Robust Least Squares Multidimensional Scaling

Jan de Leeuw

September 23, 2024

We use an iteratively reweighted version of the smacof algorithm to minimize various robust multidimensional scaling loss functions. Our results depend strongly on a general theorem on sharp quadratic majorization of De Leeuw and Lange (2009).

1 Introduction

The title of this chapter seems something paradoxical. Least squares estimation is typically not robust, it is sensitive to outliers and pays a lot of attention to fitting the larger observations. What we mean by robust least squares MDS, however, is using the smacof machinery designed to minimize loss of the form

$$\sigma_2(X) := \sum w_k(\delta_k - d_k(X))^2, \tag{1}$$

to minimize robust loss functions. The prototypical robust loss function is

$$\sigma_1(X) := \sum w_k|\delta_k - d_k(X)|, \tag{2}$$

which we will call *strife*, because stress, sstress, and strain are already taken.

Strife is not differentiable at configurations X for which there is at least one k for which either $d_k(X) = \delta_k$ or $d_k(X) = 0$ (or both). This lack of differentiability complicates the minimization problem. Moreover experience with one-dimensional and city block MDS suggests that having many points where the loss function is not differentiable leads to (many) additional local minima.

In this chapter we will discuss (and implement) various variations of σ_1 from (2). They can be interpreted in two different ways. On the one hand we use smoothers of the absolute value function, and consequently of strife. This is not unlike the distance smoothing used by Pliner (1996) and Groenen, Heiser, and Meulman (1999) in the global minimization of σ_2 from (1). On the other hand our modified loss function can be interpreted as more robust versions of the least squares loss function, and consequently of stress.

Our robust or smoother loss functions are all of the form

$$\sigma(X) := \sum w_k\, f(\delta_k - d_k(X)), \tag{3}$$

for a suitable choice of the real valued function f. We will define what we mean by "suitable" later on. For now, note that loss (1) is the special case with $f(x) = x^2$ and loss (**??**) is the special case with $f(x) = |x|$.

2 Majorizing Strife

The pioneering work in strife minimization using smacof is Heiser (1988), building on earlier work in Heiser (1987). It is based on a creative use of the Arithmetic Mean-Geometric Mean (AM/GM) inequality to find a majorizer of the absolute value function. For the general theory of majorization algorithms (now more commonly known as MM algorithms) we refer to their original introduction in De Leeuw (1994) and to the excellent book by Lange (2016).

The AM/GM inequality says that for all non-negative x and y we have

$$|x||y| = \sqrt{x^2y^2} \leq \frac{1}{2}(x^2 + y^2), \tag{4}$$

with equality if and only if $x = y$. If $y > 0$ we can write (4) as

$$|x| \leq \frac{1}{2}\frac{1}{|y|}(x^2 + y^2), \tag{5}$$

and this provides a quadratic majorization of $|x|$ at y. There is no quadratic majorization of $|x|$ at $y = 0$, which is a nuisance we must deal with.

Using the majorization (5), and assuming $\delta_k \neq d_k(Y)$ for all k, we define

$$\omega_1(X) := \frac{1}{2}\sum w_k \frac{1}{|\delta_k - d_k(Y)|}((\delta_k - d_k(Y))^2 + (\delta_k - d_k(X))^2). \tag{6}$$

Now $\sigma_1(X) \leq \omega_1(X)$ for all X and $\sigma_1(Y) = \omega_1(Y)$. Thus ω_1 majorizes σ_1 at Y.

But what if for some k we have $d_k(Y) = \delta_k$? Define

$$s_k(Y) = \begin{cases} w_k \frac{1}{|\delta_k - d_k(Y)|} & \text{if } d_k(Y) \neq \delta_k, \\ 2\ \max_k w_k & \text{otherwise.} \end{cases} \tag{8}$$

Redefine ω_1 as

$$\omega_1(X) := \frac{1}{2}\sum s_k(Y)\{(\delta_k - d_k(Y))^2 + (\delta_k - d_k(X))^2\}. \tag{9}$$

This modified ω_1 majorizes σ_1 at Y, even in the case that $\delta_k = d_k(Y)$ for all k.

