ON INVERSE MULTIDIMENSIONAL SCALING

JAN DE LEEUW

ABSTRACT. We discuss *Inverse Multidimensional Scaling*, clarifying and extending some results of De Leeuw and Groenen [1997]. R code for all computations is also provided.

1. INTRODUCTION

We use standard notation for Euclidean Multidimensional Scaling (MDS), as explained, for example, in Borg and Groenen [2005] or De Leeuw and Mair [2009]. The *stress* loss function, first introduced by Kruskal [1964a,b], is

$$\sigma(X) \triangleq \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(\delta_{ij} - d_{ij}(X))^2. \tag{1}$$

Here the w_{ij} are known non-negative *weights*, and the δ_{ij} are known non-negative *dissimilarities*. We minimize stress over the $n \times p$ *configurations* X, the coordinates of n points in $\mathbb{R}^p$.

Following De Leeuw and Heiser [1980] we introduce some matrix notation for this problem. If the e_i are unit vectors, i.e. vectors with all elements zero except element i which is equal to one, then we let

$$A_{ij} \triangleq (e_i - e_j)(e_i - e_j)'$$

Date: Monday 21st May, 2012 — 21h 46min — Typeset in LUCIDA BRIGHT.
2000 *Mathematics Subject Classification.* 62H30,91C15.
Key words and phrases. Multidimensional Scaling.

For an $n\times p$ *configuration* X, i.e. the coordinates of n points in $\mathbb{R}^p$, we define squared distance

$$d_{ij}^2(x)=\textbf{tr}\ X'A_{ij}X.$$

Let

$$V\overset{\Delta}{=}\sum_{i=1}^n\sum_{j=1}^n w_{ij}A_{ij},$$

and, at points where $d_{ij}>0$ for all $i\neq j$,

$$B(X)\overset{\Delta}{=}\sum_{i=1}^n\sum_{j=1}^n w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}A_{ij}.$$

Then, assuming without loss of generality that

$$\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n w_{ij}\delta_{ij}^2=1,$$

we can write

$$\sigma(X)=1-\textbf{tr}\ X'B(X)X+\frac{1}{2}\textbf{tr}\ X'VX, \tag{2a}$$

and we find for the partial derivatives

$$\mathcal{D}\sigma(X)=(V-B(X))X. \tag{2b}$$

In MDS we have data W and Δ and we want to find X such that $\mathcal{D}\sigma(X)=0$. In *Inverse MDS* [De Leeuw and Groenen, 1997] we want to find W and Δ for a given X such that $\mathcal{D}\sigma(X)=0$. Thus we want to find all weights and dissimilarities such that configuration X is a stationary point.

In this note we study the a slightly simpler variant of the Inverse MDS problem, also studied in De Leeuw and Groenen [1997], in which we have both X and W and we want to find all Δ such that $\mathcal{D}\sigma(X)=0$. Our main contribution is some new code in R [R Development Core Team, 2012], which we then use to explore some aspects of Inverse MDS.

2. Analysis

Assume X is column-centered and of rank p. As shown in lemma A.1 the equation $\mathcal{D}\sigma(X) = 0$ can be written as

$$V - B(X) = K_\perp C K_\perp', \tag{3}$$

where $K_\perp$ is an orthonormal basis for the null-space of X, and C is a symmetric matrix. In fact, we choose $K_\perp$ orthogonal to $(1 \mid X)$, so that all columns of $K_\perp$ are centered. Thus $K_\perp$ is an $n \times q$ matrix with $q = n - p - 1$. Both sides of (3) are symmetric and doubly-centered.

From (3) the off-diagonal elements of the dissimilarity matrix $\Delta(C)$ are

$$\delta_{ij}(C) = d_{ij}(X)\left(1 - \frac{1}{w_{ij}} k_i' C k_j\right), \tag{4}$$

where the k_i and k_j are the rows of $K_\perp$. Because both sides of (3) are doubly-centered, equality of the off-diagonal elements implies equality of the diagonal elements as well. This shows that the set of solutions for Δ is an affine set. If $\Delta_1, \cdots, \Delta_m$, corresponding with symmetric matrices $C_1, \cdots, C_m$ are solutions, and $\sum_{j=1}^m \alpha_j = 1$, then $\sum_{j=1}^m \alpha_j \Delta_j$ is a solution as well. Note that the α_j are not required to be non-negative.

