UNIVERSITÀ DI PISA

IN SUPREMÆ DIGNITATIS · 1343 ·

**Dipartimento di Matematica**
**Corso di Laurea Triennale in Matematica**

# Algorithms for manifold learning and analysis of their stability

**Supervisor:**
**Dott. Dario Trevisan**

**Presented by:**
**Federico Lazzeri**

**Academic Year 2019/2020**

# Contents

**1 Introduction** **5**

**2 Preliminaries** **7**

2.1 Schatten norms . . . . . . . . . . 7

2.1.1 The Singular Value Decomposition (SVD) . . . . . . . . . 7

2.1.2 Min-max theorem . . . . . . . . . . 8

2.1.3 Schatten norms . . . . . . . . . . 9

2.2 Moore-Penrose Pseudo-Inverse . . . . . . . . . . 10

**3 Stability results for manifold learning** **13**

3.1 Procrustes . . . . . . . . . . 13

3.2 Classical Scaling . . . . . . . . . . 20

3.2.1 Euclidean Metric and Gram Matrices . . . . . . . . . . 21

3.2.2 CMDS Algorithm and perturbation bounds . . . . . . . . 24

3.2.3 Computational cost and the Lanczos algorithm . . . . . . 28

3.3 Trilateration . . . . . . . . . . 29

3.4 Isomap . . . . . . . . . . 36

**4 Further experiments on graphs** **43**

# Chapter 1

# Introduction

Manifold learning is an approach to non-linear dimensionality reduction. Algorithms for this task are based on the idea that the dimensionality of many data sets is only artificially high. High-dimensional data sets can be very difficult to visualize. While data in two or three dimensions can be plotted to show the inherent structure of the data, equivalent high-dimensional plots are much less intuitive. To aid visualization of the structure of a dataset, the dimension must be reduced in some way. The simplest way to accomplish this dimensionality reduction is by taking a random projection of the data. Though this allows some degree of visualization of the data structure, the randomness of the choice leaves much to be desired. In a random projection, it is likely that the more interesting structure within the data will be lost. To address this concern, a number of supervised and unsupervised linear dimensionality reduction frameworks have been designed. One of this is the multidimensional scaling.

Multidimensional scaling (MDS) can be defined as the task of embedding an itemset as points in a (typically) Euclidean space based on some dissimilarity information between the items in the set. Since its inception, MDS has been one of the main tasks in the general area of multivariate analysis, a.k.a. unsupervised learning. One of the main methods for MDS is called classical scaling, which consists in first double-centering the dissimilarity matrix and then performing an eigen-decomposition of the obtained matrix. This is arguably still the most popular variant, even today, decades after its introduction at the dawn of this literature. Despite its wide use, its perturbative properties remain little understood. In the Section 3.2 we analyze the method and some of these perturbative properties, following [1].

In order to quantify the difference between two configuration of points we introduce the orthogonal procrustes problem (Section 3.1). This is a method which can be used to find out the optimal rotation and/or reflection (i.e., the optimal orthogonal linear transformation) that carries an itemset to the best approximation of another one.

In Section 3.3, it is described the trilateration algorithm, a procedure that can find a point using the positions of other points (called landmarks) and their distances from it. This technique allows us to simplify the multidimensional scaling. Indeed through the trilateration method, we can reconstruct an entire configuration of points simply knowing few of them and the distances between them and the ones that we want to find. Therefore, instead of applying MDS on the whole itemset, we can do it just with a small subset and then use the trilateration to find the others. This variation of multidimensional scaling is called landmark multidimensional scaling (LMDS).

The last algorithm that we consider is called Isomap (Section 3.4). This algorithm tries to better represent the itemset embedded by using a distance that approximates the one on the manifold. Fixed a radius $r$, it makes a weighted distance graph with points as nodes; if the Euclidean distance between two points is smaller then $r$, it links the two corresponding nodes with an edge with the distance as weight. Then it is calculated the shortest-path distances graph. Finally, the distance matrix containing the values of the edges of this last graph is passed to the classical MDS, that computes the representation.

The work is organized in this way: in chapter 2 there are some preliminaries on matrix algebra; in chapter 3, as mentioned above, there are the algorithms and perturbation bounds for them; in chapter 4 there are other experiments made on graphs.

