Chapter 3

Algorithms for finding the nearest Euclidean distance matrix

3.1 Introduction

Symmetric matrices that have nonnegative offdiagonal elements and zero diagonal elements arise as data in many experimental sciences. This occurs when the values are measurements of distances between points in a Euclidean space. Such a matrix is referred to as a Euclidean distance matrix. Because of data errors such a matrix may not be exactly Euclidean and it is desirable to find the best Euclidean matrix which approximates the non–Euclidean matrix. The aim of this chapter is to study methods for solving this problem.

This chapter contains the projection algorithm described by Glunt, Hayden, Hong and Wells [1990]. This algorithm converges linearly or slower and globally using Algorithm 2.2.7. The disadvantage of the projection algorithm is the slow rate of convergence. This can be increased by using a quasi–Newton method which converges at superlinear order. Therefore, new unconstrained methods based on using quasi–Newton methods are described here.

Some applications of the above problem are given in Section 3.2 along with the definition of the Euclidean distance matrix and its characterization. The projection algorithm is given in

Section 3.3. In Section 3.4 various iterative schemes for an unconstrained programming problem are considered. In Section 3.5 other projection methods for solving the nearest Euclidean distance matrix problem are discussed. In Section 3.6 numerical comparisons of projection methods are carried out. Also numerical comparisons between the projection algorithm in Section 3.3 and unconstrained methods in Section 3.4 are carried out. In addition an example is given which gives more illustration of the unconstrained methods.

In Chapter 4 hybrid methods are considered. These methods take the advantage of both the above methods.

3.2 Euclidean distance matrix

Definition 3.2.1 (*Euclidean distance matrix*)

A matrix D is called a Euclidean distance matrix if it satisfies the following conditions:

i. D is a symmetric matrix : $d_{ij} = d_{ji} \quad \forall\, i,j = 1,\ldots,n$

ii. Diagonal elements are all zero: $d_{ii} = 0 \quad \forall\, i = 1,\ldots,n$

iii. There exist n points : $\mathbf{p}_1,\ldots,\mathbf{p}_n$ in $\Re^r$ $(r \leq n-1)$ such that

$$d_{ij} = \| \mathbf{p}_i - \mathbf{p}_j \|_2^2 \quad (1 \leq i,j \leq n).$$

The elements of D are the squared distances between pairs of points in r–dimensional Euclidean space. Now the Euclidean distance matrix problem can be expressed in the following form

Given a real symmetric matrix $F \in \Re^{n\times n}$, find the Euclidean distance matrix $D \in \Re^{n\times n}$ that minimizes

$$\| F - D \|_F. \tag{3.2.1}$$

The distance between A and B is defined by $\| A - B \|_F$.

A matrix F is an $n \times n$ symmetric data matrix with zero diagonal elements whose elements are regarded as approximate squared distances between pairs of points in a r–dimensional

Euclidean space. F is usually a distance matrix of squared distances f_{ij} between n points, e.g. atoms, stars, cities. Therefore, F must have certain obvious properties regardless of how distances are calculated, and how many spatial dimensions are allowed. The properties can be described as follows:

Definition 3.2.2 (*distance matrix*)

A matrix F is called a distance matrix if it satisfies the following conditions:

i. F is a symmetric matrix: $f_{ij} = f_{ji} \quad \forall\, i, j = 1, \ldots, n$

ii. Diagonal elements are all zero: $f_{ii} = 0 \quad \forall\, i = 1, \ldots, n$

iii. All off–diagonal elements are strictly greater than zero:

$$f_{ij} > 0, \quad \forall\, i \neq j.$$

The motivation for this study arises from the statistical problems of multidimensional scaling and ordination. In multidimensional scaling, an observed matrix is to be approximated by a Euclidean distances in a specified dimension. The differences between observed and fitted distances are minimized. A discussion of types of multidimensional scaling may be found in De Leeuw et. al. [1980]. An application of multidimensional scaling has been applied in geography Colledge and Rushton [1972], cartography Gilbert [1974], genetics Lalouel [1977], archeology Kendall [1971] and biochemistry Crippen [1977,1978]. A broader review of scaling with application and algorithms is given by Young [1984]. A book by Meulman [1986] gives additional related applications in multivariate analysis.

Other applications arise in the conformation of molecular structures from nuclear magnetic resonance data. For a given data matrix a Euclidean distance matrix can be minimized to generate a molecular model in $\Re^3$ (see Havel et. al. [1983] and Crippen [1977,1978]). In conformation calculations Euclidean distance matrices are used to represent the squares of distances between the atoms of a molecular structure. Attempts to determine such a structure by nuclear–magnetic–resonance experiments give rise to a distance matrix F which because of data errors, may not be Euclidean.

Important characterizations for the Euclidean distance matrix which are used in the following sections are given in the following.

Schoenberg [1935] gave a modern characterization of Euclidean distance matrices. Young and Householder [1938] independently obtained similar result given in the following theorem. Schoenberg uses the fact that the first vector $\mathbf{p}_1$ in the Euclidean distance matrix definition can be translated to the origin.

Theorem 3.2.3

The distance matrix $D \in \Re^{n\times n}$ is a Euclidean distance matrix if and only if the $n-1 \times n-1$ symmetric matrix A defined by

$$a_{ij} \;=\; \tfrac{1}{2}[\, d_{1i} \;+\; d_{1j} \;-\; d_{ij}] \quad (2 \leq \; i,j \; \leq n) \tag{3.2.2}$$

is positive semi–definite, and D is irreducibly embeddable in $\Re^r$ $(r < n)$ where $r \;=\; rank(A)$.

Moreover, consider the spectral decomposition

$$A \;=\; U\Lambda U^T. \tag{3.2.3}$$

Let Λ_r be the matrix of non–zero eigenvalues in Λ and define X by

$$X \;=\; U_r\Lambda_r^{1/2}, \quad then \;\; A \;=\; XX^T \tag{3.2.4}$$

where $\Lambda_r^{1/2} \;\in\; \Re^{r\times r}$ and $U_r \;\in\; \Re^{n-1\times r}$ comprises the corresponding columns of U. Then the columns of X^T furnish coordinate choices for $\mathbf{p}_2,\; \mathbf{p}_3,\; \ldots,\; \mathbf{p}_n$ with $\mathbf{p}_1 \;=\; \mathbf{0}$.

Proof

First let D be Euclidean distance matrix and we aim to prove that A is positive semi–definite. Let $\mathbf{x} \;\in\; \Re^{n-1}$ then

$$\begin{aligned} \mathbf{x}^T A\mathbf{x} \;&=\; \tfrac{1}{2}\sum_{i,j=2}^{n}(d_{1i} \;+\; d_{1j} \;-\; d_{ij})x_{i-1}x_{j-1} \\ &=\; \sum_{i=2}^{n} d_{1i}x_{i-1}^2 \;+\; \sum_{\substack{i,j=2\\ i<j}}^{n}(d_{1i} \;+\; d_{1j} \;-\; d_{ij})x_{i-1}x_{j-1}. \end{aligned} \tag{3.2.5}$$

Also

$$\begin{aligned} d_{1i} + d_{1j} - d_{ij} &= \|\mathbf{p}_1 - \mathbf{p}_i\|^2 + \|\mathbf{p}_1 - \mathbf{p}_j\|^2 - \|\mathbf{p}_i - \mathbf{p}_j\|^2 \\ &= \sum_{k=1}^{r} p_{ik}^2 + \sum_{k=1}^{r} p_{jk}^2 - \sum_{k=1}^{r} (p_{ik} - p_{jk})^2 \\ &= 2 \sum_{k=1}^{r} p_{ik}\, p_{jk} \end{aligned} \tag{3.2.6}$$

where $\mathbf{p}_i^T = [p_{i1}, \ \ldots, p_{ir}]$. Hence from (3.2.5) and (3.2.6)

$$\begin{aligned} \mathbf{x}^T A \mathbf{x} &= \sum_{i=2}^{n} x_{i-1}^2 \sum_{k=1}^{r} p_{ik}^2 + 2 \sum_{\substack{i,j=2i \\ i<j}}^{n} x_{i-1} x_{j-1} \sum_{k=1}^{r} p_{ik} p_{jk} \\ &= \sum_{k=1}^{r} (x_1\, p_{2k} + x_2\, p_{3k} + \ \ldots \ + x_{n-1}\, p_{nk})^2 \\ &= \sum_{k=1}^{r} \ (\sum_{i=2}^{n} x_{i-1} p_{ik})^2 \end{aligned}$$

which is always nonnegative.