We can use unit vectors a basis for the space of symmetric matrices is given by $E_{st} = e_s e_t' + e_t e_s'$, for all $s < t$, and $E_{ss} = e_s e_s'$ for all s. By lemma A.2 the solutions to equation (4) are affine combinations of $D(X)$ and the matrices Δ_{st}, with $s \leq t$, and with elements

$$d_{ij}(X)\left[1 - \frac{1}{w_{ij}}(k_s k_t' + k_t k_s')\right]. \tag{5}$$

In equation (5) the k_s and k_t are now columns of $K_\perp$.

R code for computation of the spanning set of the affine subspace is in code segment 1.

Insert Code Segment 1 about here

The representaton we have chosen makes it easy to answer the slightly more general question of computing the dissimilarity matrices $\Delta(X)$ such that for some real λ the matrix λX is stationary. This is simply the subspace spanned by $D(X)$ and the Δ_{st}. Thus we can use the same spanning set but include the non-affine linear combinations.

3. Non-negativity Constraints

The affine combinations of the basis matrices given in (5) can have negative elements. If we require that $\delta_{ij} \geq 0$ for all i and j, then the solution set becomes a closed and bounded convex polyhedron.

The convex polyhedron is analyzed with the double description method, as described in Fukuda [1999], and implemented in R in the `rcdd` package [Geyer and Meeden, 2012].

Insert Code Segment 2 about here

4. Second Partials

To get a convenient formula for the second partials of stress we define the direct sums

$$\overline{\overline{A}}_{ij} = \underbrace{A_{ij} \oplus \cdots \oplus A_{ij}}_{p \text{ times}},$$

$$\overline{\overline{V}} = \underbrace{V \oplus \cdots \oplus V}_{p \text{ times}}.$$

Then, with $x \triangleq \mathbf{vec}(X)$,

$$\mathcal{D}^2\sigma(x) = \overline{\overline{V}} - \sum_{i=1}^n \sum_{j=1}^n w_{ij} \frac{\delta_{ij}}{d_{ij}(x)} \left[\overline{\overline{A}}_{ij} - \frac{\overline{\overline{A}}_{ij} x x' \overline{\overline{A}}_{ij}}{x' \overline{\overline{A}}_{ij} x} \right]. \tag{6}$$

There is code to compute the second partials in code segment 3. We also use `hessian()` from the `numDeriv` package [Gilbert, 2012] for comparison purposes.

INSERT CODE SEGMENT 3 ABOUT HERE

By the way, equation (6) and our previous expression for derivatives make it easy to implement Newton's method for MDS. Note, however, that $\mathcal{D}^2\sigma(x)$ is always singular with at least $p+1$ zero eigenvalues.

5. EXAMPLES

If $p = n-1$ then there is no null space and $\Delta = D(X)$ is the only solution for which X is stationary. In other words if Δ is not a Euclidean distance matrix then any stationary point X satisfies $\mathbf{rank}(X) < n-1$.

If $p = n-2$ then $K_\perp$ only has a single column. Thus all solutions for Δ are on the line in matrix space

$$\delta_{ij}(\alpha) \triangleq d_{ij}(X)\left\{1-\alpha\frac{1}{w_{ij}}k_ik_j\right\}. \tag{7}$$

Let us show in detail what happens in the example of four points in the corners of a square, with unit weights. Thus

$$X = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \qquad k = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \qquad d = \begin{bmatrix} 0 & & & \\ 2 & 0 & & \\ 2 & 2\sqrt{2} & 0 & \\ 2\sqrt{2} & 2 & 2 & 0 \end{bmatrix}.$$

We find

$$\Delta(\alpha) = \begin{bmatrix} 0 & & & \\ 2(1-\frac{1}{4}\alpha) & 0 & & \\ 2(1-\frac{1}{4}\alpha) & 2\sqrt{2}(1+\frac{1}{4}\alpha) & 0 & \\ 2\sqrt{2}(1+\frac{1}{4}\alpha) & 2(1-\frac{1}{4}\alpha) & 2(1-\frac{1}{4}\alpha) & 0 \end{bmatrix},$$

and

$$B(\alpha) = \begin{bmatrix} 3-\frac{1}{4}\alpha & -1+\frac{1}{4}\alpha & -1+\frac{1}{4}\alpha & -1-\frac{1}{4}\alpha \\ -1+\frac{1}{4}\alpha & 3-\frac{1}{4}\alpha & -1-\frac{1}{4}\alpha & -1+\frac{1}{4}\alpha \\ -1+\frac{1}{4}\alpha & -1-\frac{1}{4}\alpha & 3-\frac{1}{4}\alpha & -1+\frac{1}{4}\alpha \\ -1-\frac{1}{4}\alpha & -1+\frac{1}{4}\alpha & -1+\frac{1}{4}\alpha & 3-\frac{1}{4}\alpha \end{bmatrix}.$$

5.1. **Nonnegativity.** The dissimilarities are non-negative for $-4 \leq \alpha \leq +4$.

