ALSCAL IN R

JAN DE LEEUW

ABSTRACT. Algorithms and R programs are described to minimize the sstress loss function used in Multidimensional Scaling using coordinate descent methods.

1. INTRODUCTION

Least squares multidimensional scaling (MDS) minimizes a loss function of the form

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

Here Δ is a known $n \times n$ matrix of dissimilarities, and W is a known $n \times n$ matrix of *weights*. The MDS problem is to minimize (1) over all $n \times p$ *configurations* X. Loss function (1) was introduced by Kruskal [1964a,b], who called it the *stress* of configuration X We have discussed some algorithms to minimize stress, and their implementation in R, in De Leeuw [2005b].

In this paper we look at the related problem of minimizing

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

In this least squares loss function, called *sstress* by Takane et al. [1977], squared distances instead of distances are fitted to the matrix of dissimilarities. There are quite a few algorithms available [Takane, 1977; Browne,

Date: January 25, 2012.

2000 *Mathematics Subject Classification.* 62H25.

Key words and phrases. Multivariate Analysis, Correspondence Analysis.

1987; De Leeuw et al., 2005] to minimize loss function (2), but we implement the original cyclic coordinate descent (CCD) method uses by Takane et al. [1977].

2. Symmetrc

We suppose throughout that W is symmetric, hollow (zero diagonal), and non-negative and that Δ is symmetric and hollow (not necessarily non-negative). These assumptions can be made without any real loss of generality [De Leeuw, 1977].

CCD methods change one coordinate of the configuration at the time, and cycle through all np coordinates. If we change x_{ks} to $\tilde{x}_{ks} = x_{ks} + \theta$ then x_i changes to $\tilde{x}_i = x_i + \theta\epsilon^{ik}e_s$, where superscripted ϵ is the Kronecker symbol. Thus ϵ^{ik} is equal to one if $i = k$ and zero otherwise.

$$d_{ij}^2(\tilde{X}) = (\tilde{x}_i - \tilde{x}_j)'(\tilde{x}_i - \tilde{x}_j) = \\ = d_{ij}^2(X) + 2\theta(\epsilon^{ik} - \epsilon^{jk})(x_{is} - x_{js}) + \theta^2(\epsilon^{ik} - \epsilon^{jk})^2,$$

and, defining the *residuals* $\rho_{ij}(X) = \delta_{ij} - d_{ij}^2(X)$,

$$\sigma(\tilde{X}) = \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n w_{ij}(\rho_{ij}(X) - 2\theta(\epsilon^{ik} - \epsilon^{jk})(x_{is} - x_{js}) - \theta^2(\epsilon^{ik} - \epsilon^{jk})^2)^2$$

Clearly $\sigma(\tilde{X})$ is a quartic in θ. Moreover this quartic is a non-negatively weighted sum of squares of quadratics, which means it is non-negative and has a minimum.

Because W is hollow, we can use the identity $w_{ij}(\epsilon^{ik} - \epsilon^{jk})^2 = w_{ij}(\epsilon^{ik} + \epsilon^{jk})$ to simplify terms. The quartic becomes

$$\sigma(\tilde{X}) = a_0(X) + a_1(X)\theta + a_2(X)\theta^2 + a_3(X)\theta^3 + a_4(X)\theta^4,$$

where

$$
\begin{aligned}
a_0(X) &= \sigma(X),\\
a_1(X) &= -4\sum_{j=1}^n w_{kj}\rho_{kj}(X)(x_{ks}-x_{js}),\\
a_2(X) &= 4\sum_{j=1}^n w_{kj}(x_{ks}-x_{js})^2 - 2\sum_{j=1}^n w_{kj}\rho_{kj}(X),\\
a_3(X) &= 2\sum_{j=1}^n w_{kj}(x_{ks}-y_{js}),\\
a_4(X) &= w_{k\bullet}.
\end{aligned}
$$

To find the minimizer and minimum of the quartic, we have implemented a formula given by Jeffrey [1997]. This formula are applied cyclically to each coordinate, using simple updates for the configuration and the squared distances.

3. Rectangular

In the rectangular or *unfolding* case the sstress loss function becomes

$$\sigma(X,Y) = \sum_{i=1}^n\sum_{j=1}^m w_{ij}(\delta_{ij} - d_{ij}^2(X,Y))^2$$

Special algorithms for this case have been developed by **?**De Leeuw [2005a]. We simply adapt the CCA method.