Conversly, let A be positive semi–definite, we shall prove that D is a Euclidean distance matrix. Let X be the orthogonal matrix defined by (3.2.4). Denote $\mathbf{p}_1 = 0$ and

$$\mathbf{p}_i = X \mathbf{e}_{i-1} \quad i = 2, \ldots, n$$

where $\mathbf{e}_i$ denotes columns of the unit matrix. Now

$$\|\mathbf{p}_1 - \mathbf{p}_i\|^2 = \|\mathbf{p}_i\|^2 = \mathbf{e}_{i-1}^T A \mathbf{e}_{i-1} = a_{ii} = d_{1i}.$$

Note that $A \in \Re^{n-1 \times n-1}$ such that

$$A = \begin{bmatrix} a_{22} & \ldots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{2n} & \ldots & a_{nn} \end{bmatrix}.$$

Also,

$$\begin{aligned} \|\mathbf{p}_i - \mathbf{p}_j\|^2 &= (\mathbf{e}_{i-1} - \mathbf{e}_{j-1})^T A (\mathbf{e}_{i-1} - \mathbf{e}_{j-1}) \\ &= a_{ii} + a_{jj} - 2a_{ij} \\ &= d_{1i} + d_{1j} - 2[\tfrac{1}{2}(d_{1i} + d_{1j} - d_{ij})] \\ &= d_{ij} \end{aligned}$$

which shows that $\mathbf{p}_1,\ \mathbf{p}_2,\ \ldots,\ \mathbf{p}_n$ are the n points satisfing the condition 3iii in Definition 3.2.1. Since $\mathbf{e}_i's$ are independent and $rank\ X\ =\ r$ then the $\mathbf{p}_i's$ are irreducibly embeddable in $\Re^r$. $\square$

Now we want to gives an alternative characterization for the Euclidean distance matrix in the following theorem and corollory. This will be used later in Section 3.3.

Theorem 3.2.4

Let $D \in \Re^{n\times n}$ be a distance matrix; then D is a Euclidean distance matrix if and only if

$$\mathbf{x}^T(-D)\mathbf{x}\ \geq\ 0 \quad \forall\ \mathbf{x}\ \in\ M$$

where

$$M\ =\ \{\ \mathbf{x}\in\Re^n :\ \ \mathbf{e}^T\mathbf{x}\ =\ 0\ \}. \tag{3.2.7}$$

Thus $-\ D\ \in\ K_M$.

Proof

Define

$$P\ =\ I\ -\ \mathbf{e}\mathbf{e}_1^T \tag{3.2.8}$$

where

$$\mathbf{e}_1^T\ =\ [\ 1\quad 0\quad 0\quad \ldots\quad 0\ ]$$

then

$$P\ =\ \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 \\ -1 & 1 & 0 & \ldots & 0 \\ -1 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \ldots & 1 \end{bmatrix}$$

which implies that

$$P(-D)P^T = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 2\,d_{21} & d_{21}+d_{31}-d_{32} & \dots & d_{21}+d_{n1}-d_{n2} \\ 0 & d_{21}+d_{31}-d_{32} & 2\,d_{31} & \dots & d_{31}+d_{n1}-d_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & d_{21}+d_{n1}-d_{n2} & d_{31}+d_{n1}-d_{n3} & \dots & 2\,d_{n1} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & A \end{bmatrix}. \tag{3.2.9}$$

Now we show that $P(-D)P^T \geq 0$ if and only if $-D$ is positive semi–definite in M. First let $P(-D)P^T$ be a positive semi–definite then we have

$$\mathbf{x}^T\,(P(-D)P^T)\,\mathbf{x} \geq 0 \qquad \forall\ \mathbf{x}.$$

Then when $\mathbf{x}^T\mathbf{e} = 0$ we have

$$\begin{aligned} 0 \leq \mathbf{x}^T\,(P(-D)P^T)\,\mathbf{x} &= (\mathbf{x}^T - \mathbf{x}^T\mathbf{e}\mathbf{e}_1^T)(-D)(\mathbf{x} - \mathbf{e}_1^T\mathbf{e}^T\mathbf{x}) \\ &= \mathbf{x}^T(-D)\mathbf{x} \end{aligned}$$

which implies that $-D$ is positive semi–definite in M.

Conversly, let $-D$ be a positive semi–definite in M then we have

$$\mathbf{y}^T\,(-D)\,\mathbf{y} \geq 0 \qquad \forall\ \mathbf{y}^T\mathbf{e} = 0.$$

We can express $\mathbf{x} = \lambda\,\mathbf{u} + \mathbf{y}$ where $\mathbf{u}$ is any vector such that $\mathbf{u}^T\mathbf{e} \neq 0$. Take $\mathbf{u} = \mathbf{e}_1$ then

$$\mathbf{x} = \mathbf{y} + \lambda\,\mathbf{e}_1^T$$

this implies that $\mathbf{e}^T\mathbf{x} = \lambda\,\mathbf{e}^T\mathbf{e}_1 = \lambda$ since $\mathbf{y}^T\mathbf{e} = 0$ and $\mathbf{e}^T\mathbf{e}_1 = 1$. It then follows that

$$\begin{aligned} \mathbf{y} &= \mathbf{x} - \lambda\,\mathbf{u} \\ &= \mathbf{x} - \mathbf{e}^T\mathbf{x}\,\mathbf{e}_1 \\ &= (I - \mathbf{e}_1\mathbf{e}^T)\mathbf{x} \\ &= P^T\mathbf{x}. \end{aligned}$$

For an arbitrary $\mathbf{x}$, this gives

$$\mathbf{x}^T(I - \mathbf{e}_1\mathbf{e}^T)^T(-D)\,(I - \mathbf{e}_1\mathbf{e}^T)\mathbf{x} \geq 0 \qquad \forall\ \mathbf{x}$$

$$\mathbf{x}^T P(-D)P^T\mathbf{x} \geq 0. \qquad \forall\ \mathbf{x}.$$

Using Theorem 3.2.3 the proof is established. □

Corollory 3.2.5

Let Q be the Householder matrix in (1.2.1) then the distance matrix $D = D^T \in \Re^{n\times n}$ is a Euclidean distance matrix if and only if the $n-1 \times n-1$ block D_1 in

$$Q(-D)Q = \begin{bmatrix} D_1 & \mathbf{d} \\ \mathbf{d}^T & \delta \end{bmatrix} \tag{3.2.10}$$

is positive semi–definite.

Proof

Follows from Theorem 1.3.6 □.

The advantage of the formulation in (3.2.10) over that given in (3.2.9) is that it provides the basis for the construction of a projection algorithm.

In the rest of this section other characterizations for the Euclidean distance matrix are given.