Reweighted smacof to minimize strife computes $X^{(k+1)}$ by decreasing

$$\sum s_k(X^{(k)})(\delta_k - d_k(X^{(k)}))^2, \tag{10}$$

using a standard smacof step. It then computes the new weights $s_k(X^{(k+1)})$ and uses them in the next smacof step to update $X^{(k+1)}$. And so on, until convergence.

To illustrate the problems with differentability we compute the directional derivatives of strife.

Let $s_k(X) := w_k|d_k(X) - \delta_k|$.

1. If $\delta_k = 0$ and $d_k(X) = 0$ then $ds_k(X;Y) = w_k d_k(Y)$.
2. If $\delta_k > 0$ and $d_k(X) = 0$ then $ds(X;Y) = -w_k d_k(Y)$.
3. If $d_k(X) > 0$ and $d_k(X) - \delta_k > 0$ then $ds_k(X;Y) = w_k \frac{1}{d_k(X)}\text{tr } X'A_kY$.
4. If $d_k(X) > 0$ and $d_k(X) - \delta_k < 0$ then $ds_k(X;Y) = -w_k \frac{1}{d_k(X)}\text{tr } X'A_kY$.
5. If $d_k(X) > 0$ and $d_k(X) - \delta_k = 0$ then $ds_k(X;Y) = w_k \frac{1}{d_k(X)}|\text{tr } X'A_kY|$.

The directional derivative of σ_1 is consequently the sum of five terms, corresponding with each of these five cases.

In the case of σ_2 the directional derivatives could be used to prove that if $w_k\delta_k > 0$ for all k stress then is differentiable at each local minimum (De Leeuw (1984)). For strife to be differentiable we would have to prove that at a local minimum both $d_k(X) > 0$ and $(d_k(X) - \delta_k) \neq 0$. So far I have no proof and no counter example, but it's early in the game.

3 Generalizing Strife

The AM/GM inequality was used in the previous section to construct a quadratic majorization of strife. To fix the terminology we say that a function g *majorizes* a function f at y if $g(x) \geq f(x)$ for all x and $g(y) = f(y)$. Majorization is *strict* if $g(x) > f(x)$ for all $x \neq y$. If $\mathfrak{H}$ is a family of functions that all majorize f at y then $h \in \mathfrak{H}$ is *sharp* if $h(x) \leq g(x)$ for all $g \in \mathfrak{H}$.

We are specifically interested in this chapter in sharp quadratic majorization, in which $\mathfrak{H}$ is the set of all convex quadratics that majorize f at y. This case has been studied in detail (in the case of real-valued functions on the line) by De Leeuw and Lange (2009). Their Theorem 4.5 on page 2478 says

> Theorem: Suppose $f(x)$ is an even, differentiable function on $\mathbb{R}$ such that the ratio $f'(x)/x$ is non-increasing on $(0,\infty)$. Then the even quadratic
>
> $$g(x) = \frac{f'(y)}{2y}(x^2 - y^2) + f(y) \tag{11}$$
>
> is a sharp quadratic majorizer of f at the point y.

We now apply this theorem to functions of the form

$$\sigma_f(X) := \sum w_k\, f(\delta_k - d_k(X)), \tag{12}$$

where f satisfies the conditions in the theorem. If

$$\omega_f(X) := \sum w_k \frac{f'(\delta_k - d_k(Y))}{2(\delta_k - d_k(Y))}\{(\delta_k - d_k(X))^2 - (\delta_k - d_k(Y))^2\} + f(\delta_k - d_k(Y)), \tag{13}$$

then ω_f is a sharp quadratic majorization at Y.

