Smacof at 50: A Manual
Part x: Constrained Smacof

Jan de Leeuw - University of California Los Angeles

Started May 04, 2024, Version of May 04, 2024

Abstract

TBD

Contents

Note: This is a working manuscript which will be expanded/updated frequently. All suggestions for improvement are welcome. All Rmd, tex, html, pdf, R, and C files are in the public domain. Attribution will be appreciated, but is not required. The various files can be found at https://github.com/deleeuw in the repositories smacofCode, smacofManual, and smacofExamples.

1 Introduction

2 General Theory

De Leeuw and Heiser (1980)

3 Linear Constraints

3.1 Subspace Constraints

3.2 PQN Constraints

$$Z := \begin{array}{cc} & \begin{array}{ccc} p & q & n \end{array} \\ n & \left[\begin{array}{c|c|c} X & YA & D \end{array}\right] \end{array}$$

4 Circular and Elliptical Constraints

De Leeuw (2007) De Leeuw (2005) De Leeuw and Mair (2009) Minimize

$$\omega(Y,\Lambda) := \text{tr }(\overline{X}-Y\Lambda)'V(\overline{X}-Y\Lambda)$$

over diagonal Λ and Y with diag $YY'=I$.

The optimal Λ for given Y is

$$\Lambda = \text{diag } Y'V\overline{X}/\text{diag } Y'VY$$

Let's look at all Y of the form $Y=\tilde{Y}+e_i(y-\tilde{y}_i)'$. Then ω is a function of y and we can write

$$\omega(y) := \omega(\tilde{Y},\Lambda)+v_{ii}(y-\tilde{y}_i)'\Lambda^2(y-\tilde{y}_i)-2e_i'V(X-\tilde{Y}\Lambda)\Lambda(y-\tilde{y}_i)$$

Let $H=V(X-\tilde{Y}\Lambda)\Lambda$. Then

$$\omega(y)\leq\eta(y):=\omega(\tilde{y}_i)+v_{ii}\lambda^2_{\text{max}}(y-\tilde{y}_i)'(y-\tilde{y}_i)-2h_i'(y-\tilde{y}_i)$$

Now suppose $\hat{y}$ minimizes η over $y'y=1$. Then

$$\omega(\hat{y})\leq\eta(\hat{y})\leq\eta(\tilde{y}_i)=\omega(\tilde{y}_i)$$

Minimizing η over $y'y=1$ can be done by maximizing

$$y'\{\lambda^2_{\text{max}}v_{ii}\tilde{y}_i+h_i\}$$

and thus $\hat{y}$ is the term in … curly brackets, normalized to unit length.

Let's check this in R. First generate some data.

```
#set.seed(12345)
v <- -as.matrix(dist(matrix(rnorm(12), 6, 2)))
diag(v) <- -rowSums(v)
x <- matrix(rnorm(12),6,2)
x <- v %*% x
y <- matrix(rnorm(12),6,2)
y <- y / sqrt(rowSums(y ^ 2))
res <- x - y
print(sum(res * (v %*% res)), digits = 10)
```

```
## [1] 42881.56515
```

```
lbd <- diag(crossprod(y, v %*% x)) / diag(crossprod(y, v %*% y))
mlbd <- max(lbd ^ 2)
lbd <- diag(lbd)
res <- x - y %*% lbd
print(sum(res * (v %*% res)), digits = 10)
```

```
## [1] 37269.35949
```

Now change the first row of Y.

```r
h <- v %*% res %*% lbd
g <- mlbd * v[1, 1] * y[1, ] + h[1, ]
y[1, ] <- g / sqrt(sum(g ^ 2))
res <- x - y %*% lbd
print(sum(res * (v %*% res)), digits = 10)
```

```
## [1] 37105.01009
```

So far, so good. We can improve the first row again.

```r
h <- v %*% res %*% lbd
g <- mlbd * v[1, 1] * y[1, ] + h[1, ]
y[1, ] <- g / sqrt(sum(g ^ 2))
res <- x - y %*% lbd
print(sum(res * (v %*% res)), digits = 10)
```

```
## [1] 37085.35919
```

Instead of continuing to iteratively improve the first row we'll make a loop over the rows of Y.

```r
for (i in 1:6) {
  h <- v %*% res %*% lbd
  g <- mlbd * v[i, i] * y[i,] + h[i,]
  y[i,] <- g / sqrt(sum(g ^ 2))
  res <- x - y %*% lbd
  print(sum(res * (v %*% res)), digits = 10)
}
```

```
## [1] 37081.1818
## [1] 32673.02149
## [1] 30777.27021
## [1] 30754.50963
## [1] 29558.44564
## [1] 29392.35169
```

We can do this again.

```r
for (i in 1:6) {
  h <- v %*% res %*% lbd
  g <- mlbd * v[i, i] * y[i,] + h[i,]
  y[i,] <- g / sqrt(sum(g ^ 2))
  res <- x - y %*% lbd
  print(sum(res * (v %*% res)), digits = 10)
}
```

```
## [1] 29320.30176
```

```
## [1] 26668.29102
## [1] 24209.88627
## [1] 24101.5816
## [1] 24097.29437
## [1] 24085.41324
```

And again.

```
for (i in 1:6) {
  h <- v %*% res %*% lbd
  g <- mlbd * v[i, i] * y[i,] + h[i,]
  y[i,] <- g / sqrt(sum(g ^ 2))
  res <- x - y %*% lbd
  print(sum(res * (v %*% res)), digits = 10)
}
```

```
## [1] 24081.3095
## [1] 23998.12667
## [1] 23153.05054
## [1] 23103.02434
## [1] 23083.08752
## [1] 23037.69143
```

These are still for the same Λ. We can compute a new Λ.

```
lbd <- diag(crossprod(y, v %*% x)) / diag(crossprod(y, v %*% y))
mlbd <- max(lbd ^ 2)
lbd <- diag(lbd)
res <- x - y %*% lbd
print(sum(res * (v %*% res)), digits = 10)
```

```
## [1] 4731.567776
```

And then start updating Y again.

