alpha| \leq 1$. The upper bound in (5) is K_1.

This can be improved by using the max-rowsum-norm in (4).

$$\|H(z)\|_\infty = \max_{i=1}^m \sum_{j=1}^m |h_{ij}(z)| = \max_{i=1}^m \sum_{j=1}^m |\sum_\alpha p_{\alpha ij} z^\alpha| \leq \max_{i=1}^m \sum_{j=1}^m \sum_\alpha |p_{\alpha ij}|, \tag{6}$$

Define K_∞ as the bound on the right hand side.

It follows that the function

$$Q(x \mid y) = P(y) + (x-y)'g(y) + \frac{1}{2}K(x-y)'(x-y), \tag{7}$$

with K equal to either K_1 or K_∞, is a majorization of P at y. A step of the majorization algorithm minimizes $Q(x \mid y)$ over $x \in \mathbb{S}$.

The minimum is attained at

$$\hat{x} = \frac{y - \frac{1}{K} g(y)}{\| y - \frac{1}{K} g(y) \|},$$

and thus the majorization algorithm is the fixed step projected gradient algorithm

$$x^{(k+1)} = \frac{x^{(k)} - \frac{1}{K} g(x^{(k)})}{\| x^{(k)} - \frac{1}{K} g(x^{(k)}) \|}. \tag{8}$$

3. Sharper Majorization

The inequality $|z^\alpha| \leq 1$ can be improved. Define

$$\tau(\alpha, z) = \log \prod_{j=1}^m |z_j|^{\alpha_j} = \frac{1}{2} \sum_{j=1}^m \alpha_j \log |z_j^2|.$$

The function $\tau(\alpha, z)$ attains its maximum $\tau_\star(\alpha)$ over z with $\sum_{j=1}^m z_j^2 = 1$ at

$$z_j^2 = \frac{\alpha_j}{\sum_{j=1}^m \alpha_j},$$

and the maximum is equal to

$$\tau_\star(\alpha) = \frac{1}{2} \sum_{j=1}^m \alpha_j \log \frac{\alpha_j}{\sum_{j=1}^m \alpha_j} = \frac{1}{2} \left\{ \sum_{j=1}^m \alpha_j \log \alpha_j - A \log A \right\},$$

where $A = \sum_{j=1}^m \alpha_j$. Throughout we use the convention that $0 \log 0 = 0$.

Thus choosing

$$K_0 = \max_{i=1}^m \sum_{j=1}^m \sum_\alpha \exp(\tau_\star(\alpha)) |p_{\alpha i j}|, \tag{9}$$

is a sharper majorization of P at y, and a step of this improved majorization algorithm minimizes (7) with this new K_0 over $x \in \mathbb{S}$.

4. Examples

4.1. **Small.** Our first example uses the polynomial

$$P(x,y) = \begin{array}{c} \\ x^0 \\ x^1 \\ x^2 \\ x^3 \end{array} \begin{array}{c} \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{pmatrix} 3 & -5 & 0 \\ 4 & 0 & 0 \\ 0 & 0 & 1 \\ -4 & 0 & 0 \end{pmatrix} \end{array}.$$

The gradient is

$$g_1(x,y) = \begin{array}{c} \\ x^0 \\ x^1 \\ x^2 \\ x^3 \end{array} \begin{array}{c} \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{pmatrix} 4 & 0 & 0 \\ 0 & 0 & 2 \\ -12 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \qquad g_2(x,y) = \begin{array}{c} \\ x^0 \\ x^1 \\ x^2 \\ x^3 \end{array} \begin{array}{c} \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{pmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array},$$

and the Hessian is

$$h_{11}(x,y) = \begin{array}{c} \\ x^0 \\ x^1 \\ x^2 \\ x^3 \end{array} \begin{array}{c} \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 2 \\ -24 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \qquad h_{22}(x,y) = \begin{array}{c} \\ x^0 \\ x^1 \\ x^2 \\ x^3 \end{array} \begin{array}{c} \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}.$$

$$h_{12}(x,y) = h_{21}(x,y) = \begin{array}{c} \\ x^0 \\ x^1 \\ x^2 \\ x^3 \end{array} \begin{array}{c} \begin{array}{ccc} y^0 & y^1 & y^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

It follows that $K_1 = 32$, while $K_\infty = 30$. For K_0 we have to compute the optimal weights $\exp(\tau_\star(\alpha))$. They are

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 1.0000000 & 1.0000000 & 1.0000000 \\ 1.0000000 & 0.5000000 & 0.3849002 \\ 1.0000000 & 0.3849002 & 0.2500000 \\ 1.0000000 & 0.3247595 & 0.1859032 \end{pmatrix} \end{array}.$$

This shows that $K_0 = 28$.

