Accelerated Least Squares Multidimensional Scaling

Jan de Leeuw - University of California Los Angeles

Started July 23 2024, Version of August 06, 2024

Abstract

We discuss a simple accelerations of MDS smacof iterations, and compare them with recent boosted difference-of-convex algortithms.

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 Introduction

In this paper we study minimization of the multidimensional scaling (MDS) loss function

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

over all $n \times p$ *configuration* matrices X. Here $W = \{w_{ij}\}$ and $\Delta = \{\delta_{ij}\}$ are known non-negative, symmetric, and hollow matrices of *weights* and *dissimilarities* and $D(X) = \{d_{ij}(X)\}$ is the matrix of *Euclidean distances* between the rows of X. The symbol $:=$ is used for definitions.

Throughout we assume, without loss of generality, that W is irreducible, that X is column-centered, and that Δ is normalized by

$$\frac{1}{2}\mathop{\sum\sum}_{1\leq i<j\leq n} w_{ij}\delta_{ij}^2 = 1. \tag{2}$$

1.1 Notation

It is convenient to have some matrix notation for our MDS problem. We use the symmetric matrices A_{ij}, of order n, which have $+1$ at (i,i) and (j,j), -1 at (i,j) and (j,i), and zeroes everywhere else. Using unit vectors e_i and e_j we can write

$$A_{ij} := (e_i - e_j)(e_i - e_j)' \tag{3}$$

Following De Leeuw (1977) and De Leeuw (1988) we define

$$\rho(X) := \mathop{\sum\sum}_{1\leq i<j\leq n} w_{ij}\delta_{ij}d_{ij}(X) = \text{tr } X'B(X)X, \tag{4}$$

where

$$B(X) := \mathop{\sum\sum}_{1\leq i<j\leq n} w_{ij}\frac{\delta_{ij}}{r_{ij}(X)}A_{ij}, \tag{5}$$

with

$$r_{ij}(X) = \begin{cases} \frac{1}{d_{ij}(X)} & \text{if } d_{ij}(X) > 0, \\ 0 & \text{if } d_{ij}(X) = 0. \end{cases} \tag{6}$$

Also define

$$\eta^2(X) := \mathop{\sum\sum}_{1\leq i<j\leq n} w_{ij}d_{ij}^2(X) = \text{tr } X'VX, \tag{7}$$

where

$$V := \mathop{\sum\sum}_{1\leq i<j\leq n} w_{ij}A_{ij}. \tag{8}$$

Thus

$$\sigma(X) = 1 - \rho(X) + \frac{1}{2}\eta^2(X) = 1 - \text{tr } X'B(X)X + \frac{1}{2}\text{tr } X'VX. \tag{9}$$

Both $B(X)$ and V are positive semi-definite and doubly-centered. Because of the irreducibility of W the matrix V has rank $n-1$, with only the constant vectors in its null space.

Both ρ and η are homogeneous convex functions, with η being a norm on the space of column-centered configurations. If the equations $d_{ij}(X) = 0$, i.e. $x_i = x_j$, for all (i,j) for which $w_{ij}\delta_{ij} > 0$ only have the trivial solution $X = 0$ then ρ is a norm as well. Note that ρ is not differentiable if $d_{ij}(X) = 0$ for an (i,j) for which $w_{ij}\delta_{ij} > 0$.
Because

$$|d_{ij}(X) - d_{ij}(Y)|^2 \leq \text{tr } (X-Y)'A_{ij}(X-Y) \leq 2p\|X-Y\|^2 \tag{10}$$

we see that ρ, although not differentiable, is globally Lipschitz.

1.2 The Guttman Transform

The *Guttman transform* of a configuration X, so named by De Leeuw and Heiser (1980) to honor the contribution of Guttman (1968), is defined as the set-valued map

$$\Phi(X) = V^+\partial\rho(X), \tag{11}$$

with V^+ the Moore-Penrose inverse of V and $\partial\rho(X)$ the subdifferential of ρ at X, i.e. the set of all Z such that $\rho(Y) \geq \rho(X) + \text{tr } Z'(Y-X)$ for all Y. Because of homogeneity we have that $Z \in \partial\rho(X)$ if and only if tr $Z'X = \rho(X)$ and $\rho(Y) \geq$ tr $Z'Y$ for all Y. Because ρ is continuous, its subdifferential is compact and convex.