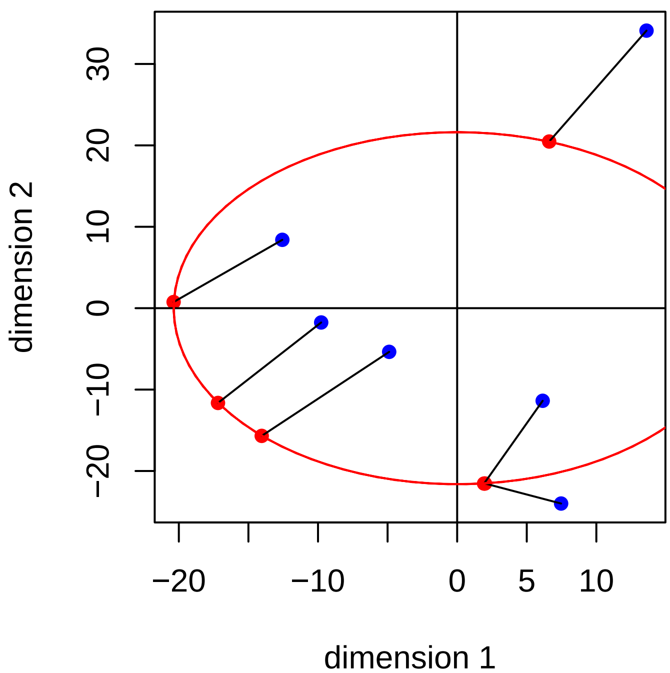
```
for (i in 1:6) {
  h <- v %*% res %*% lbd
  g <- mlbd * v[i, i] * y[i,] + h[i,]
  y[i,] <- g / sqrt(sum(g ^ 2))
  res <- x - y %*% lbd
  print(sum(res * (v %*% res)), digits = 10)
}
```

```
## [1] 4546.999526
## [1] 4438.990353
## [1] 4378.626778
## [1] 4239.591729
## [1] 3779.169994
```

```
## [1] 2072.594059
```

And so on. Let's look at the result so far. The blue points are X, the red points are Y, on an ellips with centered at the origin with the coordinate axes as minor and major axes.

```r
par(pty = "s")
z <- y %*% lbd
plot(rbind(x, z), type = "n", xlab = "dimension 1", ylab = "dimension 2")
points(x, col = "BLUE", pch = 16)
points(z, col = "RED", pch = 16)
abline(v = 0)
abline(h = 0)
for (i in 1:6) {
  lines(matrix(c(x[i, ], z[i, ]), 2, 2, byrow = TRUE))
}
sc <- seq(-2 * pi, 2 * pi, length = 100)
xc <- cbind(sin(sc), cos(sc)) %*% lbd
lines(xc, col = "RED")
```



Note that we have three nested infinite iterative processes here.

1. Alternating the updates of Λ and Y.
2. Cycling through the rows of Y.
3. Updating a single row of Y.

We could reduce this to two processes by not using majorization in process 3 but computing the exact update of a row. Go back to

$$\omega(y) := \omega(\tilde{Y}, \Lambda) + v_{ii}(y - \tilde{y}_i)'\Lambda^2(y - \tilde{y}_i) - 2h_i'(y - \tilde{y}_i)$$

Minimizing ω over $y'y = 1$ leads to the stationary equations

$$(\Lambda^2 - \mu I)y = g_i$$

with

$$g_i := \frac{v_{ii}\Lambda^2\tilde{y}_i + h_i}{v_{ii}}$$

and μ a Lagrange multiplier. This is one of the famous secular equations (see for instance Hager (2001)), which can be solved by finding a root of the equation

$$\phi(\mu) := \sum_{s=1}^{p} \frac{g_{is}^2}{(\lambda_s^2 - \mu)^2} = 1$$

Also, of course, if we are fitting a circle then Λ is fixed at the identity and matters simplify accordingly. Alternating the updates for Λ and Y is no longer necessary.

References

De Leeuw, J. 2005. "Fitting Ellipsoids by Least Squares." UCLA Department of Statistics. 2005. https://jansweb.netlify.app/publication/deleeuw-u-05-j/deleeuw-u-05-j.pdf.

———. 2007. "Quadratic Surface Embedding." UCLA Department of Statistics. 2007. https://jansweb.netlify.app/publication/deleeuw-u-07-h/deleeuw-u-07-h.pdf.

De Leeuw, J., and W. J. Heiser. 1980. "Multidimensional Scaling with Restrictions on the Configuration." In *Multivariate Analysis, Volume V*, edited by P. R. Krishnaiah, 501–22. Amsterdam, The Netherlands: North Holland Publishing Company.

De Leeuw, J., and P. Mair. 2009. "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software* 31 (3): 1–30. https://www.jstatsoft.org/article/view/v031i03.

Hager, William W. 2001. "Minimizing a Quadratic over a Sphere." *SIAM Journal on Optimization* 12: 188–208.