Notes on the C Version of Smacof

Jan de Leeuw - University of California Los Angeles

Started October 10 2023, Version of December 03, 2023

Abstract

TBD

Contents

Note: This is a working paper 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/mdsStruct.

1 Introduction

The loss function in (metric, least squares, Euclidean, symmetric) Multidimensional Scaling (MDS) is

$$\sigma(X) := \frac{1}{2}\mathop{\sum\sum}_{1\leq j<i\leq n} w_{ij}(\delta_{ij}-d_{ij}(X))^2.$$

This assumes symmetry and it uses all elements below the diagonal of both W, Δ, and $D(X)$. For missing data we set $w_{ij}=0$.

2 Example

Here is a small input example.

3 Unweighted, full matrix

3.1 Unnormalized

$W=E-I$ and $V=(I-E)+(n-1)I=nI-E=nJ$. Thus $V^+=n^{-1}J$ and $V^+B=n^{-1}B$.

3.2 Normalized

After normalization $W = \frac{1}{n(n-1)}(E-I)$ and $\sum\sum w_{ij}{\delta_{ij}}^2 = 1$ or $\sum\sum {\delta_{ij}}^2 = n(n-1)$. Also $V = \frac{1}{n(n-1)}(nI-E) = \frac{1}{(n-1)}J$, which means $V^+ = (n-1)J$. For

$$B(X) = \sum_{i<j} w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}A_{ij} = \frac{1}{n(n-1)}\sum_{i<j}\frac{\delta_{ij}}{d_{ij}(X)}A_{ij}$$

and thus

$$V^+B(X) = \frac{1}{n}\sum_{i<j}\frac{\delta_{ij}}{d_{ij}(X)}A_{ij}$$

4 Unweighting

$$\begin{aligned}\sigma(d) &= \sum_{k=1}^{K} w_k(\delta_k - d_k)^2 && (1)\\ &= \sum_{k=1}^{K} w_k(\delta_k - \overline{d}_k)^2 - 2\sum_{k=1}^{K} w_k(\delta_k - \overline{d}_k)(d_k - \overline{d}_k) + \sum_{k=1}^{K} w_k(d_k - \overline{d}_k)^2. && (2)\end{aligned}$$

Now suppose $w_k \leq w_\star$ and define

$$r_k = \frac{w_k}{w_\star}(\delta_k - \overline{d}_k)$$

$$\begin{aligned}\sigma(d) &\leq \sigma(\overline{d}) + w_\star\left\{\sum_{k=1}^{K}(d_k - \overline{d}_k)^2 - \sum_{k=1}^{K} r_k(d_k - \overline{d}_k)\right\}\\ &= \sigma(\overline{d}) + w_\star\left\{\sum_{k=1}^{K}(d_k - (\overline{d}_k + r_k))^2 - \sum_{k=1}^{K} r_k^2\right\}.\end{aligned}$$

Note

$$\delta_k(X) := \frac{w_k}{w_\star}\delta_k + (1 - \frac{w_k}{w_\star})d_k(X)$$

So given X computed the adjusted dissimilarities $\Delta(X)$ and perform one or more unweighted smacof steps to compute the update X^+. Of course if all w_k are the same then $\Delta(X) = \Delta$ and we compute a regular unweighted smacof. It is of some interest to study how many smacof steps to make before computing a new $\Delta(X)$.

5 Thoughts on ALS

In nonmetric MDS we minimize

$$\sigma(\Delta, X) = \sum_{k=1}^{K} w_k(\delta_k - d_k(X))^2$$

over the configurations X and the transformed dissimilarities Δ, where we assume $\Delta \in \mathfrak{D}$, with $\mathfrak{D}$ the intersection of a convex cone and a sphere.