The problem of characterizing the Euclidean distance matrices among the distance matrices was first solved by Menger [1931] who based his analysis on the determinant of the form:

$$A_k = det\begin{pmatrix} 0 & 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & d_{12} & \dots & d_{1\ k-1} & d_{1k} \\ 1 & d_{21} & 0 & \dots & d_{2\ k-1} & d_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & d_{k-1\ 1} & d_{k-1\ 2} & \dots & 0 & d_{k-1\ k} \\ 1 & d_{k1} & d_{k2} & \dots & d_{k\ k-1} & 0 \end{pmatrix} \quad (k = 2, 3, \dots, n) \tag{3.2.11}$$

now known as Cayley–Menger determinant. The Menger result is given in a theorem due to Blumenthal [1953, pp99–100] Four years later, Schoenberg [1935] gave his result in Theorem 3.2.3. Another characterization is given by Hayden et. al. [1988], who show that a distance matrix $D \in \Re^{n\times n}$ is a Euclidean distance matrix if and only if the bordered matrix:

$$A = \begin{bmatrix} -D & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -d_{12} & \dots & -d_{1n} & 1 \\ -d_{21} & 0 & \dots & -d_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -d_{n1} & -d_{n2} & \dots & 0 & 1 \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix}. \tag{3.2.12}$$

has exactly one negative eigenvalue. Further, the n points represented by the matrix D (see Definition 3.2.1iii) are irreducibly embeddable in $\Re^r (r \leq n - 1)$ if and only if

$$r = n - 1 - dim\ N(D)$$

where $N(D)$ is the null space of D.

Other characterizations of the Euclidean distance matrix are also given in the literature.

3.3 The projection algorithm

Glunt et. al. [1990] give a description of an algorithm for computing the nearest Euclidean distance matrix, using the alternating projection method of Dykstra [1983] which guarantees convergence to the solution of problem (3.2.1). First an equivalent problem to (3.2.1) is given. This section includes a projection algorithm based on Dykstra's algorithm. The projection algorithm requires formulae, which are also given, for calculating the projection maps on to K_d (see (3.3.1)) and on to K_M. However in the first stage a formula for K_d and the normal cone of the intersection of K_d and K_M is given. The latter is used in problem (3.3.3) below.

Define

$$K_d \;=\; \{A:\; A \in \Re^{n\times n},\;\; A^T \;=\; A,\;\; a_{ii} \;=\; 0,\;\; i \;=\; 1,2,\;\; \ldots, n\}, \tag{3.3.1}$$

then $K_M \cap K_d$ is a convex cone.

The normal cone $\partial K_M(A)$ at $A \in K_M$ is given in (1.3.25).

Let $A \;\in\; K_d$ then it is clear that

$$\partial K_d(A) \;=\; \{B : B \;=\; diag\;[b_1, b_2,\;\; \ldots, b_n]\}. \tag{3.3.2}$$

A general result for the normal cone of the intersection of two sets has been given in (1.3.9). Using this we have the following theorem.

Theorem 3.3.1

If $A \;\in\; K_M \cap K_d$ then

$$\partial(K_M \cap K_d)(A) \;=\; \partial K_M(A) \;+\; \partial K_d(A)$$

Proof (this is a special case of Rockafellar [1970])

Clearly from Theorem 3.2.4 $D \in K_M \;\cap\; K_d$ if and only if $-D$ is Euclidean distance matrix. Further, the matrices in K_M are characterized by (3.2.10). Thus, the minimization problem (3.2.1) is a special case of the following problem:

Given a distance matrix $-F \;\in\; \Re^{n\times n}$

$$\begin{aligned} &minimize \;\; \| F \;-\; D \|_F \\ &subject \;\; to \;\; D \in K_M \;\cap\; K_d. \end{aligned} \tag{3.3.3}$$

Note that in problem (3.2.1) F and D are different from F and D in this problem. Now $\partial K_M(A)$ and $\partial K_d(A)$ are given in (1.3.25) and (3.3.2) respectively. From Theorem 3.3.1 and (2.1.3) we can deduce that D^* solves problem (3.3.3) if and only if

$$F - D^* \;=\; Q \begin{bmatrix} U \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & H \end{bmatrix} U^T & \mathbf{0} \\ \mathbf{0}^T & \mathbf{0} \end{bmatrix} Q \;+\; B. \tag{3.3.4}$$

Then (3.3.4) is equivalently to

$$F \;=\; Q \;\begin{bmatrix} U \begin{bmatrix} \Lambda_r & \mathbf{0} \\ \mathbf{0} & H \end{bmatrix} U^T & \mathbf{d} \\ \mathbf{d}^T & \delta \end{bmatrix} Q \;+\; B$$

since

$$D^* \;=\; Q \;\begin{bmatrix} U \begin{bmatrix} \Lambda_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} U^T & \mathbf{d} \\ \mathbf{d}^T & \delta \end{bmatrix} Q \tag{3.3.5}$$

this is true from Theorem 1.3.6 and (1.3.24) and $D^* \in K_M$ from (3.3.3).

Dykstra's algorithm depends crucially upon the computational complexity of the relevant projections. The minimization problem (3.3.3) is solved by applying Algorithm 2.2.7 to it. Problem (3.3.3) is to find the projection of a matrix to the intersection of two convex sets by a sequence of projections to the individual set successively. First we need definition for the projection maps $P_d(\cdot)$ and $P_M(\cdot)$, later formulae for them are obtained.

Definition 3.3.2

Let

$$K \;=\; \{A:\; A \;\in\; \Re^{n\times n}, A \;=\; A^T\},$$

then define the projection map $P_M(A)$ from K on to K_M and the projection map $P_d(A)$ from K on to K_d.

Since K_d is the subspace consisting of all real symmetric $n \times n$ matrices with zero diagonals then $P_d\,(F)$ is straightforwardly given by

$$P_d\,(F) \;=\; F \;-\; Diag\,(F) \tag{3.3.6}$$

i.e., P_d maps F to the matrix obtained by replacing each diagonal element by zero.

The projection map $P_M(A)$ formula on to K_M is now introduced but first a theorem due to Higham [1988] is given which is used in proving the formula.

Theorem 3.3.3

Let $F_1 \in \Re^{n-1\times n-1}$ be a symmetric matrix and $F_1 = U\Lambda U^T$ be its spectral decomposition, then $D_1 = U\Lambda^+ U^T$ solves the following problem

$$\begin{aligned} &\text{minimize} \quad \|F_1 - D_1\|_F \\ &\textit{subject to} \quad D_1 \geq 0 \end{aligned}$$

where

$$\Lambda = diag\,[\lambda_1,\ \lambda_1,\ \ldots,\ \lambda_{n-1}]$$

and

$$\Lambda^+ = diag\,[\lambda_i :\ \lambda_i = \left\{ \begin{array}{ll} \lambda_i, & \lambda_i \geq 0 \\ 0, & \lambda_i < 0 \end{array} \right\} i = 1,\ \ldots,\ n-1\ ].$$

Proof

Take any $D_1 \geq 0$ express $D_1 = UYU^T$ where $Y \geq 0$ then

$$\begin{aligned} \|F_1 - D_1\|_F^2 &= \|\Lambda - Y\|_F^2 \\ &= \sum_{\substack{i,j=1 \\ i\neq j}}^{n-1} y_{ij}^2 + \sum_{i=1}^{n-1}(\lambda_i - y_{ii})^2 \\ &\geq \sum_{\lambda_i<0}(\lambda_i - y_{ii})^2. \end{aligned}$$

It then follows because Y is positive semi–definite and $\lambda_i < 0$ that $-2\lambda_i y_{ii} \geq 0$ which implies that

$$\|F_1 - D_1\|_F^2 \geq \sum_{\lambda_i<0}\lambda_i^2.$$

This lower bound is attained uniquely for the matrix $Y = \Lambda^+$ because

$$\begin{aligned}\|F_1 - D_1\|_F^2 &= tr(F_1 - D_1)(F_1 - D_1) \\ &= (\Lambda - \Lambda^+)(\Lambda - \Lambda^+) \\ &= \sum_{i=1}^{n-1} \lambda_i - \sum_{i=1}^{n-1} \lambda_i^+ \\ &= \sum_{i:\lambda_i<0} \lambda_i^2.\end{aligned}$$

that is $D_1 = U\Lambda^+U^T$. $\square$

Now for calculating $P_M(F)$ we need to solve the following problem

$$\begin{aligned}&minimize \ \|F - D\|_F \\ &subject \ to \ D \in K_M.\end{aligned} \tag{3.3.7}$$