# Chapter 2

# Preliminaries

## 2.1 Schatten norms

### 2.1.1 The Singular Value Decomposition (SVD)

**Theorem 1.** *Let $A \in \mathbb{C}^{m \times n}$ have rank $r$. Then there are unitary matrices $U$ and $V$ such that*

$$U^H AV = \begin{pmatrix} \Sigma_+ & 0 \\ 0 & 0 \end{pmatrix},$$

*where $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$.*

*Proof.* Let the eigenvalues of $A^H A$ be $\sigma_1^2 \geq \cdots \geq \sigma_r^2 > 0 = \sigma_{r+1}^2 = \cdots = \sigma_n^2$. Let $V = (V_1\ V_2)$, $V_1 \in \mathbb{C}^{n \times r}$ be a unitary matrix formed from the corresponding eigenvectors of $A^H A$. Then

$$V^H A^H AV = \begin{pmatrix} \Sigma_+^2 & 0 \\ 0 & 0 \end{pmatrix},$$

where $\Sigma_+$ is defined above. Thus we have

$$V_1^H A^H AV_1 = \Sigma_+^2, \quad V_2^H A^H AV_2 = 0, \tag{2.1.1}$$

and from the second of these relations we conclude that

$$AV_2 = 0. \tag{2.1.2}$$

Now let

$$U_1 = AV_1 \Sigma_+^{-1}. \tag{2.1.3}$$

Then from (2.1.1) we have $U_1^H U_1 = I$. Choose $U_2$ so that $(U_1\ U_2)$ is unitary. Then from (2.1.1)-(2.1.3) we get

$$U^H AV = \begin{pmatrix} U_1^H AV_1 & U_1^H AV_2 \\ U_2^H AV_1 & U_2^H AV_2 \end{pmatrix} = \begin{pmatrix} \Sigma_+ & 0 \\ 0 & 0 \end{pmatrix}.$$

□

### 2.1.2 Min-max theorem

Let $A \in \mathbb{C}^{n\times n}$ Hermitian. We consider the Rayleigh-Ritz quotient
$R_A \colon \mathbb{C}^n \setminus \{0\} \to \mathbb{R}$ defined by

$$R_A(x) = \frac{(Ax, x)}{(x, x)}.$$

**Theorem 2** (Min-max theorem)**.** *Let $A \in \mathbb{C}^{n\times n}$ Hermitian with eigenvalues $\nu_1 \geq \cdots \geq \nu_n$ then*

$$\nu_k = \min_U \{\max_x \{R_A(x) | x \in U, x \neq 0\} | \dim(U) = n - k + 1\}$$

*and*

$$\nu_k = \max_U \{\min_x \{R_A(x) | x \in U, x \neq 0\} | \dim(U) = k\}.$$

*Proof.* Since the matrix $A$ is Hermitian, it is diagonalizable and we can find an orthonormal basis of eigenvectors $\{u_1, \ldots, u_n\}$, respectively corresponding to the eigenvalues $\{\nu_1, \ldots, \nu_n\}$.

If $U$ is a subspace of dimension $n - k + 1$ then its intersection with the subspace $\mathrm{span}\{u_1, \ldots, u_k\}$ is not zero and hence there exists a vector $v \neq 0$ in this intersection that we can write as

$$v = \sum_{i=1}^{k} \alpha_i u_i$$

and whose Rayleigh quotient is

$$R_A(v) = \frac{\sum_{i=1}^k \nu_i \alpha_i^2}{\sum_{i=1}^k \alpha_i^2} \geq \nu_k$$

(as all $\nu_i \geq \nu_k$ for $i = 1, \ldots, k$) and hence

$$\max\{R_A(x) | x \in U, x \neq 0\} \geq \nu_k.$$

Since this is true for all $U$, we can conclude that

$$\min\{\max\{R_A(x) | x \in U, x \neq 0\} | \dim(U) = n - k + 1\} \geq \nu_k.$$

To establish the other inequality, we can choose the specific (n-k+1)-dimensional subspace $V = span\{u_k, \dots, u_n\}$, for which

$$\max\{R_A(x) | x \in V, x \neq 0\} \leq \nu_k.$$

because $\nu_k$ is the largest eigenvalue in $V$. Therefore, also

$$\min\{\max\{R_A(x) | x \in U, x \neq 0\} | \dim(U) = n-k+1\} \leq \nu_k.$$

Similarly we can achieve the second part of the thesis. □

### 2.1.3 Schatten norms

Let $A \in \mathbb{R}^{m \times d}$ with singular values $\nu_1(A) \geq \nu_2(A) \geq \dots \geq \nu_d(A)$.

Let $p > 0$; we define $\|\cdot\|_p$ the following Schatten quasi-norm[1]

$$\|A\|_p \equiv (\sum_{i=1}^{d} \nu_i(A)^p)^{1/p}$$

which is a norm if $p \in [1, \infty)$. We also define $\|\cdot\|_\infty$ as the usual operator norm: $\|A\|_\infty = \max_{\|x\|_\infty = 1} \|Ax\|_\infty$; we denote it also by $\|\cdot\|$.