Although the absolute value is not differentiable at the origin the theorem can still be applied. It just does not give a majorizer at $y = 0$. If $f(x) = |x|$ then

$$g(x) = \frac{1}{2|y|}(x^2 - y^2) + |y| = \frac{1}{2|y|}(x^2 + y^2), labeleq: abssharp \tag{14}$$

which is the same as (5). Thus the AM/GM method gives the sharp quadratic majorization.

In iteration k the robust smacof algorithm does a smacof step towards minimization of ω_f over X. We can ignore the parts of (13) that only depend on Y, and minimize

$$\sum w_k(X^{(k)})(\delta_k - d_k(X))^2, \tag{15}$$

with

$$w_k(X^{(k)}) := w_k \frac{f'(\delta_k - d_k(X^{(k)}))}{2(\delta_k - d_k(Y))}. \tag{16}$$

It then recomputes the weights $w_k(X^{(k+1)})$ and goes to the smacof step again. This can be thought of as iterativey reweighted least squares, and also as majorization within majorization, with the smacof majorization within the sharp quadratic majorization of the loss function.

A straightforward variation of the algorithm does a number of smacof steps before upgrading the weights. This still leads to a monotone, and thus convergent, algorithm. How many smacof steps we have to take in the inner iterations is something that needs further study. It is likely to depend on the fit of the data, on the shape of the function near the local minimum, and on how far the iterations are from the local minimum.

4 Pseudo-Huber Loss, Charbonnier loss

De Leeuw (2018)

$$f(x) = \sqrt{x^2 + c^2}$$

$$f(x) = c\sqrt{1 + (\frac{x}{c})^2}$$

$$f'(x) = \frac{1}{\sqrt{x^2 + c^2}}x$$

$$\frac{f'(x)}{x} = \frac{1}{\sqrt{x^2 + c^2}}$$

which is decreasing.

$$\sigma_\epsilon(X) := \sum w_k \sqrt{(\delta_k - d_k(X))^2 + \epsilon^2}$$

Now majorization using

$$\sqrt{(\delta_k - d_k(X))^2 + \epsilon^2} \leq \frac{1}{2}\frac{1}{\sqrt{(\delta_k - d_k(Y))^2 + \epsilon^2}}\{(\delta_k - d_k(X))^2 + (\delta_k - d_k(Y))^2 + 2\epsilon^2\}$$

Alt:

$$\sigma_\epsilon(X) = \epsilon^2 \left\{\sqrt{1 + \frac{x^2}{\epsilon^2}} - 1\right\}$$

5 Robustifying - Huber Loss

The Huber function (Huber (1964)) is

$$f(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| < c, \\ c|x| - \frac{1}{2}c^2 & \text{otherwise.} \end{cases}$$

The Huber function is differentiable, although not twice diffentiable. Its derivative is

$$f'(x) = \begin{cases} c & \text{if } x \geq c, \\ x & \text{if } |x| \leq c, \\ -c & \text{if } x \leq -c. \end{cases}$$

The Huber function is even and differentiable. Moreover $f'(x)/x$ decreases from. Thus the theorem applies and the sharp quadratic majorizer at y is

$$g(x) = \{$$

$$\sigma_k(X) = \begin{cases} \frac{1}{2}(\delta_k - d_k(X))^2 & \text{if } |\delta_k - d_k(X)| < c, \\ c|\delta_k - d_k(X)| - \frac{1}{2}c^2 & \text{if } |\delta_k - d_k(X)| \geq c. \end{cases}$$

$$\omega_k(x, y) = \begin{cases} \frac{1}{2}\frac{c}{|y|}(x^2 - y^2) - cy - \frac{1}{2}c^2 & \text{if } y \leq -c, \\ \frac{1}{2}x^2 & \text{if } |y| < c, \\ \frac{1}{2}\frac{c}{|y|}(x^2 - y^2) + cy - \frac{1}{2}c^2 & \text{if } y \geq +c. \end{cases}$$