Let

$$F = Q \begin{bmatrix} F_1 & \mathbf{f} \\ \mathbf{f}^T & \zeta \end{bmatrix} Q \quad and \quad D = Q \begin{bmatrix} D_1 & \mathbf{d} \\ \mathbf{d}^T & \delta \end{bmatrix} Q$$

then $\|F - D\|_F$ is minimized by minimizing

$$\|F_1 - D_1\|_F \tag{3.3.8}$$

since

$$\|F - D\|_F = \|Q(F - D)Q\|_F = \left\| \begin{matrix} F_1 - D_1 & \mathbf{f} - \mathbf{d} \\ \mathbf{f}^T - \mathbf{d}^T & \zeta - \delta \end{matrix} \right\|_F.$$

Therefore from Theorem 3.3.3

$$P_M(F) = Q \begin{bmatrix} U\Lambda^+U^T & \mathbf{f} \\ \mathbf{f}^T & \zeta \end{bmatrix} Q, \tag{3.3.9}$$

is the solution of problem (3.3.7). Here Λ^+ corresponds to $\begin{bmatrix} \Lambda_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ in (3.3.5) such that

$$\begin{bmatrix} \Lambda_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \equiv \Lambda^+$$

In Chapter 2 von Neumann's algorithm was given which solves problem (3.3.3) in Algorithm 2.2.1. However, Example 2.2.4 illustrates the failure of von Neumann algorithm in solving problem (3.3.3) for general n. Moreover Algorithm 2.2.7 was given which successfully solves problem (3.3.3) for general n.

In particular, we have a matrix projections P_M and P_d given by (3.3.9) and (3.3.6) respectively. Using Algorithm 2.2.7 in case $m = 2$ with the projections $P_1 = P_M$ and $P_2 = P_d$ we have the following algorithm

Algorithm 3.3.4 (*projection algorithm*)

Given any distance matrix $-F \in \Re^{n\times n}$, let $F^{(0)} = F$

$$For \quad k = 1, 2, \quad \ldots$$
$$F^{(k+1)} = F^{(k)} + [P_d P_M(F^{(k)}) - P_M(F^{(k)})] \tag{3.3.10}$$

This algorithm is given by Glunt et. al. [1990]. The convergence of this algorithm follows from Theorem 2.2.8 in which $P_1 = P_d$ and $P_2 = P_M$. Given any distance matrix $-F = -F^T \in \Re^{n\times n}$, then the sequences $\{P_M(F^{(k)})\}$ and $\{P_d P_M(F^{(k)})\}$ generated by Algorithm 3.3.4 converge in the Frobenius norm to the solution D^* of (3.3.3).

It is important to realize whether a distance matrix $-F$ is a Euclidean distance matrix or not before solving problem (3.3.3). This is because if $-F$ is a Euclidean distance matrix then $D^* = F$ and there is no problem to solve. In the following a test is given to indicate if the matrix $-D^{(k)}$ is Euclidean distance matrix or not.

It follows by induction that off–diagonal elements of $F^{(k)}$, $k = 1, 2, \ldots$ in Algorithm 3.3.4 are the same as $F^{(0)}(= F)$. Denote

$$\Delta^{(k)} = F^{(k)} - F \tag{3.3.11}$$

which is diagonal. Then (3.3.10) can now be written

$$\Delta^{(k+1)} = \Delta^{(k)} - Diag\,(D^{(k)}) \tag{3.3.12}$$

where $D^{(k)} = P_M\,(F + \Delta^{(k)})$. Therefore given any $F^{(k)}$ (or $\Delta^{(k)}$), the test

$$Diag\,(D^{(k)}) = 0 \tag{3.3.13}$$

where $D^{(k)} = P_M(F^{(k)}) = P_M(F + \Delta^{(k)})$ determines whether the matrix $-D^{(k)}$ is Euclidean distance matrix or not. This test is useful in Chapter 4.

3.4 Unconstrained methods

In this section we shall consider a different approach to the problem (3.3.3). The main idea is to replace the problem (3.3.3) by an unconstrained optimization problem in order to use the superlinearly convergent quasi–Newton methods. Three methods for solving problem (3.3.3) will be given along with an example showing how the points $\mathbf{p}_i'$s are represented in the space with different methods. In the end of this section a strategy is described of how to choose the initial matrix X and the rank r.

Schoenberg [1935] and Young et. al. [1938] independently formulated a characterization of the Euclidean distance matrix in Theorem 3.2.3. Using this theorem problem (3.2.1) can be expressed as:

$$\begin{aligned} &\underset{D}{\text{minimize}} \quad \phi \\ &\textit{subject to} \quad A \geq 0 \end{aligned} \tag{3.4.1}$$

where

$$\phi = \|F - D\|_F^2,$$

$-F$ is a distance matrix and A is a function of D given by (3.2.2). (Note that the matrix $-D$ is the Euclidean distance matrix).

In the following analysis an equivalent unconstrained problem to (3.4.1) is derived.

From Definition 3.2.1 $-D$ is represented in the space $\Re^r$ by the following vectors

$$\mathbf{p}_1,\ \mathbf{p}_2,\ \ldots,\ \mathbf{p}_n. \tag{3.4.2}$$

It is always possible to put the first vector $\mathbf{p}_1$ at the origin by transforming each vector $\mathbf{p}_i$ to $\mathbf{p}_i - \mathbf{p}_1$ $i = 1, 2, \ldots, n$. Assume that the rank of A is known to be r ($1 \leq r < n$). The columns of X^T are the vectors $\mathbf{p}_2, \mathbf{p}_3, \ldots, \mathbf{p}_n$ (see Theorem 3.2.3). For quasi–Newton

methods it will be convenient to store the matrix X as a one vector with $r(n-1)$ variables as follows

$$X^T = \begin{bmatrix} x_1 & x_2 & \dots & x_{n-1} \\ x_n & x_{n+1} & \dots & x_{2(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t_1} & x_{t_2} & \dots & x_{r(n-1)} \end{bmatrix}. \tag{3.4.3}$$

where $t_1 = (r-1)(n-1)+1$ and $t_2 = (r-1)(n-1)+2$. Hence

$$\begin{aligned} \mathbf{p}_2^T &= [x_1 \quad x_n \quad \dots \quad x_{t_1} \quad] \\ \mathbf{p}_3^T &= [x_2 \quad x_{n+1} \quad \dots \quad x_{t_2} \quad] \\ &\vdots \\ \mathbf{p}_n^T &= [x_{n-1} \quad x_{2(n-1)} \quad \dots \quad x_{r(n-1)}] \end{aligned} \tag{3.4.4}$$

and

$$X^T = [\mathbf{p}_2 \quad \mathbf{p}_3 \quad \dots \quad \mathbf{p}_n]. \tag{3.4.5}$$

The constraint $A \geq 0$ is equivalent to $A = XX^T$, which can be expressed as

$$A = XX^T = \begin{bmatrix} \mathbf{p}_2^T \cdot \mathbf{p}_2 & \dots & \mathbf{p}_2^T \cdot \mathbf{p}_n \\ \vdots & \ddots & \vdots \\ \mathbf{p}_n^T \cdot \mathbf{p}_2 & \dots & \mathbf{p}_n^T \cdot \mathbf{p}_n \end{bmatrix}. \tag{3.4.6}$$

Therefore satisfying the constraint in problem (3.4.1) is equivalent to expressing the elements of the matrix D as a function of X. Hence from Definition 3.2.1iii and using (3.4.4)

$$\begin{aligned} d_{ij} &= d_{ji} = \|\mathbf{p}_i - \mathbf{p}_j\|^2 \quad i,j = 2,\dots,n \\ &= \sum_{k=0}^{r-1} (x_{i+km-1} - x_{j+km-1})^2, \quad i,j = 2,\dots,n \end{aligned} \tag{3.4.7}$$

and

$$d_{1i} = d_{1j} = \sum_{k=0}^{r-1} x_{i+km-1}^2. \quad i,j = 2,\dots,n \tag{3.4.8}$$

where $m = n - 1$ and $r = rank\, X$ (through out this section). Thus there exists a matrix X that satisfies (3.4.6) and hence the constraint $A \geq 0$ in problem (3.4.1).