If we change x_{ks} to $\tilde{x}_{ks} = x_{ks} + \theta$ then

$$d_{ij}^2(\tilde{X},Y) = (\tilde{x}_i - y_j)'(\tilde{x}_i - y_j) = d_{kj}^2(X,Y) + 2\epsilon^{ik}\theta(x_{is}-y_{js}) + \epsilon^{ik}\theta^2,$$

and

$$\sigma(\tilde{X},Y) = \sum_{i\neq k}^n \sigma_{i\bullet} + \sum_{j=1}^m w_{kj}(\rho_{kj}(X,Y) - 2\theta(x_{ks}-y_{js}) - \theta^2)^2.$$

Again $\sigma(\tilde{X},Y)$ is a non-negative quartic in θ. By simplifying we see the quartic is

$$\sigma(\tilde{X},Y) = a_0(X,Y) + a_1(X,Y)\theta + a_2(X,Y)\theta^2 + a_3(X,Y)\theta^3 + a_4(X,Y)\theta^4,$$

where

$$\begin{aligned}
a_0(X,Y) &= \sigma(X,Y),\\
a_1(X,Y) &= -4\sum_{j=1}^m w_{kj}\rho_{kj}(X,Y)(x_{ks}-y_{js}),\\
a_2(X,Y) &= 4\sum_{j=1}^m w_{kj}(x_{ks}-y_{js})^2 - 2\sum_{j=1}^m w_{kj}\rho_{kj}(X,Y),\\
a_3(X,Y) &= 4\sum_{j=1}^m w_{kj}(x_{ks}-y_{js}),\\
a_4(X,Y) &= w_{k\bullet}.
\end{aligned}$$

If we change $y_{\ell s}$ to $\tilde{y}_{\ell s} = y_{\ell s} + \theta$ then

$$d_{i\ell}^2(X,\tilde{Y}) = (x_i - \tilde{y}_\ell)'(x_i - \tilde{y}_\ell) = d_{i\ell}^2(X,Y) - 2\theta(x_{is} - y_{\ell s}) + \theta^2,$$

and

$$\sigma(X,\tilde{Y}) = \sum_{j\neq\ell}^n \sigma_{\bullet,j} + \sum_{i=1}^m w_{i\ell}(\rho_{i\ell}(X,Y) + 2\theta(x_{is}-y_{\ell s}) - \theta^2)^2.$$

This means

$$\sigma(X,\tilde{Y}) = a_0(X,Y) + a_1(X,Y)\theta + a_2(X,Y)\theta^2 + a_3(X,Y)\theta^3 + a_4(X,Y)\theta^4,$$

where

$$\begin{aligned}
a_0(X,Y) &= \sigma(X,Y),\\
a_1(X,Y) &= 4\sum_{i=1}^n w_{i\ell}\rho_{i\ell}(X,Y)(x_{is}-y_{\ell s}),\\
a_2(X,Y) &= 4\sum_{i=1}^n w_{i\ell}(x_{is}-y_{\ell s})^2 - 2\sum_{i=1}^n w_{i\ell}\rho_{i\ell}(X,Y),\\
a_3(X,Y) &= -4\sum_{i=1}^n w_{i\ell}(x_{is}-y_{\ell s}),\\
a_4(X,Y) &= w_{\bullet\ell}.
\end{aligned}$$