5.2. **Triangle Inequality.** By checking all $4 \times 3 = 12$ triangle inequalities we find that $\Delta(\alpha)$ is a metric for $-(12 - 8\sqrt{2}) \leq \alpha \leq 4$. Note that $12 - 8\sqrt{2} \approx 0.6862915$.

5.3. **Euclidean Distance.** If we square the dissimilarities in (7) we have

$$\delta_{ij}^2(\alpha) = (1 + \frac{1}{16}\alpha^2)d_{ij}^2(X) - 2\alpha d_{ij}^2(X)k_ik_j \tag{8}$$

because $k_i^2k_j^2 = \frac{1}{16}$ for all i, j. Define $u_{ij} \triangleq d_{ij}^2(X)k_ik_j$. The Torgerson transformation of a matrix C is defined as $\tau(C) = -\frac{1}{2}JCJ$, with $J = I - \frac{1}{n}ee'$ the centering operator. Apply this transformation to both sides of (8). This gives

$$\tau(\Delta(\alpha)) = (1 + \frac{1}{16}\alpha^2)XX' - 2\alpha\tau(U).$$

In this example

$$U = \begin{bmatrix} 0 & -1 & -1 & 2 \\ -1 & 0 & 2 & -1 \\ -1 & 2 & -1 & -1 \\ 2 & -1 & -1 & 0 \end{bmatrix},$$

which is already doubly-centered, so $\tau(U) = -\frac{1}{2}U$. Now define

$$K \triangleq \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}.$$

In this example K diagonalizes both XX' and $\tau(U)$, and thus

$$K'\tau(\Delta(\alpha))K = 4(1 + \frac{1}{16}\alpha^2)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} - 2\alpha\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Because $4(1+\frac{1}{16}\alpha^2)-2\alpha\geq 0$ for all α we see that the eigenvalues of $\tau(\Delta(\alpha))$ are non-negative for all $\alpha\geq 0$. Thus the $\Delta(\alpha)$ are non-negative Euclidean distances for $0\leq\alpha\leq 4$.

5.4. **Second Derivatives.** Formula (6) shows that the Hessian is linear in the sense that

$$\mathcal{D}^2\sigma(X,\Delta(\alpha))=\mathcal{D}^2\sigma(X,D(X))-\alpha\mathcal{D}^2\sigma(X,D(X)*kk').$$

This makes it easy to compute the interval where $\mathcal{D}^2\sigma(X,\Delta(\alpha))$ is positive semi-definite. It turns out that in our example X is a local minimum for $\Delta(\alpha)$ only if $\alpha\leq 0$.

6. Multiple Minima

As Trosset and Mathar [1997] show, it is not easy to rigorously find non-global minima of stress. An exception is $p=1$, i.e. unidimensional scaling. Exact computations, for example by De Leeuw [2005], show hundreds of local minima, even in small examples.

Inverse MDS can be used to study some aspects of the local minimum problem. Suppose both X and Y are stationary values of the same MDS problem defined by W and Δ. Then Δ must be an affine combination of the basis U generated by X and an affine combination of the basis T generated by Y. Thus we must find vectors α and β that add up to one and that satisfy $U\alpha=T\beta$.

As indicated before, if we only care about the solutions up to a proportionality factor, then constraining α and β to add up to one is no longer necessary.

Let us use the example of five points equally spaced on a line and five points equally spaced on a circle.

$$X = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & 1 \\ \sin(\frac{2}{5}\pi) & \cos(\frac{2}{5}\pi) \\ \sin(\frac{4}{5}\pi) & \cos(\frac{4}{5}\pi) \\ \sin(\frac{6}{5}\pi) & \cos(\frac{6}{5}\pi) \\ \sin(\frac{8}{5}\pi) & \cos(\frac{8}{5}\pi) \end{bmatrix}$$

We look for the Δ for which μX is a stationary value for some μ and for which λY is stationary for some λ. U, generated by X, has seven dimensions, while V, generated by Y, has four dimensions. They are both computed by the function `inverseMDS()`.