Now $x=\delta_k-d_k(X)$ and $y=\delta_k-d_k(Y)$

$$\omega_k(X;Y)=\begin{cases}\frac12\frac{c}{|\delta_k-d_k(Y)|}\{(\delta_k-d_k(X))^2+(d_k(Y)-\delta_k)^2\}-c(\delta_k-d_k(Y))-\frac12c^2 & \text{if }\delta_k-d_k(Y)\leq\\ \frac12(\delta_k-d_k(X))^2 & \text{if }|\delta_k-d_k(Y)|\\ \frac12\frac{c}{|\delta_k-d_k(Y)|}\{(\delta_k-d_k(X))^2+(d_k(Y)-\delta_k)^2\}+c(\delta_k-d_k(Y))-\frac12c^2 & \text{if }\delta_k-d_k(Y)\geq\end{cases}$$

Thus the MDS majorization algorithm for the Huber loss is to update Y by minimizing (or by performing one smacof step to decrease)

$$\sum w_k(Y)(\delta_k-d_k(X))^2$$

where

$$w_k(Y)=\begin{cases}w_k & \text{if }|\delta_k-d_k(Y)|<c,\\ \frac{cw_k}{|\delta_k-d_k(Y)|} & \text{otherwise.}\end{cases}$$

6 Tukey biweight

$$f(x)=\begin{cases}\frac{c^2}{6}\left(1-\left[1-(\frac{x}{c})^2\right]^3\right) & \text{if }|x|\leq c,\\ \frac{c^2}{6} & \text{otherwise .}\end{cases}$$

$$f'(x)=\begin{cases}x\left[1-(\frac{x}{c})^2\right]^2 & \text{if }|x|\leq c,\\ 0 & \text{otherwise .}\end{cases}$$

It is easy to see that $f'(x)/x$ is non-increasing on $(0,+\infty)$.

$$w_k(Y)=\begin{cases}\frac12w_k\left[1-(\frac{\delta_k-d_k(Y)}{c})^2\right]^2 & \text{if }|\delta_k-d_k(Y)|<c,\\ 0 & \text{otherwise.}\end{cases}$$

7 Convolution

In De Leeuw (2018) we also study the convolution smoother proposed by Voronin, Ozkaya, and Yoshida (n.d.). The idea is to use the convolution of the absolute value function and a *mollifier* as the smoothed function.

A smooth function $\psi : \mathbb{R} \rightarrow \mathbb{R}$ is said to be a pdf if it is non-negative, and has area $\int \psi(x)dx = 1$. For any pdf ψ and any $c > 0$, define the parametric function $\psi_c : \mathbb{R} \rightarrow \mathbb{R}$ by: $\psi_c(x) := \frac{1}{c}\psi(\frac{1}{c})$, for all $x \in \mathbb{R}$. Then $\{\psi_c : c > 0\}$ is a family of pdf's, whose support decreases as $c \rightarrow 0$, but the volume under the graph always remains equal to one.

choose a Gaussian pdf.

$$f(x) = \frac{1}{c\sqrt{2\pi}} \int_{-\infty}^{+\infty} |x-y| \exp\left\{-\frac{1}{2}(\frac{y}{c})^2\right\} dy$$

Carrying out the integration gives

$$f(x) = x\{2\Phi(x/c) - 1\} + 2c\phi(x/c).$$

The derivative is

$$f'(x) = 2\Phi(x/c) - 1$$

It may not be immediately obvious in this case that $f'(x)/x$ is decreasing. We prove that its derivative is negative on $(0, +\infty)$. The derivative of $f'(x)/x$ has the sign of $xf''(x) - f'(x)$, which is $z\phi(z) - \Phi(z) + 1/2$, with $z = x/c$. It remains to show that $\Phi(z) - z\phi(z) \geq \frac{1}{2}$, or equivalently that $\int_0^z \phi(x)dx - z\phi(z) \geq 0$. Now if $0 \leq x \leq z$ then $\phi(x) \geq \phi(z)$ and thus $\int_0^z \phi(x)dx \geq \phi(z) \int_0^z dx = z\phi(z)$, which completes the proof.

$$w_k(Y) = \frac{\Phi((\delta_k - d_k(Y))/c) - \frac{1}{2}}{\delta_k - d_k(Y)}$$

8 Barron Loss

Not surprisingly there are a large number of generalizations of Huber-like losses in the engineering community, and in their maze of conference publications. Without having any confidence of selecting a representative sample from the literature, we mention Barron (2017), Barron (2019), Gokcesu and Gokcesu (2021), Gokcesu and Gokcesu (2022). These papers also give a large number of possibly useful references.