An alternative way of deriving these expressions is the following, since $A = XX^T$ then

$$\begin{aligned} a_{ij} &= \sum_{k=1}^{r} x_{i+km-1} \cdot x_{j+km-1} \\ &= \sum_{k=1}^{r} (x_{i+km-1}^2 + x_{j+km-1}^2 - (x_{i+km-1} - x_{j+km-1})^2) \\ &= \tfrac{1}{2}[\, d_{1i} + d_{1j} - d_{ij}], \end{aligned}$$

$$d_{1i} = \sum_{k=0}^{r-1} x_{i+km-1}^2 \quad and \quad d_{1j} = \sum_{k=0}^{r-1} x_{j+km-1}^2 \qquad i, j = 2, \ldots, n \tag{3.4.9}$$

and

$$d_{ij} = d_{ji} = \sum_{k=0}^{r-1} (x_{i+km-1} - x_{j+km-1})^2, \qquad i, j = 2, \ldots, n \tag{3.4.10}$$

which is equivalent to (3.4.8) and (3.4.7).

Therefore, problem (3.4.1) above can be expressed in unconstrained form as follows:

$$\underset{X}{\text{minimize}} \quad \phi \tag{3.4.11}$$

where $\phi = \|F - D\|_F^2$, $-F$ is a distance matrix and the elements of the matrix D are given by (3.4.9) and (3.4.10).

Now three methods for solving problem (3.4.11) will be given. The quasi–Newton method (Algorithm 1.6.3) is used to solve problem (3.4.11) and the BFGS formula is used to update the Hessian matrix $H^{(k+1)}$. Quasi–Newton methods require only the function f and the first derivative $\mathbf{g}$ where f is ϕ and $\mathbf{g} = \nabla\phi$. However some difficulties arise, one of these is that the index $r\ (= rank(A))$ used in partitioning $A\ (= XX^T)$ is not known in advance. Fortunately it can be shown that by solving a sequence of problems for different r, each of which is well behaved, the correct value of r can be located. Excluding the function ϕ and the derivative $\nabla\phi$ which differ from one method to another, the procedures in the three methods are the same.

Method 3.4.1

Consider the matrix X with the vector $\mathbf{p}_1$ untranslated to the origin. Thus the vectors $\mathbf{p}_1,\ \mathbf{p}_2,\ \ldots\ ,\ \mathbf{p}_n$ are all vectors of variables, so that

$$\begin{aligned}
\mathbf{p}_1^T &= [x_1 \quad x_{n+1} \quad \ldots \quad x_{(r-1)n+1}] \\
\mathbf{p}_2^T &= [x_2 \quad x_{n+2} \quad \ldots \quad x_{(r-1)n+2}] \\
&\vdots \\
\mathbf{p}_n^T &= [x_n \quad x_{2n} \quad \ldots \quad x_{rn} \quad].
\end{aligned} \tag{3.4.12}$$

Then

$$X^T = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \\ x_{n+1} & x_{n+2} & \ldots & x_{2n} \\ x_{2n+1} & x_{2n+2} & \ldots & x_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t_3} & x_{t_4} & \ldots & x_{rn} \end{bmatrix}. \tag{3.4.13}$$

where $t_3 = (r-1)n+1$ and $t_4 = (r-1)n+2$. So the elements of the matrix $D \in \Re^{n\times n}$ take the form

$$d_{ij} = d_{ji} = \sum_{k=0}^{r-1}(x_{i+kn} - x_{j+kn})^2 \tag{3.4.14}$$

then

$$\begin{aligned}
\phi &= \sum_{i,j=1}^{n}(d_{ij} - f_{ij})^2 \\
&= \sum_{i,j=1}^{n}\{\sum_{k=0}^{r-1}(x_{i+kn} - x_{j+kn})^2 - f_{ij})^2 \\
&= 2\sum_{\substack{i,j=1 \\ i<j}}^{n}\{\sum_{k=0}^{r-1}(x_{i+kn} - x_{j+kn})^2 - f_{ij})^2
\end{aligned} \tag{3.4.15}$$

and

$$\nabla\phi = [\ \tfrac{\partial\phi}{\partial x_1} \quad \tfrac{\partial\phi}{\partial x_2} \quad \ldots \quad \tfrac{\partial\phi}{\partial x_{rn}}\ ]^T$$

where

$$\frac{\partial \phi}{\partial x_s} = 2\sum_{j=1}^{n}\{2[\sum_{k=0}^{r-1}(\ x_{l+kn} - x_{j+kn}\ )^2 - f_{lj}]\ \ 2(\ x_s - x_{j+tn}\ )\}$$

$$= 8\sum_{j=1}^{n}\{[\sum_{k=0}^{r-1}(\ x_{l+kn} - x_{j+kn}\ )^2 - f_{lj}](\ x_s - x_{j+tn}\ )\} \qquad (3.4.16)$$

for all $s = 1, \ \ldots\ , rn$ where $t = \frac{(s-l)}{n}$ and $l = mod\ (s, n)$ and if $l = 0$ then $l = n$.

Method 3.4.2

In this method, as explained earlier the first vector $\mathbf{p}_1$ is transformed to the origin (see Figures 3.4.1 and 3.4.2), so the number of variables is reduced from rn to $r(n-1)$. The matrix X^T is given in (3.4.3). The elements of the matrix D take the form

$$d_{i1} = d_{1j} = \sum_{k=0}^{r-1} x_{i+km-1}^2 \quad i = 2, \ \ldots\ , n \qquad (3.4.17)$$

$$d_{ij} = d_{ji} = \sum_{k=0}^{r-1}(x_{i+km-1} - x_{j+km-1}\ )^2 \quad i, j = 2, \ \ldots\ , n \qquad (3.4.18)$$

where $r = rank\ X$ and $m = n - 1$. Hence

$$\phi = \sum_{i,j=1}^{n}(d_{ij} - f_{ij})^2$$

$$= 2\{\sum_{i=1}^{n}(d_{i1} - f_{i1})^2 + \sum_{\substack{i,j=2 \\ i>j}}^{n}(d_{ij} - f_{ij})^2\}$$

$$= 2\{\sum_{i=1}^{n}(\sum_{k=0}^{r-1} x_{i+km-1}^2 - f_{i1})^2 + \sum_{\substack{i,j=2 \\ i>j}}^{n}(\sum_{k=0}^{r-1}(x_{i+km-1} - x_{j+km-1}\ )^2 - f_{ij})^2\} \qquad (3.4.19)$$

and

$$\nabla\phi \;=\; [\;\; \tfrac{\partial\phi}{\partial x_1} \quad \tfrac{\partial\phi}{\partial x_2} \quad \ldots \quad \tfrac{\partial\phi}{\partial x_{r(n-1)}} \;\;]^T.$$

where

$$\frac{\partial\phi}{\partial x_s} \;=\; 8x_s\{\sum_{k=0}^{r-1} x_{l+km}^2 \;-\; f_{l+1\ 1}\}$$
$$+\; 8\{\sum_{j=1}^{m}[\sum_{k=0}^{r-1}(\; x_{l+km} \;-\; x_{j+km}\;)^2 \;-\; f_{l+1\ j+1}](\; x_s \;-\; x_{j+tm}\;)\} \qquad (3.4.20)$$

for all $s \;=\; 1,\; \ldots\; , r(n \;-\; 1)$ where $t \;=\; \frac{(s\;-\;l)}{m}$ and $l \;=\; mod\;(s,m)$ and if $l \;=\; 0$ then $l \;=\; m.$

Method 3.4.3

In this method translation and rotation are used. First, the vector $\mathbf{p}_1$ transformed to the origin so the number of variables will be reduced to $r(n-1)$ as in the second method. Since $-D$ is a Euclidean distance matrix, it is always possible to make rotation about the origin, axes, planes and spaces depending on the dimension of r (e.g. if $r=3$ 2 rotations, $r=4$ 3 rotations etc.). Make a rotation in the second vector $\mathbf{p}_2$ around the origin until the vector $\mathbf{p}_2$ is located on one of the axes. Then the components of $\mathbf{p}_2$ are zeros except one, assume it is the first component. Similarly, rotate the vector $\mathbf{p}_3$ but this time around the axis where $\mathbf{p}_2$ located until $\mathbf{p}_3$ is located in one of the planes. Then the components of $\mathbf{p}_3$ are zeros except two; assume they are the first two.