The 10×11 matrix $\begin{bmatrix} U \mid -V \end{bmatrix}$ turns out to have rank nine, which means there are two linearly independent solutions f $U\alpha = T\beta$. And there is a two-dimensional subspace of the space of hollow symmetric matrices with the property that for each matrix in the subspace there exist μ and λ such that μX and λY are stationary.

INSERT CODE SEGMENT 4 ABOUT HERE

Within the subspace we can easily find the two extreme rays of the cone of non-negative dissimilarity matrices. They turn out to be

```
       [,1]   [,2]
 [1,] 0.0467 0.0989
 [2,] 0.0989 0.0467
 [3,] 0.0233 0.1223
 [4,] 0.0756 0.0700
 [5,] 0.0000 0.1456
 [6,] 0.1456 0.0000
 [7,] 0.0233 0.1223
 [8,] 0.0000 0.1456
 [9,] 0.0989 0.0467
[10,] 0.0467 0.0989
```

Note that the two rows, say f_1 and f_2, add up to $0.1456\cdots$. If we look at the convex combinations $\lambda f_1 + (1-\lambda) f_2$, it turns out that these are Euclidean for $0.2282\cdots \leq \lambda \leq 0.6016\cdots$. For $\lambda = .5$ we find the matrix with all dissimilarities equal.

For the same convex combinations we find that X, suitably normalized, is a local minimum of $\lambda f_1 + (1-\lambda) f_2$ for all $0 \leq \lambda \leq 1$. This corresponds with the fact that in one-dimensional MDS we only have local minima, no saddle-points, and no local maxima (except for $X = 0$). Normalized Y is a local minimum for all $0.1712\cdots \leq \lambda \leq 1$.

Clearly the techniques in this section can be generalized to deal with solving for subspaces or cones of dissimilarity matrices that have more than two given solutions to the stationary equations, although it is clear that if we choose $X_1, \cdots, X_m$ completely generally we will rapidly run out of dimensions for the intersection of the dissimilarity spaces.

Appendix A. Lemmas

Lemma A.1. *Suppose K is $n \times p$ with $K'K = I$ and $K_\perp$ is $n \times (n-p)$ such that $K'K_\perp = 0$ and $K_\perp'K_\perp = I$. Then a real symmetric T satisfies $TK = 0$ if and only if $T = K_\perp C K_\perp'$, with C real symmetric.*

Proof. If $T = K_\perp C K_\perp'$ then $TK = 0$. For necessity write T in the form

$$T = \begin{bmatrix} K & K_\perp \end{bmatrix} \begin{bmatrix} A & B \\ B' & C \end{bmatrix} \begin{bmatrix} K' \\ K_\perp' \end{bmatrix}$$

Then we must have

$$TK = \begin{bmatrix} K & K_\perp \end{bmatrix} \begin{bmatrix} A \\ B' \end{bmatrix} = 0,$$

which implies $A = 0$ and $B = 0$. □

Lemma A.2. *If $\mathcal{L}$ is a subspace with basis $x_1, \cdots, x_m$ and $\mathcal{M} \overset{\Delta}{=} x_0 + L$ is the affine subspace obtained by translating $\mathcal{L}$ by x_0. Then each $z \in \mathcal{M}$ can be written as an affine linear combination of $x_0 \in \mathcal{M}$ and $x_0 + x_1 \in \mathcal{M}, \cdots, x_0 + x_m \in \mathcal{M}$.*

Proof. Suppose $y = \sum_{j=i}^m \alpha_j x_j$. Then

$$z = x_0 + y = (1 - \sum_{j=i}^m \alpha_j) x_0 + \sum_{j=i}^m \alpha_j (x_j + x_0).$$

□

APPENDIX B. CODE

Code Segment 1 Basis Computation

```
inverseMDS <- function (x) {
    n <- nrow (x)
    m <- ncol (x)
    x <- apply (x, 2, function (y) y - mean (y))
    nm <- n - (m + 1)
    kk <- cbind (1, x, matrix (rnorm (n * nm), n, nm))
    kperp <- as.matrix (qr.Q ( qr (kk))[, -(1 : (m + 1))])
    dd <- Euclid (x)
    k <- 1
    base <- matrix (0, n * (n - 1) / 2, nm * (nm + 1) / 2)
    for (i in 1 : nm) {
        for (j in 1 : i) {
            oo <- outer (kperp[, i], kperp[, j])
            if (j != i) {
                oo <- oo + t(oo)
            }
            base[, k] <- lower_triangle (dd * (1 - oo))
            k <- k + 1
            print (c(i,j,k))
        }
    }
    return (base = cbind (lower_triangle (dd), base))
}
```