It is also clear that we can use any location-scale family of probability densities to define convolution smoothers. There is an infinite number of possible choices, with finite or infinite support, smooth or nonsmooth, using splines or wavelets, and so on.

9 Example

10 Discussion

10.1 Fixed weights

11 Code

We wrote separate programs in R (R Core Team (2024)) for pseudo-Huber, Huber, Tukey, and the convulution smoother. Each programs is about 50 lines of code, and they differ in less tha 10 of the 50 lines. That leads to a lot of duplicate code, obviously, and it would make sense to merge the four programs into a single one.

```
smacofRobustPseudoHuber <- function(delta,
                                    weights = 1 - diag(nrow(delta)),
                                    ndim = 2,
                                    cons = 0,
                                    itmax = 1000,
                                    eps = 1e-15,
                                    verbose = TRUE) {
  nobj <- nrow(delta)
  wmax <- max(weights)
  xold <- smacofTorgerson(delta, ndim)
  dold <- as.matrix(dist(xold))
  rold <- sqrt((delta - dold) ^ 2 + cons)
  sold <- sum(weights * rold)
  wold <- weights / (rold + diag(nobj))
  itel <- 1
  repeat {
    vmat <- -wold
    diag(vmat) <- -rowSums(vmat)
    vinv <- solve(vmat + (1 / nobj)) - (1 / nobj)
    bmat <- -wold * delta / (dold + diag(nobj))
    diag(bmat) <- -rowSums(bmat)
    xnew <- vinv %*% (bmat %*% xold)
    dnew <- as.matrix(dist(xnew))
    rnew <- sqrt((delta - dnew) ^ 2 + cons)
    wnew <- weights / (rnew + diag(nobj))
```

```r
    snew <- sum(weights * rnew)
    if (verbose) {
      cat(
        "itel ",
        formatC(itel, width = 4, format = "d"),
        "sold ",
        formatC(sold, digits = 10, format = "f"),
        "snew ",
        formatC(snew, digits = 10, format = "f"),
        "\n"
      )
    }
    if ((itel == itmax) || ((sold - snew) < eps)) {
      break
    }
    xold <- xnew
    dold <- dnew
    sold <- snew
    wold <- wnew
    rold <- rnew
    itel <- itel + 1
  }
  return(list(
    x = xnew,
    s = snew,
    d = dnew,
    r = rnew,
    itel = itel
  ))
}

smacofRobustHuber <- function(delta,
                              weights = 1 - diag(nrow(delta)),
                              ndim = 2,
                              cons = 0,
                              itmax = 1000,
                              eps = 1e-10,
                              verbose = TRUE) {
  nobj <- nrow(delta)
  wmax <- max(weights)
```

```
xold <- smacofTorgerson(delta, ndim)
dold <- as.matrix(dist(xold))
fold <- abs(delta - dold)
rold <- ifelse(fold < cons, (fold ^ 2) / 2, cons * fold - (cons ^ 2) / 2)
sold <- sum(weights * rold)
wold <- ifelse(fold < cons, weights, cons * weights / (fold + diag(nobj)))
itel <- 1
repeat {
  vmat <- -wold
  diag(vmat) <- -rowSums(vmat)
  vinv <- solve(vmat + (1 / nobj)) - (1 / nobj)
  bmat <- -wold * delta / (dold + diag(nobj))
  diag(bmat) <- -rowSums(bmat)
  xnew <- vinv %*% (bmat %*% xold)
  dnew <- as.matrix(dist(xnew))
  fnew <- abs(delta - dnew)
  rnew <- ifelse(fnew < cons, (fnew ^ 2) / 2, cons * fnew - (cons ^ 2) / 2)
  snew <- sum(weights * rnew)
  wnew <- ifelse(fnew < cons, weights, cons * weights / (fnew + diag(nobj)))
  if (verbose) {
    cat(
      "itel ",
      formatC(itel, width = 4, format = "d"),
      "sold ",
      formatC(sold, digits = 10, format = "f"),
      "snew ",
      formatC(snew, digits = 10, format = "f"),
      "\n"
    )
  }
  if ((itel == itmax) || ((sold - snew) < eps)) {
    break
  }
  xold <- xnew
  dold <- dnew
  sold <- snew
  wold <- wnew