Figures 3.4.1–6 shows an example of translation then rotation of vectors $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ in the space $\Re^3$.

If we continue likewize with the rest of the vectors $\mathbf{p}_4$, $\mathbf{p}_5$, $\ldots$, $\mathbf{p}_r$ then matrix X^T has the following form

$$X^T \;=\; \begin{bmatrix} x_1 & x_2 & x_3 & \ldots & \ldots & \ldots & x_{n-1} \\ 0 & x_n & x_{n+1} & \ldots & \ldots & \ldots & x_{2n-3} \\ 0 & 0 & x_{2n-2} & \ldots & \ldots & \ldots & x_{3n-6} \\ \vdots & \vdots & \vdots & \ddots & & & \vdots \\ 0 & 0 & 0 & \ldots & x_p & \ldots & x_q \end{bmatrix} \qquad (3.4.21)$$

where $p = (r-1)\,m - \frac{r(r-1)}{2}$ and $q = rm - \frac{r(r+1)}{2}$. Then the vectors $\mathbf{p}_1,\ \mathbf{p}_2,\ \ldots,\ \mathbf{p}_n$ have the form

$$\begin{array}{lcccccc}
\mathbf{p}_1^T & = & [0 & 0 & 0 & \ldots & 0 & 0] \\
\mathbf{p}_2^T & = & [x_1 & 0 & 0 & \ldots & 0 & 0] \\
\mathbf{p}_3^T & = & [x_2 & x_n & 0 & \ldots & 0 & 0] \\
\vdots & & & & & & & \\
\mathbf{p}_r^T & = & [x_{r-1} & x_{n+r-3} & x_{2n+r-4} & \ldots & x_{p+r-m} & 0] \\
\mathbf{p}_{r+1}^T & = & [x_r & x_{n+r-2} & x_{2n+r-5} & \ldots & x_{p+r-m+1} & x_p] \\
\vdots & & & & & & & \\
\mathbf{p}_n^T & = & [x_{n-1} & x_{2n-3} & x_{3n-6} & \ldots & x_{p-1} & x_q]
\end{array} \tag{3.4.22}$$

In this method, the number of variables is reduced to $rm - \frac{r(r+1)}{2}$ where $m = n-1$. Thus the number of variables is reduced by $\frac{r(r+1)}{2}$ from Method 3.4.2. The elements of the matrix $D \in \Re^{n\times n}$ ($-D$ is the Euclidean distance matrix) take the form

$$d_{i1} = d_{1j} = \sum_{k=0}^{p-1} x_{i+t}^2 \qquad i = 2,\ 3,\ \ldots,n \tag{3.4.23}$$

with $p = min\ (i-1, r)$ and $t = km - \frac{k(k+1)}{2} - 1$

$$d_{ij} = d_{ji} = \sum_{k=0}^{l-2}(x_{i+t} - x_{j+t})^2 + \sum_{k=l-1}^{p-1} x_{j+t}^2$$
$$for \quad i = 2,\ \ldots,r \ \ and \ \ j = i+1,\ \ldots,n \tag{3.4.24}$$

where $l = min\ (i,j)$, $p = min\ (j-1, r)$, $t = km - \frac{k(k+1)}{2} - 1$ and $m = n-1$. Also

$$d_{ij} = d_{ji} = \sum_{k=0}^{r-1}(x_{i+t} - x_{j+t})^2 \qquad \forall\ i,j = r+1,\ ,..,n \tag{3.4.25}$$

Thus

$$\begin{aligned}
\phi &= \sum_{i,j=1}^{n} [d_{ij} - f_{ij}]^2 \\
&= 2\{\sum_{i=2}^{n}[d_{i1} - f_{i1}]^2 \\
&\qquad + \sum_{i=2}^{r} \sum_{j=i+1}^{n} [d_{ij} - f_{ij}]^2 \\
&\qquad + \sum_{\substack{i,j=r+1 \\ i>j}}^{n} [d_{ij} - f_{ij}]^2\} \\
&= 2\{\sum_{i=2}^{n}[\sum_{k=0}^{p} x_{i+t}^2 - f_{i1}]^2 \\
&\qquad + \sum_{i=2}^{r} \sum_{j=i+1}^{n} [\sum_{k=0}^{l-2}(x_{i+t} - x_{j+t})^2 + \sum_{k=l-1}^{p-1} x_{j+t}^2 - f_{ij}]^2 \\
&\qquad + \sum_{\substack{i,j=r+1 \\ i>j}}^{n} [\sum_{k=0}^{r-1}(x_{i+t} - x_{j+t})^2 - f_{ij}]^2\}
\end{aligned} \tag{3.4.26}$$

and the gradient vector $\nabla\phi$ can be calculated from Algorithm 3.4.4 where

$$\nabla\phi = [\ \frac{\partial\phi}{\partial x_1} \quad \frac{\partial\phi}{\partial x_2} \quad \ldots \quad \frac{\partial\phi}{\partial x_q}\ ]^T$$

with $q = rm - \frac{r(r-1)}{2}$.

Algorithm 3.4.4 (*gradient calculation* : $\nabla\phi$)

$$\begin{aligned}
&q = 0 \\
&is = rm - \frac{r(r-1)}{2} \\
&For \quad s = 1,\ 2,\ \ldots,\ is \\
&\qquad q = (s - 1 + \frac{q(q+1)}{2})/m
\end{aligned}$$

$$
\begin{aligned}
& l \;=\; s \;-\; qm \;+\; \frac{q(q+1)}{2} \\
& For \qquad k \;=\; 1,\; 2,\; \ldots,\; n \\
& \qquad\qquad b_k \;=\; d_{l+1\;k} \;-\; f_{l+1\;k} \\
& \qquad\qquad p \;=\; min\;(\;r,\; q+1\;) \\
& \qquad\qquad If \qquad k \;\leq\; p \qquad Then \\
& \qquad\qquad\qquad\qquad b_k \;=\; 8\; b_k x_s \\
& \qquad\qquad Else \\
& \qquad\qquad\qquad\qquad t \;=\; mq \;-\; \frac{q(q+1)}{2} \;-\; 1 \\
& \qquad\qquad\qquad\qquad b_k \;=\; 8\; b_k\;(\;x_s \;-\; x_{k+t}) \\
& \qquad\qquad End\;\; If \\
& \qquad End\;\; For \\
& \qquad \frac{\partial \phi}{\partial x_s} \;=\; \sum_{k=1}^{n} b_k \\
& End\;\; For
\end{aligned}
$$

Example 3.4.5

As an example illustrating translation and rotation for Method 3.4.2 and Method 3.4.3, let

$$X^T \;=\; \begin{bmatrix} 2 & 6 & 4 \\ 3 & 5 & 4 \\ 2 & 4 & 5 \end{bmatrix}.$$

First transform $\mathbf{p}_1 \;=\; (2,\; 3,\; 2)$ to the origin then

$$X'^T \;=\; \begin{bmatrix} 0 & 4 & 2 \\ 0 & 2 & 1 \\ 0 & 2 & 3 \end{bmatrix}.$$

Figure 3.4.1: Transform the point $\mathbf{p}_1$ to the origin in order to reduce the number of variables from rn to $r(n-1)$.