Code Segment 2 Vertex Computation

```
require("rcdd")

inversePlus <- function (base, affine = TRUE) {
    if (affine) {
        hrep <- makeH (a1 = d2q(-base), b1 = d2q (rep (0,
            nrow (base))),
            a2 = d2q (rep (1, ncol(base))), b2 = d2q (1))
        } else {
            hrep <- makeH (a1 = d2q(-base), b1 = d2q (rep
                (0, nrow (base))))
        }
    vrep <- scdd (hrep)
    hrep <- q2d (hrep)
    vrep <- q2d (vrep $ output)
    pr <- tcrossprod (hrep[, -c(1,2)], vrep[, -c(1,2)])[-1,
         ]
    return (list (base = base, hrep = hrep, vrep =  vrep,
        pr =  pr))
}
```

Code Segment 3 Second Partials

```
require ("numDeriv")

second_partials_stress <- function (x, delta, w = nonDiag (
    nrow (x))) {
    n <- nrow (x)
    p <- ncol (x)
    d <- Euclid (x)
    fac <- (w * delta) / (d + diag (n))
    dd <- d * d
    v <- vmat (w)
    deri <- direct_sum (repList (v, p))
    xx <- as.vector (x)
    for (i in 1 : (n - 1)) {
        for (j in (i + 1) : n) {
            aa <- direct_sum (repList (aijn (i, j, n), p))
            ax <- drop (aa %*% xx)
            deri <- deri - fac[i, j] * (aa - outer (ax, ax)
                / dd [i, j])
        }
    }
    return (deri)
}

second_partials_numerical <- function (x, delta, w =
    nonDiag (nrow (x))) {
    stress <- function (x, delta, w) {
        n <- nrow (delta)
        p <- length (x) / n
        d <- Euclid (matrix (x, n, p))
        res <- delta - d
        return (sum (w * res * res) / 2)
    }
    return (hessian (stress, x, delta = delta, w = w))
}
```

Code Segment 4 Two Stationary Points

```
twoPoints <- function (x, y) {
    dx <- inverseMDS (x)
    dy <- inverseMDS (y)
    mx <- ncol (dx)
    my <- ncol (dy)
    ez <- eigen (crossprod (cbind (dx, -dy)))
    iz <- which (abs (ez $ values) < 1e-10)
    ax <- ez $ vectors[1 : mx, iz]
    ay <- ez $ vectors[mx + (1 : my), iz]
    fx <- dx %*% ax
    fy <- dy %*% ay
    return (list (fx = fx, fy = fy))
}
```

Code Segment 5 Check Functions

```
checkMinima <- function (x, base, w = nonDiag (nrow (x))) {
    m <- ncol (base)
    n <- nrow (x)
    p <- ncol (x)
    np <- n * p
    eval <- matrix (0, m, np)
    for (i in 1 : m) {
        delta <- fill_symmetric (base[, i])
        eval[i, ] <- eigen (second_partials_stress (x,
            delta, w)) $ values
    }
    return (eval)
}

checkStationary <- function (x, base, w = nonDiag (nrow (x)
    ), scale = FALSE) {
    n <- nrow (x)
    m <- ncol (base)
    d <- Euclid (x)
    eps <- rep (0, m)
    lbd <- rep (0, m)
    for (i in 1 : m) {
        delta <- fill_symmetric (base [, i])
        b <- bmat (x, delta, w)
        v <- vmat (w)
        vx <- v %*% x
        bx <- bmat %*% x
        lbd[i] <- sum (x * bx) / sum (x * vx)
        if (scale) {
            eps[i] <- max (abs (lbd[i] * vx - bx))
        } else {
            eps[i] <- max (abs (vx - bx))
        }
    }
    return (list(eps = eps, lbd = lbd))
}
```