  rold <- rnew
  itel <- itel + 1
}
```

```
  return(list(
    x = xnew,
    s = snew,
    d = dnew,
    r = rnew,
    itel = itel
  ))
}

smacofRobustTukey <- function(delta,
                              weights = 1 - diag(nrow(delta)),
                              ndim = 2,
                              cons = 0,
                              itmax = 1000,
                              eps = 1e-10,
                              verbose = TRUE) {
  nobj <- nrow(delta)
  wmax <- max(weights)
  xold <- smacofTorgerson(delta, ndim)
  dold <- as.matrix(dist(xold))
  fold <- delta - dold
  rold <- ((cons ^ 2) / 6) * ifelse(abs(fold) < cons, (1 - (1 - (fold / cons) ^ 2) ^ 3
  sold <- sum(weights * rold)
  wold <- ifelse(abs(fold) < cons, weights * (1 - (fold / cons) ^ 2) ^ 2, 0) / 2
  itel <- 1
  repeat {
    vmat <- -wold
    diag(vmat) <- -rowSums(vmat)
    vinv <- solve(vmat + (1 / nobj)) - (1 / nobj)
    bmat <- -wold * delta / (dold + diag(nobj))
    diag(bmat) <- -rowSums(bmat)
    xnew <- vinv %*% (bmat %*% xold)
    dnew <- as.matrix(dist(xnew))
    fnew <- delta - dnew
    rnew <- ((cons ^ 2) / 6) * ifelse(abs(fnew) < cons, (1 - (1 - (fnew / cons) ^ 2) ^
    snew <- sum(weights * rnew)
    wnew <- ifelse(abs(fnew) < cons, weights * (1 - (fnew / cons) ^ 2) ^ 2, 0) / 2
    if (verbose) {
      cat(
        "itel ",
        formatC(itel, width = 4, format = "d"),
        "sold ",
        formatC(sold, digits = 10, format = "f"),
        "snew ",
        formatC(snew, digits = 10, format = "f"),
        "\n"
      )
    }
    if ((itel == itmax) || ((sold - snew) < eps)) {
      break
    }
    xold <- xnew
    dold <- dnew
    sold <- snew
    wold <- wnew
    rold <- rnew
    itel <- itel + 1
  }
  return(list(
    x = xnew,
    s = snew,
    d = dnew,
    r = rnew,
    itel = itel
  ))
}

smacofRobustConvolution <- function(delta,
                              weights = 1 - diag(nrow(delta)),
                              ndim = 2,
                              cons = 0,
                              itmax = 1000,
                              eps = 1e-10,
                              verbose = TRUE) {
  nobj <- nrow(delta)
  wmax <- max(weights)
  xold <- smacofTorgerson(delta, ndim)
  dold <- as.matrix(dist(xold))
  fold <- delta - dold
  rold <- fold * (2 * pnorm(fold / cons) - 1) + 2 * cons * dnorm(fold / cons)
```

```
sold <- sum(weights * rold)
wold <- (pnorm(fold / cons) - 0.5) / (fold + diag(nobj))
itel <- 1
repeat {
  vmat <- -wold
  diag(vmat) <- -rowSums(vmat)
  vinv <- solve(vmat + (1 / nobj)) - (1 / nobj)
  bmat <- -wold * delta / (dold + diag(nobj))
  diag(bmat) <- -rowSums(bmat)
  xnew <- vinv %*% (bmat %*% xold)
  dnew <- as.matrix(dist(xnew))
  fnew <- delta - dnew
  rnew <- fnew * (2 * pnorm(fnew / cons) - 1) + 2 * cons * dnorm(fnew / cons)
  snew <- sum(weights * rnew)
  wnew <- (pnorm(fnew / cons) - 0.5) / (fnew + diag(nobj))
  if (verbose) {
    cat(
      "itel ",
      formatC(itel, width = 4, format = "d"),
      "sold ",
      formatC(sold, digits = 10, format = "f"),
      "snew ",
      formatC(snew, digits = 10, format = "f"),
      "\n"
    )
  }
  if ((itel == itmax) || ((sold - snew) < eps)) {
    break
  }
  xold <- xnew
  dold <- dnew
  sold <- snew
  wold <- wnew
  rold <- rnew
  itel <- itel + 1
}
return(list(
  x = xnew,
  s = snew,
  d = dnew,
    r = rnew,
    itel = itel
  ))
}

smacofTorgerson <- function(delta, ndim) {
  dd <- delta ^ 2
  rd <- apply(dd, 1, mean)
  md <- mean(dd)
  sd <- -.5 * (dd - outer(rd, rd, "+") + md)
  ed <- eigen(sd)
  return(ed$vectors[, 1:ndim] %*% diag(sqrt(ed$values[1:ndim])))
}
```