From Moreau-Rockafellar theorem (Rockafellar (1970)) the subdifferential of ρ is the Minkovski linear combination

$$\partial\rho(X) = \mathop{\sum\sum}_{1\leq i<j\leq n} w_{ij}\delta_{ij}\partial d_{ij}(X) \tag{12}$$

For completeness we give an explicit formula for the subdifferential of the distance function between rows i and j of an $n \times p$ matrix.

$$\partial d_{ij}(X) = \begin{cases}\left\{\frac{1}{d_{ij}}(e_i - e_j)(x_i - x_j)'\right\} & \text{if } d_{ij}(X) > 0,\\ \left\{Z \mid Z = (e_i - e_j)z' \text{ with } z'z \leq 1\right\} & \text{if } d_{ij}(X) = 0.\end{cases} \tag{13}$$

Thus of $d_{ij}(X) > 0$, i.e. if d_{ij} is differentiable at X, then $\partial d_{ij}(X)$ is a singleton, containing only the gradient at X.

It follows that

$$\partial\rho(X) = B(X)X + Z \tag{14}$$

with

$$Z \in \sum\sum\{w_{ij}\delta_{ij}\partial d_{ij}(X) \mid d_{ij}(X) = 0\}. \tag{15}$$

stationary point

This little excursion into convex analysis is rarely needed in practice. It is shown by De Leeuw (1984) that a necessary condition for a local minimum at X is that $d_{ij}(X) > 0$ for all (i,j) for which $w_{ij}\delta_{ij} > 0$. At those points σ is differentiable, and thus the subdifferential is a singleton, containing only the gradient.

Cauchy Schwartz

If $Z \in \partial\rho(X)$ then $\rho(Y) \geq$ tr $Z'Y$.

Using the Guttman transform we can derive the equality

$$\sigma(X) = 1 + \eta^2(X - \Phi(X)) - \eta^2(\Phi(X)) \tag{16}$$

for all X and the inequality

$$\sigma(X) \leq 1 + \eta^2(X - \Phi(Y)) - \eta^2(\Phi(Y)) \tag{17}$$

for all X and Y.

Taken together (16) and (17) imply the *sandwich inequality*

$$\sigma(\Phi(Y)) \leq 1 - \eta^2(\Phi(Y)) \leq 1 + \eta^2(Y - \Phi(Y)) - \eta^2(\Phi(Y)) = \sigma(Y). \tag{18}$$

If Y is not a fixed point of Φ then the second inequality in the chain is strict and thus $\sigma(\Phi(Y)) < \sigma(Y)$. It also follows from (18) that $\eta^2(\Phi(Y)) \leq 1$.

2 One-point Methods

2.1 Basic Iteration

The basic smacof algorithm generates the iterative sequence

$$X^{(k+1)} = \Phi(X^{(k)}),$$

where it is understood that we stop if $X^{(k)}$ is a fixed point. If $X^{(k)}$ is not a fixed point it follows from (18) that $\sigma(X^{(k+1)}) < \sigma(X^{(k)})$.

De Leeuw (1988) derives some additional results. Using up-arrows and down-arrows to indicate monotone convergence we have

- $\rho(X^{(k)}) \uparrow \rho_\infty$,
- $\eta^2(X^{(k)}) \uparrow \eta^2_\infty = \rho_\infty$,
- $\sigma(X^{(k)}) \downarrow \sigma_\infty = 1 - \rho_\infty$,

and, last but not least, the sequence $\{X^{(k)}\}$ is *asymptotically regular*, i.e.

$$\eta^2(X^{(k+1)} - X^{(k)}) \rightarrow 0$$

Since the subdifferential is a upper semi-continuous (closed) map, and all iterates are in the compact set $\eta^2(X) \leq 1$, and Φ is strictly monotonic (decreases stress at non-fixed points), it follows from Meyer (1976) that all accumulation points are fixed points and have the same function value σ_∞. Moreover, from Ostrowski (1966), either the sequence converges or the accumulation points form a continuum.

In order to prove actual convergence, additional conditions are needed. Meyer (1976) proves convergence if the number of fixed points with function value σ_∞ is finite, or if the sequence has an accumulation point that is an isolated fixed point. Both these conditions are not met in our case, because of rotational indeterminacy. If X_∞ is a fixed point, then the continuum of rotations of X_∞ are all fixed points.

De Leeuw (1988) argues that the results so far are sufficient from a practical point of view. If we define an ϵ-fixed-point as any X with $\eta(X - \Phi(X)) < \eta$ then smacof produces such an ϵ-fixed-point in a finite number of steps.

In two very recent papers Ram and Sabach (2024 (in press)) and Robini, Wang, and Zhu (2024) use the powerful Kurdyka-Łojasiewicz (KL) machinery (ref) to prove actual convergence of smacof to a fixed point. We shall use the more classical approach based on the differentiability of the Guttman transform.

