Smacof at 50: Part xx
Utilities

Jan de Leeuw

November 23, 2024

Table of 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 files can be found at https://github.com/deleeuw in the repositories smacofCode, smacofManual, and smacofExamples.

1 Simultaneous Iteration

The problem in this section is to maximize

$$\lambda(K) := \text{tr } K'AK \tag{1}$$

over all $n \times p$ orthonormal K. Suppose for the time being that A is positive semi-definite.

Define $k = \text{vec}(K)$ and

$$A_p := \underbrace{A \oplus \cdots \oplus A}_{p\text{times}} \tag{2}$$

Then $\lambda = k'A_pk$, which shows that λ i/s convex, and that consequently for all k and $\tilde{k}$ we have the minorization

$$k'A_pk \geq \tilde{k}'A_p\tilde{k} + 2\tilde{k}'A_p(k - \tilde{k}) = 2k'A_p\tilde{k} - \tilde{k}'A_p\tilde{k} \tag{3}$$

It follows that we increase λ by increasing tr $K'A\tilde{K}$ over $K'K = I$.

$$K^{(\nu+1)} = \underset{K'K=I}{\text{argmax}} \text{ tr } K'AK^{(\nu)} \tag{4}$$

Computing K^{(□+1)} is an orthoginal Procrustus problem (Gower and Dijksterhuis (2004)). Thus from the singular value decomposition $AK^{(\nu)} = P\Phi Q'$ we find $K^{(\nu+1)} = PQ'$.

There are two remaining elaborations of this result. First, we want to get rid of the assumption that A is positive semi-definite. We do this by adding a constant to the diagonal of A. To compute a suitable constant μ, use the fact that a diagonally dominant symmetric matrix with a non-negative diagonal is positive semidefinite. Thus if

$$\mu \geq \max_i \sum_{j \neq i} |a_{ij}| - a_{ii} \tag{5}$$

for all i then the matrix $\overline{A} := A + \mu I$ is positive semi-definite. We now use Equation 4 with the adjusted $\overline{A}$ and after convergence we subtract $p\mu$ from λ.

Second, instead of computing the singular value decomposition to update K in each iteration we can use the Q from the less expensive QR decomposition. The argument is the same as in Gifi (1990) (page 98-99, page 171). The QR update of K is a rotation of the Procrustus update of K, and it consequently gives the same value of λ from Equation 1.

2 Symmetric Matrix Approximation

The problem in this section is to minimize

$$\sigma(X) = \text{tr}\,(C - XX')^2 \tag{6}$$

over all $n \times p$ matrices X. Note there are no weights. The solution is well-known (Eckart and Young (1936), Keller (1962)). If $C = K\Lambda K'$ is the eigen-decomposition of C, with eigenvalues in non-increasing order on the diagonal of Λ, then define $\overline{\Lambda}$ with elements $\max(0, \lambda_s)$. Then use the p largest eigenvalues and corresponding vectors to compute $X = K_p\overline{\Lambda}_p$. Note that $\text{rank}(X)$ is less than p if C has fewer than p positive eigenvalues.

Our algorithm to minimize, or at least decrease, the loss function in Equation 6. Uses a combination of alternating least squares (De Leeuw (1994)) and majorization (De Leeuw (1994)) or MM (Lange (2016)). We first write X in the form $X = K\Lambda$, with $K'K = I$ and Λ diagonal. Then rewrite Equation 6 as

$$\sigma(K, \Lambda) = \text{tr}\,(C - K\Lambda^2K')^2 = \text{tr}\,C^2 - 2\text{tr}\,K'CK\Lambda^2 + \text{tr}\,\Lambda^4. \tag{7}$$

Again, for the time being, assume C is positive semi-definite. The minimum over Λ for given K has $\lambda_s^2 = k_s'Ck_s$. Thus

$$\min_{K'K=I}\min_{\Lambda}\sigma(K, \Lambda) = \text{tr}\,C^2 - \max_{K'K=I}\sum_{s=1}^p(k_s'Ck_s)^2 \tag{8}$$

The term depending on K is convex and can consequently be minorized by the linear function

$$\sum_{s=1}^p(k_s'Ck_s)^2 = \sum_{s=1}^p(\tilde{k}_s'C\tilde{k}_s)^2 + 2\sum_{s=1}^p(\tilde{k}_s'C\tilde{k}_s)\tilde{k}_s'C(k_s - \tilde{k}_s).$$

In iteration $\nu + 1$ we must maximize $\text{tr}\,K'CK^{(\nu)}\Lambda^{(\nu)}$ over $K'K = I$, where $\Lambda^{(\nu)} = \text{diag}\{K^{(\nu)}\}'CK^{(\nu)}$. Again, this is an orthogonal Procrustus problem.