5.1 The Single-Step approach

Kruskal (1964a), Kruskal (1964b) defines

$$\sigma_\star(X) := \min_{\Delta\in\mathfrak{D}} \sigma(\Delta, X)$$

and

$$\Delta(X) = \operatorname*{argmin}_{\Delta\in\mathfrak{D}} \sigma(\Delta, X)$$

Thus

$$\sigma_\star(X) = \sigma(\Delta(X), X)$$

which is now a function of X only. Under some conditions, which are usually true in MDS,

$$\mathcal{D}\sigma_\star(X) = \mathcal{D}_2\sigma(\Delta(X), X)$$

where $\mathcal{D}_2\sigma(\Delta(X), X)$ are the partials of σ with respect to X. Thus the partials of $\sigma_\star$ can be computed by evaluating the partials of σ with repesct to X at $(X, \Delta(X))$. This has created much confusion in the past. We can now solve the problem of minimizing $\sigma_\star$, which is a function of X alone.

I think Guttman calls this the *single step approach*. A variation of Kruskal's single-step approach defines

$$\sigma_G(X) = \sum_{k=1}^K w_k(\delta_k^\#(X) - d_k(X))^2$$

where the $\delta_k^\#(X)$ are *Guttman's rank images*, i.e. the permutation of the $d_k(X)$ that makes it monotone with the δ_k (Guttman (1968)). Or, alternatively, we define

$$\sigma_S(X) := \sum_{k=1}^K w_k(\delta_k^\%(X) - d_k(X))^2$$

where the $\delta_k^\%(X)$ are *Shepard's rank images*, i.e. the permutation of the δ_k that makes it monotone with the $d_k(X)$ (Shepard (1962a), Shepard (1962b)).

The Shepard and Guttman alternatives are computationally more intricate and more complicated than the Kruskal *monotone regression* approach, mostly because of problems with uniqueness and differentiation, but they are obviously both single step approaches.

5.2 The Two-step Approach

The *two-step approach* or *alternating least squares* approach alternates minimization of $\sigma(\Delta, X)$ over X for our current best estimate of Δ with minimization of $\sigma(\Delta, X)$ over $\Delta \in \mathfrak{D}$ for our current best value of X. Thus an update looks like

$$\Delta^{(k)} = \operatorname*{argmin}_{\Delta} \sigma(\Delta, X^{(k)}), \tag{3}$$

$$X^{(k+1)} = \operatorname*{argmin}_{X} \sigma(\Delta^{(k)}, X). \tag{4}$$

This approach to MDS was in the air since the early (unsuccessful) attempts around 1968 of Young and De Leeuw to combine Torgerson's classic metric MDS method with Kruskal's monotone regression transformation.

As formulated, however, there are some problems with the ALS algorithm. Step (**??**) is easy to carry out, using monotone regression. Step (**??**) means solving a metric scaling problem, which is an iterative proces that requires an infinite number of iterations. Thus, what is usually implemented, is to combine step (**??**) with one of more iterations of a convergent iterative procedure for metric MDS, such as smacof. If we take only one of these *inner iterations* the algorithm becomes indistinguishable from Kruskal's single step method. This has also created much confusion in the past.

It is somewhat worrisome that in the ALS approach we solve the first subproblem (**??**) exactly, while we take only a single step towards the solution for given Δ in the second subproblem (**??**). If we have an infinite iterative procedure to compute the optial $\Delta \in \mathfrak{D}$ for given X, then a more balanced approach is to take several inner iterations in the first step and several innewr iterations in the second step. How many of each, nobody knows.