References

Barron, J. T. 2017. "A More General Robust Loss Function." https://arxiv.org/pdf/1701.03077v1.

———. 2019. "A General and Adaptive Robust Loss Function." In *Proceedings 2019 IEEE/CVF Conferehce on Computer Vision and Pattern Recognition*, 4331–39. https://openaccess.thecvf.com/content_CVPR_2019/papers/Barron_A_General_and_Adaptive_Robust_Loss_Function_CVPR_2019_paper.pdf.

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

———. 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.

———. 2018. "MM Algorithms for Smoothed Absolute Values." 2018.

De Leeuw, J., and K. Lange. 2009. "Sharp Quadratic Majorization in One Dimension." *Computational Statistics and Data Analysis* 53: 2471–84.

Gokcesu, K., and H. Gokcesu. 2021. "Generalized Huber Loss for Robust Learning and Its Efficient Minimization for a Robust Statistics." https://arxiv.org/abs/2108.12627.

———. 2022. "Nonconvex Extension of Generalized Huber Loss for Robust Learning and Pseudo-Mode Statistics." arXiv:2202.11141v1 [stat.ML]. https://arxiv.org/abs/2202.11141.

Groenen, P. J. F., W. J. Heiser, and J. J. Meulman. 1999. "Global Optimization in Least-Squares Multidimensional Scaling by Distance Smoothing." *Journal of Classification* 16: 225–54.

Heiser, W. J. 1987. "Correspondence Analysis with Least Absolute Residuals." *Computational Statistica and Data Analysis* 5: 337–56.

———. 1988. “Multidimensional Scaling with Least Absolute Residuals.” In *Classification and Related Methods of Data Analysis*, edited by H. H. Bock, 455–62. North-Holland Publishing Co.

Huber, P. J. 1964. “Robust Estimation of a Location Parameter.” *Annals of Mathematical Statistics* 35 (1): 73–101.

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

Pliner, V. 1996. “Metric Unidimensional Scaling and Global Optimization.” *Journal of Classification* 13: 3–18.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Voronin, S., G. Ozkaya, and Y. Yoshida. n.d. “Convolution Based Smooth Approximations to the Absolute Value Function with Application to Non-smooth Regularization.”