Code Segment 6 Utility Functions I

```
lower_triangle <- function (x) {
    n <- nrow (x)
    return (x[outer (1:n, 1:n, ">")])
}

fill_symmetric <- function (x) {
    m <- length (x)
    n <- 0.5 + sqrt (0.25 + 2 * m)
    d <- matrix (0, n, n)
    d[outer (1:n, 1:n, ">")] <- x
    return (d + t(d))
}

Euclid <- function (x) {
    c <- tcrossprod (x)
    d <- diag (c)
    return (sqrt (outer (d, d, "+") - 2 * c))
}

circular <- function (n) {
    x <- seq (0, 2 * pi, length =  n + 1)
    z <- matrix (0, n + 1, 2)
    z[, 1] <- sin (x)
    z[, 2] <- cos (x)
    return (z[-1, ])
}

direct_sum <- function (x) {
    n <- length (x)
    nr <- sapply (x, nrow)
    nc <- sapply (x, ncol)
    s <- matrix (0, sum (nr), sum (nc))
    k <- 0
    l <- 0
    for (j in 1 : n) {
        s[k + (1 : nr[j]), l + (1 : nc[j])] <- x[[j]]
        k <- k + nr[j]
        l <- l + nc[j]
        }
    return (s)
}
```

Code Segment 7 Utility Functions II

```
1
2  aijn <- function (i, j, n) {
3      a <- matrix (0, n, n)
4      a[i, i] <- 1
5      a[j, j] <- 1
6      a[i, j] <- -1
7      a[j, i] <- -1
8      return (a)
9  }
10
11 repList <- function(x, n) {
12     z <- list()
13     for (i in 1 : n)
14         z <- c (z, list (x))
15     return (z)
16 }
17
18 nonDiag <- function (n) {
19     return (matrix (1, n, n) - diag (n))
20 }
21
22 bmat <- function (x, delta, w = nonDiag (nrow (x))) {
23     d <- Euclid (x)
24     bmat <- -(w * delta) / (d + diag (n))
25     diag (bmat) <- -rowSums (bmat)
26     return (bmat)
27 }
28
29 vmat <- function (w = nonDiag (nrow (x))) {
30     vmat <- -w
31     diag (vmat) <- -rowSums (vmat)
32     return (vmat)
33 }
34
35 torgerson <- function (delta) {
36     n <- nrow (delta)
37     dd <- delta * delta
38     sd <- rowSums (dd) / n
39     ed <- sum (dd) / (n * n)
40     cc <- -(dd - outer (sd, sd, "+") + ed) / 2
41     return (eigen (cc))
42 }
```

REFERENCES

I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, Second edition, 2005.

J. De Leeuw. Unidimensional Scaling. In B.S. Everitt and D.C Howell, editors, *The Encyclopedia of Statistics in Behavioral Science*, volume 4, pages 2095–2097. Wiley, New York, N.Y., 2005. URL http://www.stat.ucla.edu/~deleeuw/janspubs/2005/chapters/deleeuw_C_05h.pdf.

J. De Leeuw and P.J.F. Groenen. Inverse Multidimensional Scaling. *Journal of Classification*, 14(3–21), 1997.

J. De Leeuw and W. J. Heiser. Multidimensional Scaling with Restrictions on the Configuration. In P.R. Krishnaiah, editor, *Multivariate Analysis, Volume V*, pages 501–522, Amsterdam, The Netherlands, 1980. North Holland Publishing Company.

J. De Leeuw and P. Mair. Multidimensional Scaling Using Majorization: SMACOF in R. *Journal of Statistical Software*, 31(3):1–30, 2009. URL http://www.jstatsoft.org/v31/i03.

K. Fukuda. *cdd/cdd+ Reference Manual*. Institute for Operations Research ETH, Zurich, Switzerland, 1999. URL ftp://ftp.ifor.math.ethz.ch/pub/fukuda/cdd/cddman/cddman.html. (cdd ver. 0.61 and cdd+ ver. 0.76, March 17, 1999).

C.J. Geyer and G.D. Meeden. *rcdd: rcdd (Computational Geometry)*, 2012. URL http://CRAN.R-project.org/package=rcdd. R package version 1.1-7.

P. Gilbert. *numDeriv: Accurate Numerical Derivatives*, 2012. URL http://R-Forge.R-project.org/projects/optimizer/. R package version 2012.3-1/r586.

J. B. Kruskal. Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, 29:1–27, 1964a.

J.B. Kruskal. Nonmetric Multidimensional Scaling: a Numerical Method. *Psychometrika*, 29:115–129, 1964b.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing,

Vienna, Austria, 2012. URL http://www.R-project.org. ISBN 3-900051-07-0.

M.W. Trosset and R. Mathar. On the Existence on Nonglobal Minimizers of the STRESS Criterion for Metric Multidimensional Scaling. In *Proceedings of the Statistical Computing Section*, pages 158–162, Alexandria, VA, 1997. American Statistical Association.

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

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

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