The Schatten quasi-norms are unitarily invariant and they satisfy the following inequality

$$\|AB\|_p \leq \|A\|_\infty \|B\|_p \tag{2.1.4}$$

for all $A, B$ with compatible sizes.

If $p \geq 1$, the norm $\|\cdot\|_p$ is submultiplicative (it follows by (2.1.4), noting that $\|A\|_\infty \leq \|A\|_p \ \forall p \geq 1$)

Moreover, $\|A\|_p = \|A^T\|_p$ and $\|A\|_p = \|A^T A\|_{p/2}^{1/2} = \|AA^T\|_{p/2}^{p/2}$ due to the fact that

$$\|A\|_p^p = \sum_j \nu_j(A)^p = \sum_j \nu_j(A^T A)^{p/2} = \|A^T A\|_{p/2}^{p/2}.$$

If $A$ and $B$ are positive semidefinite and they satisfy $A \preceq B$, where $\preceq$ denotes the Loewner order[2], then $\|A\|_p \leq \|B\|_p$. In fact, using the variational principle of eigenvalues (min-max Courant-Fischer theorem), for all $j$ we have

$$\begin{aligned}\nu_j(A) &= \min_{V:\, dim(V)=n-j+1} \max_{v \in V, \|v\|=1} v^T A v \\ &\leq \min_{V:\, dim(V)=n-j+1} \max_{v \in V, \|v\|=1} v^T B v = \nu_j(B).\end{aligned}$$

[1] a quasi-norm satisfies the norm axioms, except that the triangle inequality is replaced by $\|x + y\| \leq K(|x\| + \|y\|)$, $K \geq 1$

[2] $A \preceq B$ if $B - A$ is positive semidefinite

## 2.2 Moore-Penrose Pseudo-Inverse

Let $A \in \mathbb{R}^{m\times k}$ with $m \geq k$ and let $A = UDV^T$ be a singular value decomposition with $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{k\times k}$ orthogonal and $D \in \mathbb{R}^{k\times k}$ diagonal with diagonal entries $\nu_1 \geq \cdots \geq \nu_l > 0 = \cdots = 0$. The pseudo-inverse of $A$ is $A^\ddagger = VD^\ddagger U^T$ where $D^\ddagger = \mathrm{diag}(\nu_1^{-1}, \dots, \nu_l^{-1}, 0, \dots, 0)$. If the matrix $A$ is tall and full rank, then $A^\ddagger = (A^TA)^{-1}A^T$. In particular, if a matrix is square and non-singular, its pseudo-inverse coincides with its inverse.

**Theorem 3** (Wedin). *Let $A$ be an $m \times n$ matrix with $m \geq n$ and $\tilde{A} = A + E$ denote a perturbaiton of $A$. The error $\tilde{A}^\ddagger - A^\ddagger$ has the following bound:*

$$\|\tilde{A}^\ddagger - A^\ddagger\| \leq 3\max\{\|A^\ddagger\|_2^2, \|\tilde{A}^\ddagger\|_2^2\}\|\tilde{A}\| \tag{2.2.1}$$

*Proof.* Let $P, R$ be respectively the projection onto the column space and onto the row space of $A$. The complementary projectors will be denoted by $P_\perp$ and $R_\perp$. Moreover, consider $\tilde{P}, \tilde{R}, \tilde{P}_\perp$ and $\tilde{R}_\perp$ as the analogous matrices for $\tilde{A}$. Let $E = \tilde{A} - A$. By replacing all quantities by their definitions in terms of $A, \tilde{A}, A^\ddagger, \tilde{A}^\ddagger$ and symplifying, we have

$$\tilde{A}^\ddagger - A^\ddagger = -\tilde{A}^\ddagger E A^\ddagger + \tilde{A}^\ddagger P_\perp - \tilde{R}_\perp A^\ddagger,$$

$$\tilde{A}^\ddagger - A^\ddagger = -\tilde{A}^\ddagger \tilde{P} E R A^\ddagger + \tilde{A}^\ddagger \tilde{P} P_\perp - \tilde{R}_\perp R A^\ddagger,$$

and

$$\tilde{A}^\ddagger - A^\ddagger = -\tilde{A}^\ddagger \tilde{P} E R A^\ddagger + (\tilde{A}^H\tilde{A})^\ddagger \tilde{R} E^H P_\perp + \tilde{R}_\perp E^H P (AA^H)^\ddagger.$$

Taking norms in the last equation we achieve the thesis. □

**Theorem 4** (Mirsky). *Let $X$ and $\tilde{X}$ be matrices of the same dimensions with singular values*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p,$$

$$\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_p.$$

*Then for any $p \geq 1$,*

$$\|\mathrm{diag}(\tilde{\sigma}_i - \sigma_i)\|_p \leq \|\tilde{X} - X\|_p. \tag{2.2.2}$$

See ([3],Thm III.3.8) for a proof.