6 Memory

This is mostly for my own education.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int n = 10, p = 4;
  char *a = NULL;
  double **x = (double **)calloc((size_t)n, sizeof(double *));
  double y[10][4] = {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0},
                     {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0},
                     {0, 0, 0, 0}, {0, 0, 0, 0}};
  for (int i = 0; i < n; i++) {
    x[i] = (double *)calloc((size_t)p, sizeof(double));
  }
  printf("x[i][j] (dynamic allocation on heap)\n");
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < p; j++) {
      printf("%ld ", (uintptr_t)&x[i][j]);
    }
    printf("\n");
  }
  printf("\n\n");
  printf("x[i] and &x[i] and sizeof x[i]\n");
  for (int i = 0; i < n; i++) {
    printf("%ld %ld %ld\n", (uintptr_t)x[i], (uintptr_t)&x[i], sizeof x[i]);
  }
  printf("\n\n");
  printf("x\n");
  printf("%ld\n", (uintptr_t)x);
  printf("\n\n");
  printf("******************************************\n");
  printf("\n\n");
  printf("y[i][j] (static allocation on stack)\n");
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < p; j++) {
      printf("%ld ", (uintptr_t)&y[i][j]);
    }
    printf("\n");
  }
  printf("\n\n");
  printf("y[i] and &y[i] and sizeof y[i]\n");
  for (int i = 0; i < n; i++) {
    printf("%ld %ld %ld\n", (uintptr_t)y[i], (uintptr_t)&y[i], sizeof y[i]);
  }
  printf("\n\n");
  printf("y\n");
  printf("%ld\n", (uintptr_t)y);
  printf("\n\n");
  for (int i = 0; i < n; i++) {
    free(x[i]);
  }
  free(x);
  return EXIT_SUCCESS;
}
```

7 Hessian

There are several ways to think of the Hessian. The simplest one (perhaps) is as an $np \times np$ symmetric matrix (corresponding to column-major R vector of length $\frac{1}{2}np(np+1)$). This is what we would use for a straightforward version of Newton-Raphson.

It is more elegant, however, to think of H as a symmetric super-matrix of order p, with as elements $n \times n$ matrices. And, for some purposes, such as the pseudo-confidence ellipsoids in De Leeuw (2017b), as a super-matrix of order n with as elements $p \times p$ matrices. Both the super-matrix interpretations lead to four-dimensional arrays, the first a $p \times p \times n \times n$ array, the second an $n \times n \times p \times p$ array. The different interpretations lead to different ways to store the Hessian in memory, and to different ways to retrieve its elements.

8 Appendix: Code

We use *qsort* to sort the rows of the input data frame by increasing delta. The sorting is done in C, the R version is a wrapper around the compiled C code. The C code also contains a main which analyzes the same small example as we have used in the text. Compile with "clang -o runner -O2 smacofSort.c" and then start "runner" in the shell. The C code uses the .C() interface in R, which can be improved using .Call(), but probably in this case with little gain.

9 smacofMaximumSum

Maximize

$$\sum\sum_{1\leq j<i\leq n} w_{ij}\delta_{ij}^2 d_{ij}^2(X)$$

over X with tr $(X'X)^2 = 1$. This gives $BX = X\Lambda$, with

$$B = \sum\sum_{1\leq j<i\leq n} w_{ij}\delta_{ij}^2 A_{ij}$$

There is also a non-metric version. Maximize over tr $(X'X)^2$

$$\sum\sum_{1\leq j<i\leq n}\sum\sum_{1\leq k<l\leq n} w_{ij,kl}\text{sign}(\delta_{ij}-\delta_{kl})(d_{ij}^2(X)-d_{kl}^2(X))$$

which simplifies to

$$2\sum\sum_{1\leq j<i\leq n} d_{ij}^2(X)\sum\sum_{1\leq k<l\leq n} w_{ij,kl}\text{sign}(\delta_{ij}-\delta_{kl})$$