In the Appendix we give R code that implements the majorization algorithm using the `multipol` package [Hankin, 2009]. Throughout the paper we iterate until the function decreases less than $\epsilon = 1e-10$. The iterative majorization algorithm (8), started at $x = 1$ and $y = 0$, converges to the function value -2.805344, attained at $x = -0.3588192$ and $y = 0.9334071$. It can be verified that this is indeed the unique minimum on the circle.

The small differences between K_1, K_∞ and K_0 already indicate that the choice of bound will not be very important in this case. For K_1 we need 55 iterations, for K_∞ 47 iterations, and for K_0 44 iterations.

4.2. **Full.** We also tried the three-variable example

```
pisfull <- as.multipol (array (1 : 64, c (4, 4, 4)
   ))
```

We start with $x = 1$ and $y = z = 0$. For bound $K_1 = 54000$, the algorithm converges smoothly, albeit slowly, to a minimum of -47.1303347324 at $x = 0.3340020, y = 0.3194996$, and $z = -0.8867709$. This requires 4025 iterations (if $\epsilon = 1e-10$). Using $K_\infty = 20520$ brings this down to 1907 iterations. Using $K_0 = 5143.482$ as an upper bound leads to 510 iterations.

4.3. **Sparse.** A four-variable example with all coefficients zero or one is given by

```
set.seed(12345)
psparse <- as.multipol (array (sample (c (0, 1),
   81, replace = TRUE), c (3, 3, 3, 3)))
```

For $K_1 = 630$ we need 3405 iterations, for $K_\infty = 166$ we need 960, and for $K_0 = 49.94036$ we need 308.

4.4. **Large.** This example involves large coefficients, and a relatively high polynomial order.

```
plarge <- as.multipol (array (sample(1:5^5,5^5), c
   (5,5,5,5,5)))
```

This taxes the method greatly. If we set $\epsilon = 1e-3$ and start at `c(1,0,0,0,0)` the algorithm with $K_0 = 1,896,223$ converges in 3,054 iterations to a function value of -2115. What is perhaps most interesting is that we observe a long period in which the iterations take larger and larger steps, instead of smaller and smaller steps. This seems to happen in order to get away from a non-optimal stationary point, and arrive at a lower function value.

If we use $K_\infty = 97,965,560$ we see no convergence after 10,000 iterations. The function value is -523, the change in function value in an iteration is still 0.030. We have not tried K_1.

5. Applications

5.1. **Eigenvalues and Singular Values.** Consider the symmetric matrix A

```
     [,1] [,2] [,3]
[1,]    4   -2   -2
[2,]   -2    5   -2
[3,]   -2   -2    6
```

Its eigenvalues and eigenvectors are

```
$values
[1] 7.6298133 6.4802788 0.8899079

$vectors
           [,1]       [,2]      [,3]
[1,] -0.1921651  0.7154086 0.6717612
```

```
[2,] -0.4972795 -0.6611152 0.5618183
[3,]  0.8460412 -0.2260912 0.4828013
```

The quadratic form defined by A is the polynomial

```
> peigenv
, , z^0

    y^0 y^1 y^2
x^0   0   0   5
x^1   0  -4   0
x^2   4   0   0

, , z^1

    y^0 y^1 y^2
x^0   0  -4   0
x^1  -4   0   0
x^2   0   0   0

, , z^2

    y^0 y^1 y^2
x^0   6   0   0
x^1   0   0   0
x^2   0   0   0
```

In this case $K_\infty = K_0 = 20$ and the majorization algorithm converges in 14 iterations to the smallest eigenvalue and corresponding eigenvector (started with $x = 1$ and $y = z = 0$ and using $\epsilon = 1e - 10$). The majorization algorithm is equivalent in this case to the power method applied to the matrix $I - \frac{2}{K}V$.