2.2 Majorization and Difference-of-Convex Function Algorithms

The original derivation of the smacof algorithm (De Leeuw (1977), De Leeuw and Heiser (1977)) used the framework of maximizing the ratio of norms discussed by Robert (1967). Later derivations (De Leeuw and Heiser (1980), De Leeuw (1988)) used the fact that (17) defines a majorization scheme for stress. Convergence then follows from the general *majorization principle* (these days mostly known as the *MM principle*). A recent overview of the MM approach is Lange (2016).

It was also realized early on that the smacof algorithm was a special case of the the difference-of-convex functions algorithm (DCA), introduced by Pham Dinh Tao around 1980. Pham Dinh also started his work in the context of ratio's of norms, using Robert's fundamental ideas. Around 1985 he generalized his approach to minimizing DC functions of the form $h = f - g$, with both f and g convex. The basic idea is to use the subgradient inequality $g(x) \geq g(y) + z'(x-y)$, with $z \in \partial g(x)$, to construct the majorization $h(x) := f(x) - g(y) - z'(x-y)$. Now h is obviously convex in x. The DC algorithm then chooses the successor of y as the minimizer of this convex majorizer over x. In smacof the role of f is played by η^2 and the role of g by ρ. The convex subproblem in each step is quadratic, and has the closed form solution provided by the Guttman transform. DCA is applied to MDS in Le Thi and Tao (2001), and extensive recent surveys of the DC/DCA approach are Le Thi and Tao (2018) and Le Thi and Tao (2024).

2.3 Rate of Convergence

In order to study the asymptotic rate of convergence of smacof, we have to study the Jacobian of the Guttman transform and its eigenvalues (Ortega and Rheinboldt (1970), chapter 10). Thus we assume we are in the neighborhood of a local minimum, where the Guttman transform is (infinitely many times) differentiable. The derivative is

$$\mathcal{D}\Phi_X(H) = V^+ \sum w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ A_{ij}H - \frac{\text{tr } X'A_{ij}H}{\text{tr } X'A_{ij}X} A_{ij}X \right\}. \tag{19}$$

Thus $\mathcal{D}\Phi_X(X) = 0$ for all X and the Jacobian has at least one zero eigenvalue. If we think of (19) as a map on the space of all $n \times p$ matrices, then there are an additional p zero eigenvalues corresponding with translational invariance. If we define (19) on the column-centered matrices, then these eigenvalues disappear.

If S is anti-symmetric and $H = XS$ then tr $X'A_{ij}H = 0$ and thus $\mathcal{D}\Phi_X(XS) = \Phi(X)S$. If in addition X is a fixed point then $\mathcal{D}\Phi_X(XS) = V^+B(X)XS = XS$, which means $\mathcal{D}\Phi_X$ has $\frac{1}{2}p(p-1)$ eigenvalues equal to one. They quantify the rotational indeterminacy of the MDS problem and the smacof iterations.

Since $\Phi(X) = V^+\mathcal{D}\rho(X)$ the Jacobian of the Guttman transform has a simple relationship with the second derivatives of ρ, which are

$$\mathcal{D}^2\rho_X(H,H) = \sum w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ \text{tr } H'A_{ij}H - \frac{\{\text{tr } H'A_{ij}X\}^2}{d_{ij}^2(X)} \right\} = \text{tr } H'V\mathcal{D}\Phi_X(H). \tag{20}$$

It follows that $0 \lesssim \mathcal{D}^2\rho_X \lesssim B(X)$ in the Loewner sense. Of course $\mathcal{D}^2\sigma_X = V - \mathcal{D}^2\rho_X$ At a local minimum $\mathcal{D}^2\sigma_X \gtrsim 0$, and consequently $\mathcal{D}\Phi_X \lesssim I$. Thus all eigenvalues of the Jacobian are between zero and one.

We apply basic iterations to the color-circle example from Ekman (1954), which has $n = 14$ points. The fit is very good and convergence is rapid. We stop when $\sigma(X^{(k)}) - \sigma(X^{(k+1)}) < 1e-15$. The results for the final iteration are

```
## itel  57 sold 2.1114112739 snew 2.1114112739 chng 0.0000000000 labd 0.7669812392
```

In this output chng is $\eta(X^{(k)} - X^{(k+1)}$, and labd is an estimate of the asymptotic convergence ratio, the chng divided by the chng of the previous iteration. We always start with the classical Torgerson-Gower solution. To fifteen decimals stress is 2.1114112739076