4. Three-way Data

Appendix A. Code

```
#
# alscal package
# functions to minimize sstress
#
# version 0.0.1  only metric, one matrix
#
# version 0.1.0       02-06-06 (added elegant
    algorithm)
# version 0.1.1       02-06-06 (elegant algorithm
    with power iterations)
# version 0.1.2     02-07-06 (removed elegant to
    separate package)


require("mdsutils")
require("lowpoly")

alscalRect<-function(
        delta,
        w="none",
        p=2,
        init="svd",
        verbose=FALSE,
        ithist=TRUE,
        itmax=100,
        eps=1e-6) {
n<-dim(delta)[1]; m<-dim(delta)[2]; itel<-1
if (!is.matrix(w)) w<-matrix(1,n,m)
nn<-1:n; mm<-1:m; pp<-1:p; wr<-rowSums(w); wc<-colSums
    (w)
if (is.list(init)) {
        x<-init[[1]]; y<-init[[2]]
```

```
            }
else {
            e<- -0.5*(delta-outer(rowSums(delta)/m,colSums(delta)/n,"+")+(sum(delta)/(n*m)))
            z<-svd(e,nu=p,nv=0); x<-z$u; y<-crossprod(e,x)
            }
d<-dist2Rect(x,y); r<-w*(delta-d); t<-r*(delta-d)
rr<-rowSums(r); rc<-colSums(r)
tr<-rowSums(t); tc<-colSums(t); sstress<-sum(tc); psstress=sstress
repeat {
    for (i in nn) for (s in pp) {
            u<-x[i,s]-y[,s]; ww<-w[i,]
                a4<-wr[i]; a3<-4*sum(ww*u)
                a2<-(4*sum(ww*u*u))-(2*rr[i])
                a1<--4*sum(r[i,]*u); a0<-sstress
                mn<-minimizeQuartic(c(a0,a1,a2,a3,a4)); th<-mn[1]
                x[i,s]<-x[i,s]+th; d[i,]<-d[i,]+(2*th*u)+(th*th)
                r[i,]<-ww*(delta[i,]-d[i,]); t[i,]<-r[i,]*(delta[i,]-d[i,])
                rr[i]<-sum(r[i,]); tr[i]<-sum(t[i,]); sstress=sum(tr)
                if (verbose) print(sstress)
                }
            rc<-colSums(r); tc<-colSums(t)
    for (j in mm) for (s in pp) {
            u<-x[,s]-y[j,s]; ww<-w[,j]
                a4<-wc[j]; a3<--4*sum(ww*u)
                a2<-(4*sum(ww*u*u))-(2*rc[j])
                a1<-4*sum(r[,j]*u); a0<-sstress
```

```
                    mn<-minimizeQuartic(c(a0,a1,a2,a3,a4)); th<-mn[1]
                    y[j,s]<-y[j,s]+th; d[,j]<-d[,j]-(2*th*u)+(th*th)
                    r[,j]<-ww*(delta[,j]-d[,j]); t[,j]<-r[,j]*(delta[,j]-d[,j])
                    rc[j]<-sum(r[,j]); tc[j]<-sum(t[,j]); sstress=sum(tc)
                    if (verbose) print(sstress)
                    }
          rr<-rowSums(r); tr<-rowSums(t)
          if (ithist)     {
                    cat("Iteration: ",formatC(itel,digits=6,width=6),
                          "   SStress:   ",formatC(psstress,digits=6,width=12,format="f"),
                          " ==>",formatC(sstress,digits=6,width=12,format="f"),"\n")
                    }
          if (((psstress-sstress)<eps) || (itel==itmax)) break()
          psstress<-sstress; itel<-itel+1
}
return(list(x=x,y=y,d=d,sstress=sstress))
}

alscalSym<-function(
          delta,
          w=matrix(1,dim(delta)[1],dim(delta)[1]),
          p=2,
          init="torg",
          verbose=FALSE,
```

```
        ithist=TRUE,
        itmax=100,
        eps=1e-6) {
n<-dim(delta)[1]; itel<-1
nn<-1:n; pp<-1:p; wr<-rowSums(w)
if (is.matrix(init)) x<-init
else {
        e<- -0.5*(delta-outer(rowSums(delta)/n,colSums(delta)/n,"+")+(sum(delta)/(n*n)))
        z<-svd(e,nu=p,nv=0); x<-t(sqrt(z$d[1:p])*t(z$u))
        }
d<-dist2Sym(x); r<-delta-d; rr<-rowSums(w*r)
sstress=sum(w*r*r)/2; psstress<-sstress
repeat {
    for (i in nn) for (s in pp) {
            u<-x[i,s]-x[,s]; ww<-w[i,]
                a4<-wr[i]; a3<-2*sum(ww*u)
                a2<-(4*sum(ww*u*u))-(2*rr[i])
                a1<--4*sum(ww*r[i,]*u); a0<-sstress
                mn<-minimizeQuartic(c(a0,a1,a2,a3,a4)); th<-mn[1]
                x[i,s]<-x[i,s]+th; d[i,]<-d[i,]+(2*th*u)+(th*th)
                d[,i]<-d[,i]+(2*th*u)+(th*th); d[i,i]<-d[i,i]-(2*th*th)
                r<-delta-d; rr<-rowSums(w*r); sstress=sum(w*r*r)/2
                if (verbose) print(sstress)
                }
        if (ithist)     {
                cat("Iteration: ",formatC(itel,digits=6,width=6),
```

```
                    "␣␣␣SStress:␣␣␣",formatC(
                        psstress,digits=6,width=12,
                        format="f"),
                    "␣==>",formatC(sstress,digits
                        =6,width=12,format="f"),"\n
                        ")
            }
        if (((psstress-sstress)<eps) || (itel==itmax))
            break()
        psstress<-sstress; itel<-itel+1
}
return(list(x=x,d=d,sstress=sstress))
}
```

References

M.W. Browne. The Young-Householder Algorithm and the Least Squares Multdimensional Scaling of Squared Distances. *Journal of Classification*, 4:175–190, 1987.

J. De Leeuw. Algorithm Construction by Decomposition. In preparation, 2005a.

J. De Leeuw. SMACOF in R. In preparation, 2005b.

J. De Leeuw. Applications of Convex Analysis to Multidimensional Scaling. In J.R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, editors, *Recent developments in statistics*, pages 133–145, Amsterdam, The Netherlands, 1977. North Holland Publishing Company.

J. De Leeuw, P.J.F. Groenen, and R. Pietersz. Augmentation and Majorization Algorithms for Squared Distance Scaling. In preparation, 2005.

D.J. Jeffrey. Formulae, Algorithms, and Quartic Extrema. *Mathematics Magazine*, 70:349–356, 1997.

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.

Y. Takane. On the Relations among Four Methods of Multidimensional Scaling. *Behaviormetrika*, 4:29–42, 1977.

Y. Takane, F.W. Young, and J. De Leeuw. Nonmetric Individual Differences in Multidimensional Scaling: An Alternating Least Squares Method with Optimal Scaling Features. *Psychometrika*, 42:7–67, 1977.

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