Singular values of a rectangular A can be handled similarly by considering the bilinear form $x'Ay$, or the eigenvalue problem for $\begin{bmatrix} I & A \\ A' & I \end{bmatrix}$.

5.2. **Quadratic on Sphere.** Consider the problem of minimizing $p(x) = c - 2b'x + x'Ax$ on the sphere. This is a classical problem in numerical analysis, ever since the important papers by Forsythe and Golub [1965] and Golub [1973]. It leads to the "secular equation" $(A - \lambda I)x = b$, which must be solved over $x'x = 1$. There are important applications to trust-region methods for general nonlinear programming [Conn et al., 2000]. Note that there is no need for A to be positive definite or non-singular in this case, because the minimum on the sphere is always attained.

Let us use the same A as before, set $c = 1$, and set $b_1 = -b_2 = -3$ and $b_3 = 0$. This leads to the polynomial

```
, , z^0

    y^0 y^1 y^2
x^0   1  -6   5
x^1   6  -4   0
x^2   4   0   0

, , z^1

    y^0 y^1 y^2
x^0   0  -4   0
x^1  -4   0   0
x^2   0   0   0

, , z^2
```

```
17       y^0 y^1 y^2
18  x^0    6   0   0
19  x^1    0   0   0
20  x^2    0   0   0
```

As before $K_0 = 20$. From $x = 1$ and $y = z = 0$ we converge to the minimum -0.8825536818 in 74 iterations.

5.3. **Squared Euclidean MDS.** The squared distance scaling can be written as minimizing the polynomial $p(x) = \sum_{i=1}^n w_i(\delta_i - x'A_ix)^2$, where the A_i are known positive semi-definite matrices and the weights w_i and dissimilarities δ_i are known non-negative numbers [De Leeuw, 1993].

By transforming the variables linearly we can assume without loss of generality that $\sum_{i=1}^n w_i\delta_iA_i = I$ and we can scale the data such that $\sum_{i=1}^n w_i\delta_i^2 = 1$. Thus $p(x) = 1 - 2x'x + \sum_{i=1}^n w_i(x'A_ix)^2$. Using the homogeneity in the problem, as in De Leeuw [1977], shows that this is equivalent to minimizing $\sum_{i=1}^n w_i(x'A_ix)^2$ on the unit sphere, which can be done by our majorization algorithm.

6. Discussion

Clearly the same approach to algorithm construction can be used when majorizing a general twice-differentiable function on a sphere, as long as we can easily calculate upper bounds for the elements of the Hessian.

By making the sphere large enough we can also tackle the problem of minimizing functions with continuous but not necessarily bounded derivatives.

REFERENCES

D. Böhning and B.G. Lindsay. Monotonicity of Quadratic-approximation Algorithms. *Annals of the Institute of Statistical Mathematics*, 40(4):641–663, 1988.

A.R. Conn, N.I.M. Gould, and P.L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, PA, 2000.

J. De Leeuw. Applications of Convex Analysis to Multidimensional Scaling. In J.R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, editors, *Recent developments in statistics*, pages 133–145, Amsterdam, The Netherlands, 1977. North Holland Publishing Company.

J. De Leeuw. Fitting Distances by Least Squares. Preprint Series 130, UCLA Department of Statistics, 1993. URL `http://preprints.stat.ucla.edu/download.php?paper=130`.

J. De Leeuw. Using Jacobi Plane Rotations in R. Preprint Series 556, UCLA Department of Statistics, Los Angeles, CA, 2008. URL `http://preprints.stat.ucla.edu/556/jacobi.pdf`.

J. De Leeuw and K. Lange. Sharp Quadratic Majorization in One Dimension. *Computational Statistics and Data Analysis*, 53:2471–2484, 2009.

G.E. Forsythe and G.H. Golub. On the Stationary Values of a Second Degree Polynomial on the Unit Sphere. *Journal of the Society for Industrial and Applied Mathematics*, 13:1050–1068, 1965.

G.H. Golub. Some Modified Matrix Eigenvalue Problems. *SIAM Review*, 15:318–334, 1973.