We compute the Jacobian using the numDeriv package (Gilbert and Varadhan (2019)). Its eigenvalues are

```
##  [1]   +1.0000000000   +0.7669965027   +0.7480939418   +0.7185926293
##  [5]   +0.7007452300   +0.6920114811   +0.6859492532   +0.6593334523
##  [9]   +0.6541779410   +0.6477573342   +0.6237683212   +0.6178713315
## [13]   +0.5735285948   +0.5483330654   +0.5260355535   +0.5112510731
## [17]   +0.5064703617   +0.5059294793   +0.4919752629   +0.4827646549
## [21]   +0.4782034983   +0.4757907684   +0.4682965897   +0.4619226490
## [25]   +0.4559704883   +0.0000000000   +0.0000000000   -0.0000000000
```

Note that the second largest and forst non-trivial eigenvalue is equal to labd from the final iteration.

2.4 Rotated Basic Iteration

As De Leeuw (1988) mentions, we cannot apply the basic Ostrowski point-of-attraction theorem 10.1.3 and the linear convergence theorem 10.1.4 from Ortega and Rheinboldt (1970), because there are these $\frac{1}{2}p(p-1$ eigenvalues equal to one.

One way around this problem is to rotate each update to orthogonality. Thus the update formula is $\Phi_o(X) = \Pi(\Phi(X))$ by applying the QR decomposition or the singular value decomposition to $\Phi(X)$. Somes simple R code which can be used for this purpose is

```r
x <- x %*% qr.Q(qr(t(x[1:ndim, ])))
x <- x %*% svd(x)$v
```

Now clearly this modified algorithm generates the same sequence of function values as basic smacof. Moreover $\Theta_o^n(X) = \Pi(\Phi^n(X))$, which means that we can find any term of the orthogonal sequence by orthogonalizing the corresponding term in the basic sequence. Thus, in actual computation, there is no need to orthogonalize, we may as well run the basic seuence and orthogonalize after convergence.

Theoretically, however, orthogonalization gives the same convergence rate as the basic sequence, but the Jacobian of Φ_o at a local minimum does not have the unit eigenvalues any more. They are replaced by zeroes, reflecting the fact that we are iterating on the nonlinear manifold or orthogonal column-centered matrices. It is now sufficient for linear convergence to assume that the largest eigenvalue of the Jacobian at the solution is strictly less than one, or alternatively assume that one of the accumulation points is an isolated local minimum.

We guvce the results by applying the two orthogonalization methods to the Ekman sequence.

```
## itel  54 sold 2.1114112739 snew 2.1114112739 chng 0.0000000000 labd 0.7669843896
```

```
##  [1]   +0.7669964894   +0.7480939420   +0.7185926297   +0.7007452335
##  [5]   +0.6920114817   +0.6859492534   +0.6593334543   +0.6541779412
##  [9]   +0.6477573345   +0.6237683217   +0.6178713316   +0.5735285959
## [13]   +0.5483330651   +0.5260355534   +0.5112510730   +0.5064703618
## [17]   +0.5059294793   +0.4919752632   +0.4827646574   +0.4782035029
## [21]   +0.4757907649   +0.4682965885   +0.4619226493   +0.4559704884
## [25]   -0.0000000000   -0.0000000000   +0.0000000000   +0.0000000000
```

```
## itel  56 sold 2.1114112739 snew 2.1114112739 chng 0.0000000000 labd 0.7669940008
```

```
##  [1]   +0.7669964993   +0.7480939419   +0.7185926294   +0.7007452309
##  [5]   +0.6920114813   +0.6859492533   +0.6593334529   +0.6541779410
##  [9]   +0.6477573343   +0.6237683213   +0.6178713317   +0.5735285948
## [13]   +0.5483330653   +0.5260355535   +0.5112510731   +0.5064703617
## [17]   +0.5059294792   +0.4919752629   +0.4827646550   +0.4782034999
## [21]   +0.4757907672   +0.4682965894   +0.4619226495   +0.4559704888
## [25]   -0.0000000006   -0.0000000000   -0.0000000000   +0.0000000000
```

3 Two Point Iteration

3.1 Basic

De Leeuw and Heiser (1980) suggested the "relaxed" update

$$\Psi(X) = 2\Phi(X) - X$$

The reasoning here is two-fold. The smacof inequality (17) says

$$\sigma(X) \leq 1 + \eta^2(X - \Phi(Y)) - \eta^2(\Phi(Y))$$

If $X = \alpha\Phi(Y) + (1-\alpha)Y$ then this becomes

$$\sigma(\alpha\Phi(Y) + (1-\alpha)Y) \leq 1 + (1-\alpha)^2\eta^2(Y - \Phi(Y)) - \eta^2(\Phi(Y))$$