Rotating $\mathbf{p}_2$ around the origin (see Figures 3.4.4 and 3.4.5) gives

$$X^{''T} = \begin{bmatrix} 0 & 4.99 & 3.26 \\ 0 & 0 & 1.58 \\ 0 & 0 & 1.41 \end{bmatrix}$$

whilst a second rotation around the x–axis of $\mathbf{p}_3$ located in the plane of y,z–axes (see Figures 3.4.5 and 3.4.6) gives

$$X^{'''T} = \begin{bmatrix} 0 & 4.99 & 3.26 \\ 0 & 0 & 1.83 \\ 0 & 0 & 0 \end{bmatrix}.$$

Figure 3.4.2: The location for each point after the translation.

An important consideration is the choice of the initial matrix X. In the following a procedure is given for finding a suitable initial matrix X. Let $-F$ be the given distance matrix, and let $r^{(0)}$ be the estimated rank. The initial matrix X for Method 3.4.2 can be calculated using Theorem 3.2.3 as follows:

Define the elements of A from F by

$$a_{ij} = -\tfrac{1}{2}[\, f_{1i} + f_{1j} - f_{ij}] \quad (2 \leq i, j \leq n). \tag{3.4.27}$$

Figure 3.4.3: Rotate the point $\mathbf{p}_2$ around the origin so that it is located on the x-axis. This removes $r-1$ variables.

Figure 3.4.4: The location for each point after the first rotation.

Figure 3.4.5: Rotate the point $\mathbf{p}_3$ around the x–axis so that it is located on the x,y–axis. This removes $r-1$ variables.

Figure 3.4.6: The final location for each point with variables reduced from 9 variables to only 3 variables.

Consider the spectral decomposition

$$A \;=\; U\;\Lambda\;U^T,$$

then the initial matrix X for the unconstrained Method 3.4.2 is given by

$$X \;=\; U\;\Lambda_r^{1/2} \tag{3.4.28}$$

where $\Lambda_r \;=\; diag\;[\lambda_1,\;\lambda_2,\ldots,\;\lambda_r]$, are the r largest eigenvalues in Λ.

The above equations can be used to form an initial matrix X for Method 3.4.1 and Method 3.4.3. However, the initial matrix X can be any independent vectors. The independence is important because if one of the vectors is dependent on the other vectors, error will occur in the minimizer of $\phi(X)$, and D will be embeddable in $\Re^{r-1}$ when it should be irreducibly embeddable in $\Re^r$. For example if

$$X^T \;=\; [\mathbf{p}_1,\;\mathbf{p}_2,\;\mathbf{p}_3]$$

and D^* is irreducibly embeddable in $\Re^2$, choose $\mathbf{p}_3$ such that it is dependent on $\mathbf{p}_2$ and $\mathbf{p}_1$ then the resultant matrix will be embeddable in $\Re^1$ which is not correct (see Figure 3.4.7–8).

Another important consideration for the unconstrained method is how the integer $r^* \;=\; rank(D_1^*) \;=\; X^*$ (D_1 given in (3.2.10)) can be identified correctly. Since r^* is not known in advance it is necessary to estimate it by an integer denoted by $r^{(k)}$. Any change to $r^{(k)}$ causes a change to $\phi(X)$, and the number of variables in $\phi(X)$. It is important to consider the effect of making a fixed incorrect estimate r to r^*. If $r^{(k)} \;<\; r^*$ then the methods described so far converges satisfactorily and ultimately at a superlinear rate of convergence to a minimizer of $\phi(X)$. Since r is too small the minimizer of $\phi(X)$ is not a solution of (3.4.11), however the matrix $-\,D^{(k)}$ is a Euclidean distance matrix but it is not the nearest Euclidean distance matrix to $F^{(0)}$. On the other hand if $r^{(k)} \;>\; r^*$ then the methods converges to the minimizer of $\phi(X)$, which is the solution of (3.4.11) but the rate of convergence is very slow because the number of variables in $\phi(X)$ are increased. It seems to be difficult to find an estimate of the rank r^* from the structure of the distance matrix $-\,F$.

Figure 3.4.7: Illustrates the dependence of $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ which makes D embeddable in $\Re^1$.

Figure 3.4.8: Illustrates the independence of $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ which makes D irreducibly embeddable in $\Re^2$.

A strategy has been selected to estimate r^*. The above observations suggest we should choose $r^{(0)}$ arbitrarily as a small integer. Subsequently $r^{(k)}$ is increased by one and $\phi(X)$ is minimized by the methods described above for each $r^{(k)}$. Let $-D^{(k)}$ denote the resulting Euclidean distance matrix. If $D^{(k)} = D^{(k+1)}$ then the algorithm terminates. Otherwise $r^{(k)}$ is increased by one which adds $n - 1$ new variables to problem (3.4.11), and it is necessary to add a new vector to the matrix X. This vector is determined randomly. It is important that the independence mentioned above is satisfied by the new vector. In Chapter 4 an alternative approach is studied in which the projection method is used to give a better estimate of the new vector. After adding one to $r^{(k)}$ the problem (3.4.11) is minimized again using one of the unconstrained methods and the above procedures are repeated. As $r^{(k)}$ can only be increased, the correct value r^* will be identified after a few repetitions of the iterative process.

Finally, an advantage of unconstrained method is that it allows the spatial dimensions to be chosen by the user. This is useful where the rank is already known. For example if the distance matrix are distances between cities then the dimension will be no more than $r = 2$. Likewize if the distance matrix are distances between atoms in a molecule or stars in space, then the maximum dimension is $r = 3$.

The disadvantage of Methods 3.4.1–3 is if the rank is unknown. The algorithm may have to be repeated many times before we find the correct rank. This makes convergence very slow. Therefore in Chapter 4 new methods will be introduced for solving problem (3.4.11) (or equivalently problem (3.3.3)) which avoid this disadvantage.

3.5 The Elegant algorithm

In the previous sections a complete description of the projection method and unconstrained methods have been given. The projection method along with Method 3.4.2 described in the previous section are used to construct the methods in Chapter 4.

In this section another method for solving problem (3.3.3) is given. The Elegant algorithm is described by Takane [1977] using a method related to the alternating least squares approach, later modified by Browne [1987].

Their methods are based on computing the gradient of

$$\phi \;=\; \|F \;-\; D\|. \tag{3.5.1}$$

It is then found that if

$$P \;=\; I \;-\; \frac{\mathbf{e}\mathbf{e}^T}{n}$$

and

$$S(F) \;=\; diag\,[\sum_{j=1}^{n} f_{1j}, \quad \ldots, \sum_{j=1}^{n} f_{nj}]$$

then

$$\frac{\partial \phi}{\partial D} \;=\; (F \;-\; D \;-\; S(F) \;+\; S(D))\; PDP \;=\; 0$$

which is a necessary condition for minimality.

Let $-F$ be a distance matrix, then this matrix can be transformed into a $n \times n$ matrix

$$A \;=\; -\tfrac{1}{2}\; PFP.$$

Now let $\Lambda_r \;=\; diag\,[\lambda_1,\; \lambda_2,\; \ldots,\; \lambda_r]$ be the diagonal matrix formed from the r positive eigenvalues of A, and U_r the $n \times r$ matrix of corresponding eigenvectors, and define

$$Y \;=\; U_r \Lambda_r^{1/2}. \tag{3.5.2}$$

Now the Elegant algorithm can be expressed as

Algorithm 3.5.1 (*Elegant algorithm*)

Given any distance matrix $-F \;\in\; \Re^{n \times n}$, choose $\alpha, \quad 0 < \alpha < 1, \quad$ let $F^{(0)} \;=\; F$

$$For \quad k \;=\; 1,\; 2, \quad \ldots$$

$$F^{(k+1)} \;=\; \alpha(F \;-\; S(F) \;+\; S(F^{(k)})) \;+\; (1-\alpha) Y^{(k)} Y^{(k)T}$$

where Y is given in (3.5.2).

To improve the rate of convergence, Browne [1987] added a penalty function to (3.5.1) and introduced an intermediate Newton–Raphson step, he called this method the Newton–Raphson method.