3 Various Simple

```r
smacofEi <- function(i, n) {
  return(ifelse(i == 1:n, 1, 0))
}

smacofAij <- function(i, j, n) {
  ei <- ifelse(i == 1:n, 1, 0)
  ej <- ifelse(j == 1:n, 1, 0)
  return(outer(ei - ej, ei - ej))
}

smacofDoubleCenter <- function(a) {
  r <- apply(a, 1, mean)
  s <- mean(a)
  return(a - outer(r, r, "+") + s)
}

smacofCenter <- function(x) {
  return(apply(x, 2, function(x) x - mean(x)))
}

smacofTrace <- function(a) {
  return(sum(diag(a)))
}

smacofMakeDoubleCenter <- function(w) {
  v <- -w
  diag(v) <- -rowSums(v)
  return(v)
}

smacofDoubleCenterGeneralizedInverse <- function(v) {
  n <- nrow(v)
  return(solve(v + (1 / n)) - 1 / n)
}
```

4 Data formats

```r
smacofMakeData <-
  function(delta,
           weights = rep(1, length(delta)),
           winclude = FALSE,
           fname) {
    m <- length(delta)
    n <- as.integer((1 + sqrt(1 + 8 * m)) / 2)
    h <- fullIndex(n)
    g <- cbind(h$ii, h$jj, delta, weights)
    for (k in 1:m) {
      if ((g[k, 4] == 0) || is.na(g[k, 4]) || is.na(g[k, 3])) {
        continue
      } else {
        if (winclude) {
          cat(
            formatC(g[k, 1], digits = 3, format = "d"),
            formatC(g[k, 2], digits = 3, format = "d"),
            formatC(g[k, 3], digits = 6, format = "f"),
            formatC(g[k, 4], digits = 6, format = "f"),
            "\n",
            file = fname, append = TRUE
          )
        } else {
          cat(
            formatC(g[k, 1], digits = 3, format = "d"),
            formatC(g[k, 2], digits = 3, format = "d"),
            formatC(g[k, 3], digits = 6, format = "f"),
            "\n",
            file = fname, append = TRUE
          )
        }
      }
    }
  }


fullIndex <- function(n) {
```

```
  ii <- c()
  jj <- c()
  for (j in 1:(n - 1)) {
    for (i in (j + 1):n) {
      ii <- c(ii, i)
      jj <- c(jj, j)
    }
  }
  return(list(ii = ii, jj = jj))
}
```

5 I/O

```
smacofMatrixPrint <- function(x,
                              digits = 6,
                              width = 8,
                              format = "f",
                              flag = "+") {
  print(noquote(
    formatC(
      x,
      digits = digits,
      width = width,
      format = format,
      flag = flag
    )
  ))
}
```

6 Indices

sindex tindex mindex vindex

References

De Leeuw, J. 1994. "Block Relaxation Algorithms in Statistics." In *Information Systems and Data Analysis*, edited by H. H. Bock, W. Lenski, and M. M. Richter, 308–24. Berlin: Springer Verlag. https://jansweb.netlify.app/publication/deleeuw-c-94-c/deleeuw-c-94-c.pdf.

Eckart, C., and G. Young. 1936. "The Approximation of One Matrix by Another of Lower Rank." *Psychometrika* 1 (3): 211–18.

Gifi, A. 1990. *Nonlinear Multivariate Analysis*. New York, N.Y.: Wiley.

Gower, J. C., and G. B. Dijksterhuis. 2004. *Procrustus Problems*. Oxford University Press.

Keller, J. B. 1962. "Factorization of Matrices by Least Squares." *Biometrika* 49: 239–42.

Lange, K. 2016. *MM Optimization Algorithms*. SIAM.