If $(1-\alpha)^2 \leq 1$ then

$$1 + (1-\alpha)^2\eta^2(Y - \Phi(Y)) - \eta^2(\Phi(Y)) \leq 1 + \eta^2(Y - \Phi(Y)) - \eta^2(\Phi(Y)) = \sigma(Y)$$

Thus updating with $X^{(k+1)} = \alpha\Phi(X^{(k)}) + (1-\alpha)X^{(k)}$ is a stricly monotone algorithm as long as $0 \leq \alpha \leq 2$.

rate

It turns out that applying the relaxed update has some unintended consequences, which basically imply that it should never be used without some additional computation. Let's take a look at the Ekman results.

```
## itel  23 sold 3.9946270666 snew 3.9946270666 chng 3.7664315853 labd 1.0000000000
```

We see that $\eta^2(X^{(k+1)} - X^{(k)})$ does not converge to zero, and that σ_k converges to a value which does not even correspond to a local minimum of σ.

```
##  [1]   -1.0000000000   +1.0000000000   -1.0000000000   -1.0000000000
##  [5]   +0.5339929781   +0.4961878839   +0.4371852597   +0.4014904677
##  [9]   +0.3840229636   +0.3718985068   +0.3186669088   +0.3083558824
## [13]   +0.2955146690   +0.2475366434   +0.2357426633   +0.1470571918
## [17]   +0.0966661301   -0.0880590224   -0.0761547015   -0.0634068231
## [21]   +0.0520711068   -0.0484184707   -0.0435929937   -0.0344706856
## [25]   +0.0225021460   -0.0160494738   +0.0129407235   +0.0118589584
```

A more thorough analysis of the results show that the method produces a sequence $X^{(k)}$ with two subseqences. If $\overline{X}$ is a fixed point of Φ then there is a $\tau > 0$ such that the subsequence with k even converges to $\tau\overline{X}$ while the subsequence with k odd converges to $(2-\tau)\overline{X}$.

what goes wrong ? not strictly monotone at $\tau\overline{X}$

stress is 2.1114112739076

```
## itel  18 sold 3.9946270666 snew 3.9946270666 chng 0.0000000000 labd 0.2737973829
```

stress is 2.1114112739076

```
##  [1]   +1.0000000000   -1.0000000000   -1.0000000000   -0.9999999999
##  [5]   +0.5339930271   +0.4961878833   +0.4371852579   +0.4014904541
##  [9]   +0.3840229612   +0.3718985063   +0.3186669016   +0.3083558816
## [13]   +0.2955146680   +0.2475366415   +0.2357426629   +0.1470571877
## [17]   +0.0966661315   -0.0880590242   -0.0761547024   -0.0634068192
## [21]   +0.0520711070   -0.0484184575   -0.0435930109   -0.0344706938
## [25]   +0.0225021463   -0.0160494743   +0.0129407234   +0.0118589585
```

3.2 Doubling

3.3 Scaling

3.4 Switching

$$\Phi(\Psi(X))$$

$$\max_s |\lambda_s(2\lambda_s - 1)|$$

# Benchmarking

Mersmann (2023)

```
## Warning in microbenchmark(smacofAccelerate(delta, ndim = 2, opt = 1, halt = 2,
## : less accurate nanosecond times to avoid potential integer overflows
```

```
## Unit: milliseconds
##                                                                   expr      min
##  smacofAccelerate(delta, ndim = 2, opt = 1, halt = 2, verbose = FALSE) 3.188775
##  smacofAccelerate(delta, ndim = 2, opt = 2, halt = 2, verbose = FALSE) 3.151875
##  smacofAccelerate(delta, ndim = 2, opt = 3, halt = 2, verbose = FALSE) 3.847276
##  smacofAccelerate(delta, ndim = 2, opt = 4, halt = 2, verbose = FALSE) 1.421019
##  smacofAccelerate(delta, ndim = 2, opt = 5, halt = 2, verbose = FALSE) 1.564027
##  smacofAccelerate(delta, ndim = 2, opt = 6, halt = 2, verbose = FALSE) 1.639467
##  smacofAccelerate(delta, ndim = 2, opt = 7, halt = 2, verbose = FALSE) 1.526307
##        lq     mean   median       uq      max neval
##  3.304825 3.602914 3.358987 3.476656 6.070009   100
##  3.241993 3.521317 3.294924 3.374894 5.645618   100
##  3.939567 4.335232 4.029644 4.215682 6.634046   100
##  1.466795 1.601369 1.498222 1.541600 3.919846   100
##  1.618536 1.762409 1.650004 1.712775 3.995286   100
##  1.692829 1.827641 1.720585 1.775444 6.503174   100
##  1.578336 1.718087 1.618106 1.683132 6.439214   100
```