R.K.S. Hankin. *multipol: multivariate polynomials*, 2009. URL `http://CRAN.R-project.org/package=multipol`. R package version 1.0-4.

Appendix A. Code

```
library(multipol)
library(apl)

set.seed(12345)

psmally <- as.multipol (array (c (3, 4, 0, -4, -5, 0, 0, 0, 0, 0, 1, 0),
    c(4, 3)))
pisfull <- as.multipol (array (1 : 64, c (4, 4, 4)))
psparse <- as.multipol (array (sample (c (0, 1), 81, replace = TRUE), c
    (3, 3, 3, 3)))
plarger <- as.multipol (array (sample (1 : 5^5, 5^5), c(5, 5, 5, 5, 5)))
peigenv <- as.multipol (array (c (0, 0, 4, 0, -4, 0, 5, 0, 0, 0, -4, 0,
    -4, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0), c (3, 3, 3)))
pquadra <- as.multipol (array (c (1, 6, 4, -6, -4, 0, 5, 0, 0, 0, -4, 0,
    -4, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0), c (3, 3, 3)))

majPol <- function (p, xold, step = 3, itmax = 100, eps = 1e-10, verbose
    = TRUE) {
    if (!is.multipol (p)) {
        p <- as.multipol (p)
    }
    d <- dim (p)
    a <- astar (d)
    r <- length (d)
    s <- 1 : r
    g <- lapply (s, function (i) deriv (p, i))
    h <- lapply (g, function (f) lapply (s, function (i) deriv (f,i)))
    K <- switch (step, k2 (h), k1 (h), k0 (h, a))
    fold <- as.function (p) (xold)
    itel <- 1
    repeat {
        grad <- sapply(s, function (i) as.function (g [[i]]) (xold))
        xraw <- xold - grad / K
        xnew <- xraw / sqrt (sum (xraw ^ 2))
        fnew <- as.function (p) (xnew)
        if (verbose) {
            cat("Iteration: ",
                formatC (itel, digits = 6, width = 6),
                "   f old :   ",
                formatC (fold, digits = 10, width = 15, format="f"),
                "   f new :   ",
```

```
                formatC (fnew, digits = 10, width = 15, format="f"),
                "    diff :   ",
                formatC (fold - fnew, digits = 10, width = 15, format="f"
                    ),
                "\n")
        }
        if ((itel == itmax) || ((fold - fnew) < eps)) {
            return (list (K = K, itel = itel, f = fnew, x = xnew))
        }
        itel <- itel + 1
        fold <- fnew
        xold <- xnew
    }
}

k2 <- function (h) {
        return (sum (abs (unlist (h))))
}

k1 <- function (h) {
        return (max (sapply (h, function (x) sum (abs (unlist(x))))))
}

k0 <- function (h, a) {
        m <- length (dim (a))
        t0 <- 0
        for (i in 1:m) {
                s0 <- 0
                for (j in 1:m) {
                        hh <- h[[i]][[j]]
                        dh <- dim (hh)
                        ah <- aplSelect (a, lapply (1:m, function (x) 1:
                            dh[x]), drop = FALSE)
                        s0 <- s0 + sum (abs (hh) * ah)
                }
        t0 <- max (s0, t0)
        }
        return (t0)
}

tstar <- function(a) {
        a <- as.integer (a)
        s <- sum (a)
```

```
78          asum <- sum (s * ifelse (s == 0, 0, log (s)))
79          araw <- sum (a * ifelse (a == 0, 0, log (a)))
80          return (exp ( (araw - asum) / 2))
81  }
82
83  astar <- function (d) {
84          a <- array (0, d)
85          pd <- prod (d)
86          for (i in 1:pd) {
87                  id <- aplEncode (i, d)
88                  at <- tstar (id - 1)
89                  a <- aplSet (a, at, id)
90          }
91          return (a)
92  }
```

Department of Statistics, University of California, Los Angeles, CA 90095-1554

Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

E-mail address, Jan de Leeuw: deleeuw@stat.ucla.edu

E-mail address, Patrick Groenen: groenen@few.eur.nl

URL, Jan de Leeuw: http://gifi.stat.ucla.edu