	NRA		EA		PA	
n	NI	CPU	NI	CPU	NI	CPU
4	18	0.12	21	0.17	26	0.16
8	31	0.25	17	0.29	19	0.22
16	71	1.94	16	1.62	30	0.78
32	175	32.0(0.05)	14	17.64(2.3)	36	5.01(0.08)
64	367	2189.4(5.23)	14	233.23(17.4)	56	53.46(2.9)
100	708	3095.0(540)	13	948.11(529.8)	68	241.56(16.0)

Table 3.6.1: Numerical comparisons between the three projection algorithms.

PA: Projection Algorithm 3.3.4.
EA: Elegant Algorithm 3.5.1.
NRA: Newton–Raphson algorithm.
NI: Average number of iteration.
CPU: Average CPU time in seconds.
(): Standard deviation in CPU time.

3.6 Numerical results

In this section numerical examples are given for unconstrained methods. First an example of order 4 is given in some detail and then another six examples are given showing how the unconstrained methods behave.

However in the first part comparisons between the projection algorithm and methods of Section 3.5 are considered. In Chapter 4 larger examples for both Algorithm 3.3.4 and Method 3.4.2 are given.

Table 3.6.1 given by Glunt et. al. [1990] compares the three algorithms: the projection Algorithm 3.3.2, the Elegant Algorithm 3.5.1 and the Newton–Raphson algorithm. In Elegant Algorithm $\alpha = \frac{1}{2}$ as long as $F^{(k)}$ is monotonically decreasing and then reducing α by a factor of $\frac{1}{2}$ at non–decreasing. All three algorithm converge to essentially the same values.

The matrices in Table 3.6.1 were randomly generated distance matrices. Table 3.6.1 shows that the projection method consumes less CPU time than the other methods. Therefore it will be used in Chapter 4.

In the rest of this section the numerical result for the quasi–Newton methods of Section 3.4 are discussed. A Fortran program has been written to solve problem (3.4.11) on a Sun computer. The results in this section are accurate to 7–8 decimal places in the distance between the given matrix and the Euclidean distance matrix.

Methods	Initial X	Optimal X	no. of variables	no. of line search	Distance
3.4.1	1 0	1.8104 0.5052	8	15	0
	0 1	0.4601 0.9253			
	0 0	0.0400 -0.4253			
	0 0	-1.3105 -0.0052			
3.4.2	1 0	-1.3682 -0.3581	6	19	0
	0 1	-1.7261 1.0100			
	0 0	-3.0943 0.6523			
3.4.3	1	-1.4142	5	22	0
	0 1	-1.4142 1.4142			
	0 0	-2.8284 1.4142			

Table 3.6.2: Results from example (3.6.1).

In the following an example is given in which $-F$ is a 4×4 Euclidean distance matrix given by

$$-F = \begin{bmatrix} 0 & 2 & 4 & 10 \\ 2 & 0 & 2 & 4 \\ 4 & 2 & 0 & 2 \\ 10 & 4 & 2 & 0 \end{bmatrix}. \tag{3.6.1}$$

This matrix is embedded in $\Re^2$ and F_1 is of rank 2 see Figure 3.6.1. Table 3.6.2 shows the results from the three methods of Section 3.4. They confirm that the programs work since the three distances are zero (see Table 3.6.2). In Table 3.6.2, Method 3.4.1 gives the optimal solution X which implies that the matrix D is the optimal matrix and it turns out to be the same as the matrix $-F$ in (3.6.1) since $-F$ is already a Euclidean distance matrix. This is also true for Methods 3.4.2 and 3.4.3. The distances in Figure 3.6.1 are squared before being stored in the matrix $-F$.

Another thing which is worth noting is that the matrix X is not unique. For example, X for Method 3.4.3 in Table 3.6.2

$$X^T = \begin{bmatrix} -1.41421 & -1.4142 & -2.8284 \\ & 1.4142 & 1.4142 \end{bmatrix}$$

Figure 3.6.1: The Euclidean distance matrix represented in $\Re^2$.

can be replaced by the matrix

$$X^T = \begin{bmatrix} 1.41421 & 1.4142 & 2.8284 \\ & 1.4142 & 1.4142 \end{bmatrix}$$

which gives the same result.

Finally the initial matrix X is chosen to be $[\mathbf{e}_1,\ \mathbf{e}_2,\ 0]$, but using equations (3.4.27–28) to reform the given distance matrix $-F$ to an initial matrix X for Method 3.4.2 reduces the number of line searches to zero. The superlinear convergence turns out to be true in this example.

In Table 3.6.3 six examples are chosen randomly to show how the three methods behave.

The initial matrix

$$X^T = [\mathbf{e}_1,\ \mathbf{e}_2,\ \ldots,\ \mathbf{e}_r,\ 0,\ \ldots,\ 0]$$

is used for all three methods. Probably the best method when n is large $(n \geq 9)$ is Method 3.4.2 because it has the least number of line searches. On the other hand, Method 3.4.1 is better when n is sufficiently small $(n < 9)$.

Method 3.4.3 is the worst because it takes the greatest number of line searches. Also, it fails with certain initial matrices and different initial matrix is needed to solve the problem. Specially when the given matrix is already a Euclidean distance matrix (see Table 3.6.3 $n = 11$).

In the first example ($n = 11$) the given matrix is a Euclidean distance matrix and Method 3.4.3 fails to find the optimal solution for many given initial vectors when $r > 5$ (marked by (*)). In the other methods sometimes the above initial matrix does not find the optimal solution and a different initial matrix is used (marked by (**)). Perhaps, the reason behind this is that Method 3.4.1 and Method 3.4.2 have more freedom to choose the optimal vectors $\mathbf{p}_i$'s near the initial vectors(The optimal vectors $\mathbf{p}_i$'s are not unique). In Method 3.4.3 because of rotation, the initial vectors have to search further for the optimal vectors because they are more specific. One can see this from Table 3.6.3, when r is small, where very few rotations occur, the number of line searches is almost similar with the other methods. When r is bigger the difference in the number of line searches becomes greater. It is clear that Method 3.4.2 is better than 3.4.1 because its number of variables is less than those of Method 3.4.1 by r variables.

In Table 3.6.3 there are two columns for Method 3.4.2. The first column for 3.4.2 has the above initial matrix as initial data every time we increase $r^{(k)}$. This make it comparable with the other methods. In the second column for Method 3.4.2 the initial matrix is reformed from F using equations (3.4.27–28), updating the initial matrix every time we increase $r^{(k)}$ using the previous result. This gives faster convergence for the method.

n	$r^{(k)}$	NL 3.4.1	NL 3.4.2	NL 3.4.2(+)	NL 3.4.3	NIP method	Dist–ance
11	1	23	23	14	23		25.573
	2	40	38	32	33		16.449
	3	49	50	59	63		10.090
	4	63	71	48	86		7.026
	5	80	84	50	198		5.289
	6	91	79	62	109(*)		3.967
	7	90	77	69	81(*)		2.638
	8	92	78	52	83(*)		1.684
	9	70	79	70	99(*)		0.961
	10*	66	72	58	72(*)	3	0
10	1	30	25	17	26		149.63
	2	41	37	31	43		71.407
	3*	45	41	37	47		62.3131
	4	52	51	38	65	47	62.3131
10	1	21	21	17	21(**)		856.302
	2	39	38	26	43		785.213
	3*	42	55	13	66		785.190
	4	55	62	16	79	56	785.190
10	1	29	30(**)	16	43		8107.56
	2	66(**)	48	40	62		6767.53
	3	53	59	47	76		6061.81
	4*	179(**)	77	26	103		5904.95
	5	84	113	52	141	64	5904.95
10	1*	29	35	5	35		990.88
	2	37	48	18	37	125	990.88
10	1	25	26	19	27(**)		34.021
	2	33	30	27	30		26.021
	3	43	40	34	41		24.172
	4*	46	48	40	62		23.973
	5	55	54	45	82	30	23.973

Table 3.6.3: Numerical comparisons between unconstrained methods and the projection algorithm.

(+): Using equations (3.4.27–28). Then updating the initial matrix every time we increase $r^{(k)}$.
NL: Number of line searches.
NIP: Number of iterations for projection method.