De Gruijter (1967)

```
## Unit: milliseconds
##                                                                   expr      min
##  smacofAccelerate(delta, ndim = 2, opt = 1, halt = 2, verbose = FALSE) 43.06710
##  smacofAccelerate(delta, ndim = 2, opt = 2, halt = 2, verbose = FALSE) 44.09382
##  smacofAccelerate(delta, ndim = 2, opt = 3, halt = 2, verbose = FALSE) 54.69146
##  smacofAccelerate(delta, ndim = 2, opt = 4, halt = 2, verbose = FALSE) 20.16495
##  smacofAccelerate(delta, ndim = 2, opt = 5, halt = 2, verbose = FALSE) 14.30547
##  smacofAccelerate(delta, ndim = 2, opt = 6, halt = 2, verbose = FALSE) 22.89670
##  smacofAccelerate(delta, ndim = 2, opt = 7, halt = 2, verbose = FALSE) 20.67150
##        lq     mean   median       uq      max neval
##  44.02676 45.24634 44.87503 45.65168 66.89129   100
##  45.18950 46.15677 46.23545 46.68055 50.64677   100
##  56.24936 56.93894 56.64076 57.10158 74.41721   100
##  20.85358 22.01609 22.03133 22.24851 40.76454   100
##  14.63911 15.66496 14.95206 16.28350 34.73516   100
##  24.80289 25.22137 25.04815 25.37377 44.09050   100
##  22.44541 22.52770 22.71785 22.93573 24.76105   100
```

```
## Warning in microbenchmark(smacofAccelerate(delta, ndim = 2, opt = 1, halt = 2,
## : less accurate nanosecond times to avoid potential integer overflows
```

```
## Unit: milliseconds
##                                                                     expr      min
##  smacofAccelerate(delta, ndim = 2, opt = 1, halt = 2, verbose = FALSE) 46.56747
##  smacofAccelerate(delta, ndim = 2, opt = 2, halt = 2, verbose = FALSE) 48.06479
##  smacofAccelerate(delta, ndim = 2, opt = 3, halt = 2, verbose = FALSE) 54.45587
##  smacofAccelerate(delta, ndim = 2, opt = 4, halt = 2, verbose = FALSE) 18.63007
##  smacofAccelerate(delta, ndim = 2, opt = 5, halt = 2, verbose = FALSE) 14.92945
##  smacofAccelerate(delta, ndim = 2, opt = 6, halt = 2, verbose = FALSE) 25.26014
##  smacofAccelerate(delta, ndim = 2, opt = 7, halt = 2, verbose = FALSE) 24.38951
##        lq     mean   median       uq      max neval
##  47.51435 50.23612 47.91568 48.57957 93.45950   100
##  48.68481 50.85570 49.08241 49.91586 71.12811   100
##  55.36041 58.24445 55.79360 57.31894 78.80885   100
##  19.09593 21.63385 19.50159 22.16286 43.51994   100
##  15.35686 17.05628 15.63457 18.36261 38.77669   100
##  28.05058 29.18501 28.82068 29.54122 49.39791   100
##  25.12099 28.54070 27.63353 28.10698 49.94571   100
```