Simplifies more $w_{ij,kl} = w_{ij}$ or $_{ij,kl} = w_{ij}w_{kl}$

Also note

$$\rho(X) = \sum\sum_{1\leq j<i\leq n} w_{ij}\delta_{ij}d_{ij}(X) = \sum\sum_{1\leq j<i\leq n}\frac{w_{ij}}{\delta_{ij}d_{ij}(X)}\delta_{ij}^2 d_{ij}^2(X) \approx \sum\sum_{1\leq j<i\leq n}\frac{w_{ij}}{\delta_{ij}^2}\delta_{ij}^2 d_{ij}^2(X) = \eta^2(X)$$

10 smacofElegant

11 smacofAdjustDiagonal

12 smacofImpute

13 smacofHildreth

Hildreth (1957)

Consider the QP problem of minimizing $f(x) = \frac{1}{2}(x-y)'W(x-y)$ over all $x \in \mathbb{R}^n$ satisfying $Ax \geq 0$, where A is $m \times n$. Wlg we can assume $a_j'Wa_j = 1$. The Lagrangian is

$$\mathcal{L}(x,\lambda) = \frac{1}{2}(x-y)'W(x-y) - \lambda'Ax.$$

$$\max_{\lambda\geq 0}\mathcal{L}(x,\lambda)=\begin{cases}\frac12(x-y)'W(x-y) & \text{if } Ax\geq 0,\\ +\infty & \text{otherwise.}\end{cases}$$

and thus

$$\min_{x\in\mathbb{R}^n}\max_{\lambda\geq 0}\mathcal{L}(x,\lambda)=\min_{Ax\geq 0}\frac{1}{2}(x-y)'W(x-y).$$

By duality

$$\min_{x\in\mathbb{R}^n}\max_{\lambda\geq 0}\mathcal{L}(x,\lambda)=\max_{\lambda\geq 0}\min_{x\in\mathbb{R}^n}\mathcal{L}(x,\lambda).$$

The inner minimum over x is attained for

$$x=y+W^{-1}A'\lambda,$$

and is equal to

$$\min_{x\in\mathbb{R}^n}\mathcal{L}(x,\lambda)=-\frac{1}{2}\lambda'AW^{-1}A'\lambda-\lambda'Ay.$$

Thus

$$\max_{\lambda\geq 0}\min_{x\in\mathbb{R}^n}\mathcal{L}(x,\lambda)=-\frac{1}{2}\min_{\lambda\geq 0}\{(Wy+A'\lambda)'W^{-1}(Wy+A'\lambda)-y'Wy\}$$

We minimize $h(\lambda)=(Wy+A'\lambda)'W^{-1}(Wy+A'\lambda)$ with coordinate descent. Let $\lambda_j(\epsilon)=\lambda+\epsilon e_j$. Then

$$h(\lambda_j(\epsilon))=(Wy+A'\lambda+\epsilon a_j)'W^{-1}()=\epsilon^2a_j'W^{-1}a_j+2\epsilon a_j'W^{-1}(y+W^{-1}A'\lambda)+$$

which must be minimized over $\epsilon\geq-\lambda_j$. So the minimum is attained at

$$\epsilon=-\frac{a_j'W^{-1}x}{a_j'W^{-1}a_j}$$

with $x=y+W^{-1}A'\lambda$ (cf …), provided … satisfies .. Otherwise $\epsilon=-\lambda_j$. Now update both λ and x, and go to the next j.

14 smacofDykstra

15 smacofJacobi

taken from De Leeuw (2017a)

partial jacobi

16 smacofIndividualDifferenceModels.c

$$
\begin{aligned}
X_k &= X, && (5)\\
X_k &= X\Lambda_k \text{ with } \Lambda_k \text{ diagonal}, && (6)\\
X_k &= XC_k, && (7)\\
X_k &= X\Lambda_k Y' \text{ with } \Lambda_k \text{ diagonal}, && (8)\\
X_k &= XC_k Y', && (9)\\
X_k &= X\Lambda_k Y_k' \text{ with } \Lambda_k \text{ diagonal} && (10)\\
X_k &= XC_k Y' \text{ with } C_k = \sum_{s=1}^{r} z_{ks} H_s. && (11)
\end{aligned}
$$

17 smacofSort

18 Code

18.1 smacofSort.R

18.2 smacofSort.c

```
```

References

De Leeuw, J. 2017a. "Jacobi Eigen in R/C with Lower Triangular Column-wise Compact Storage." 2017. https://jansweb.netlify.app/publication/deleeuw-e-17-o/deleeuw-e-17-o.pdf.

———. 2017b. "Pseudo Confidence Regions for MDS." 2017.

Guttman, L. 1968. "A General Nonmetric Technique for Fitting the Smallest Coordinate Space for a Configuration of Points." *Psychometrika* 33: 469–506.

Hildreth, C. 1957. "A Quadratic Programming Procedure." *Naval Research Logistic Quarterly* 14 (79–85).

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

———. 1964b. "Nonmetric Multidimensional Scaling: a Numerical Method." *Psychometrika* 29: 115–29.

Shepard, R. N. 1962a. "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I." *Psychometrika* 27: 125–40.

———. 1962b. "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II." *Psychometrika* 27: 219–46.