4 Code

```r
library(MASS)
library(microbenchmark)
library(numDeriv)

smacofAccelerate <- function(delta,
                             ndim = 2,
                             wgth = 1 - diag(nrow(delta)),
                             xold = smacofTorgerson(delta, ndim),
                             opt = 1,
                             halt = 1,
                             itmax = 1000,
                             epsx = 1e-10,
                             epsf = 1e-15,
                             verbose = 1) {
  v <- -wgth
  diag(v) <- -rowSums(v)
  vinv <- ginv(v)
  n <- nrow(xold)
  cold <- Inf
  itel <- 1
  last <- FALSE
  repeat {
    xold <- apply(xold, 2, function(x)
      x - mean(x))
    dold <- as.matrix(dist(xold))
    sold <- sum(wgth * (delta - dold) ^ 2)
    bold <- -wgth * delta / (dold + diag(n))
    diag(bold) <- -rowSums(bold)
    xbar <- vinv %*% bold %*% xold
    if (opt == 1) {
      xnew <- xbar
      dnew <- as.matrix(dist(xnew))
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    if (opt == 2) {
      lbd <- sqrt(sum(xbar[1, ] ^ 2))
      cs <- xbar[1, 2] / lbd
      sn <- xbar[1, 1] / lbd
      rot <- matrix(c(sn, cs, -cs, sn), 2, 2)
      xnew <- xbar %*% rot
      dnew <- as.matrix(dist(xnew))
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    if (opt == 3) {
      xnew <- xbar %*% svd(xbar)$v
      dnew <- as.matrix(dist(xnew))
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    if (opt == 4) {
      xnew <- 2 * xbar - xold
      dnew <- as.matrix(dist(xnew))
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    if (opt == 5) {
      xaux <- 2 * xbar - xold
      daux <- as.matrix(dist(xaux))
      baux <- -wgth * delta / (daux + diag(n))
      diag(baux) <- -rowSums(baux)
      xbaz <- vinv %*% baux %*% xaux
      xnew <- 2 * xbaz - xaux
      dnew <- as.matrix(dist(xnew))
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    if (opt == 6) {
      xaux <- 2 * xbar - xold
      daux <- as.matrix(dist(xaux))
      alpa <- sum(wgth * daux * delta) / sum(wgth * daux ^ 2)
      xnew <- alpa * xaux
      dnew <- alpa * daux
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    if (opt == 7) {
      xaux <- 2 * xbar - xold
      daux <- as.matrix(dist(xaux))
      baux <- -wgth * delta / (daux + diag(n))
      diag(baux) <- -rowSums(baux)
      xnew <- vinv %*% baux %*% xaux
      dnew <- as.matrix(dist(xnew))
      snew <- sum(wgth * (delta - dnew) ^ 2)
      cnew <- sum((xold - xnew) * (v %*% (xold - xnew)))
    }
    labd <- sqrt(cnew / cold)
    if (verbose == 2) {
      cat(
        "itel",
        formatC(itel, digits = 2, format = "d"),
        "sold",
        formatC(sold, digits = 10, format = "f"),
        "snew",
        formatC(snew, digits = 10, format = "f"),
        "chng",
        formatC(cnew, digits =  10, format = "f"),
        "labd",
        formatC(labd, digits =  10, format = "f"),
        "\n"
      )
    }
    if (halt == 1) {
      converge <- cnew < epsx
    } else {
      converge <- (sold - snew) < epsf
    }
    if ((itel == itmax) || converge) {
      break
    }
    itel <- itel + 1
    sold <- snew
    xold <- xnew
    cold <- cnew
  }
  if (verbose == 1) {
    cat(
      "itel",
      formatC(itel, digits = 2, format = "d"),
      "sold",
      formatC(sold, digits = 10, format = "f"),
      "snew",
      formatC(snew, digits = 10, format = "f"),
      "chng",
      formatC(cnew, digits =  10, format = "f"),
      "labd",
      formatC(labd, digits =  10, format = "f"),
      "\n"
    )
  }
  if (opt == 4) {
    xaux <- (xnew + xold) / 2
    xnew <- (xnew + xold) / 2
    dnew <- as.matrix(dist(xnew))
    snew <- sum(wgth * (delta - dnew) ^ 2)
  }
  if (opt == 5) {
    bold <- -wgth * delta / (dnew + diag(n))
    diag(bold) <- -rowSums(bold)
    xnew <- vinv %*% bold %*% xnew
    dnew <- as.matrix(dist(xnew))
    snew <- sum(wgth * (delta - dnew) ^ 2)
  }
  return(list(
    x = xnew,
    s = snew,
    d = dnew,
    itel = itel,
    chng = cnew,
    labd = labd,
    wgth = wgth,
    delta = delta
  ))
}

smacofCompare <- function(delta, ndim = 2) {
  n <- nrow(delta)
  wgth <- 1 - diag(n)
  xold <- smacofTorgerson(delta, ndim)
  return(
    microbenchmark(
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 1,
        halt = 2,
        verbose = FALSE
      ),
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 2,
        halt = 2,
        verbose = FALSE
      ),
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 3,
        halt = 2,
        verbose = FALSE
      ),
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 4,
        halt = 2,
        verbose = FALSE
      ),
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 5,
        halt = 2,
        verbose = FALSE
      ),
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 6,
        halt = 2,
        verbose = FALSE
      ),
      smacofAccelerate(
        delta,
        ndim = 2,
        opt = 7,
        halt = 2,
        verbose = FALSE
      )
    )
  )
}

smacofTorgerson <- function(delta, ndim) {
  n <- nrow(delta)
  dd <- delta ^ 2
  rd <- rowSums(dd) / n
  sd <- mean(dd)
  cc <- -.5 * (dd - outer(rd, rd, "+") + sd)
  ee <- eigen(cc)
  x <- ee$vectors[, 1:ndim] %*% diag(sqrt(ee$values[1:ndim]))
  return(x)
}

numFunc <- function(x, nobj, ndim, wgth, delta, opt = 1) {
  xx <- matrix(x, nobj, ndim)
  dd <- as.matrix(dist(xx))
  vv <- -wgth
  diag(vv) <- -rowSums(vv)
  vinv <- ginv(vv)
  bb <- -wgth * delta / (dd + diag(nobj))
  diag(bb) <- -rowSums(bb)
  xaux <- vinv %*% bb %*% xx
  if (opt == 1) {
    yy <- xaux
  }
  if (opt == 2) {
    lbd <- sqrt(sum(xaux[1, ] ^ 2))
    cs <- xaux[1, 2] / lbd
    sn <- xaux[1, 1] / lbd
    rot <- matrix(c(sn, cs, -cs, sn), 2, 2)
    yy <- xaux %*% rot
  }
  if (opt == 3) {
    yy <- xaux %*% svd(xaux)$v
  }
  if (opt == 4) {
    yy <- 2 * xaux - xx
  }
  return(as.vector(yy))
}

numHess <- function(x, delta, wgth = 1 - diag(nrow(x)), opt = 1) {
  nobj <- nrow(x)
  ndim <- ncol(x)
  x <- as.vector(x)
  h <- jacobian(numFunc, x, nobj = nobj, ndim = ndim, wgth = wgth, delta = delta, opt =
  return(h)
}
```

References

De Gruijter, D. N. M. 1967. "The Cognitive Structure of Dutch Political Parties in 1966." Report E019-67. Psychological Institute, University of Leiden.

De Leeuw, J. 1977. "Applications of Convex Analysis to Multidimensional Scaling." In *Recent Developments in Statistics*, edited by J. R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, 133–45. Amsterdam, The Netherlands: North Holland Publishing Company.

———. 1984. "Differentiability of Kruskal's Stress at a Local Minimum." *Psychometrika* 49: 111–13.

———. 1988. "Convergence of the Majorization Method for Multidimensional Scaling." *Journal of Classification* 5: 163–80.

De Leeuw, J., and W. J. Heiser. 1977. "Convergence of Correction Matrix Algorithms for Multidimensional Scaling." In *Geometric Representations of Relational Data*, edited by J. C. Lingoes, 735–53. Ann Arbor, Michigan: Mathesis Press.

———. 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.

Ekman, G. 1954. "Dimensions of Color Vision." *Journal of Psychology* 38: 467–74.

Gilbert, P., and R. Varadhan. 2019. *numDeriv: Accurate Numerical Derivatives*. https://CRAN.R-project.org/package=numDeriv.

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

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

Le Thi, H. A., and P. D. Tao. 2001. "D.c. Programming Approach to the Multidimensional Scaling Problem." In *From Local to Global Optimization*, edited by A. Migdalas, P. M. Pardalos, and P. Värbrand, 231–76. Springer Verlag.

———. 2018. "DC Programming and DCA: Thirty Years of Developments." *Mathematical Programming, Series B*.

———. 2024. "Open Issues and Recent Advances in DC Programming and DCA." *Journal of Global Optimization* 88: 533–90.

Mersmann, O. 2023. *microbenchmark: Accurate Timing Functions*. https://CRAN.R-project.org/package=microbenchmark.

Meyer, R. R. 1976. "Sufficient Conditions for the Convergence of Monotonic Mathematical Programming Algorithms." *Journal of Computer and System Sciences* 12: 108–21.

Ortega, J. M., and W. C. Rheinboldt. 1970. *Iterative Solution of Nonlinear Equations in Several Variables*. New York, N.Y.: Academic Press.

Ostrowski, A. M. 1966. *Solution of Equations and Systems of Equations*. New York, N.Y.: Academic Press.

Ram, N., and S. Sabach. 2024 (in press). "A Globally Convergent Inertial First-Order Optimization Method for Multidimensional Scaling." *Journal of Optimization Theory and Applications*, 2024 (in press).

Robert, F. 1967. "Calcul du Rapport Maximal de Deux Normes sur $\mathbb{R}^n$." *Revue Francaise d'Automatique, d'Informatique Et De Recherche Operationelle* 1: 97–118.

Robini, M., L. Wang, and Y. Zhu. 2024. "The Appeals of Quadratic Majorization-Minimization."

Journal of Global Optimization 89: 509–58.
Rockafellar, R. T. 1970. *Convex Analysis*. Princeton University Press.