**Lemma 1.** *Let $A$ be a tall full rank matrix. Then $A^\ddagger$ is nonsingular and for all $B$ of compatible size*

$$\|B\|_p \leq \|A^\ddagger\|_\infty \|AB\|_p.$$

*Proof.* Since $A$ is tall and full rank, $A^{\ddagger}A = I$, so that

$$\|B\|_p = \|A^{\ddagger}AB\|_p \leq \|A^{\ddagger}\|_{\infty}\|AB\|_p$$

by (2.1.4). □

**Lemma 2.** *Let $A$ and $B$ be matrices of same size. Then, for $p \in \{2, \infty\}$*

$$\|B^{\ddagger} - A^{\ddagger}\|_p \leq \frac{\sqrt{2}\|A^{\ddagger}\|^2\|B-A\|_p}{(1 - \|A^{\ddagger}\|\|B-A\|)_+^2}.$$

*Proof.* From the Theorem of Wedin, we have

$$\|B^{\dagger} - A^{\dagger}\|_p \leq \sqrt{2}(\|B^{\ddagger}\| \vee \|A^{\ddagger}\|)^2\|B-A\|_p, \quad p \in \{2, \infty\}. \tag{2.2.3}$$

Assuming $B$ has exactly $k$ nonzero singular values and using Mirsky's inequality, we have

$$\|B^{\ddagger}\|^{-1} = \nu_k(B) \geq (\nu_k(A) - \|B-A\|)_+ \geq (\|A^{\ddagger}\|^{-1} - \|B-A\|)_+. \tag{2.2.4}$$

By combining (2.2.3) and (2.2.4), we get

$$\|B^{\ddagger} - A^{\ddagger}\|_p \leq \sqrt{2}\Big(\|A^{\ddagger}\| \vee \frac{1}{(\|A^{\ddagger}\| - \|B-A\|)_+}\Big)^2\|B-A\|_p,$$

from which the result folllows.

□

**Lemma 3.** *Let $A$ and $B$ be matrices of the same size such that $A^TB = 0$ or $AB^T = 0$. Then*

$$\|A+B\|_p \geq \|A\|_p \vee \|B\|_p.$$

*Proof.* We assume without lost of generality that $A^TB = 0$; then $(A+B)^T(A+B) = A^TA + B^TB$. Due to the fact that $A^TA \preceq A^TA + B^TB$ and the properties listed above,

$$\|A\|_p = \|A^TA\|_{p/2}^{1/2} \leq \|A^TA + B^TB\|_{p/2}^{1/2} = \|A+B\|_p.$$

Similarly,

$$\|B\|_p \leq \|A+B\|_p.$$

□

**Lemma 4.** *For any matrix $A$ and any positive semidefinite matrix $B$, we have*

$$\|A\|_p \leq \|A(B+I)\|_p,$$

*where $I$ denotes the identity matrix with the same dimension as $B$.*

*Proof.*

$$A(B+I)(B+I)^T A^T = A(B^2+2B+I)A^T = AA^T + A(B^2+2B)A^T.$$

Since $A(B^2+2B)A^T \succeq 0$, for all $k$ we have

$$\nu_k(A(B+I)(B+I)^T A^T) \geq \nu_k(AA^T),$$

which then implies that $\nu_k(A(B+I)) \geq \nu_k(A)$ and so the thesis. $\square$

# Chapter 3

# Stability results for manifold learning

We will denote with $\mathcal{O}$ the orthogonal group of matrices in the appropriate Euclidean space (which will be clear from context) and with $\|\cdot\|$, when applied to a vector, the Euclidean norm.

## 3.1 Procrustes

The orthogonal procrustes problem is that of aligning two ordered point sets (of same cardinality) using an orthogonal transformation. In formulas, given two point sets, $x_1, \dots, x_n$ and $y_1, \dots, y_n$ in $\mathbb{R}^d$, the task consists in solving

$$\min_{Q \in \mathcal{O}} \sum_{i=1}^{n} \|y_i - Qx_i\|^2.$$

In matrix form, the problem can be posed as follows. Given matrices $X, Y \in \mathbb{R}^{n \times d}$, solve

$$\min_{Q \in \mathcal{O}} \|Y - XQ\|_2,$$

where $\|\cdot\|_2$ denotes the Frobenius norm. Indeed

$$\|Y - XQ\|_2 = \Big(\sum_{j=1}^{n} \sum_{i=i}^{n} |(Y - XQ)_{j,i}|^2\Big)^{1/2} = \Big(\sum_{i=1}^{n} \|y_i - Q^T x_i\|^2\Big)^{1/2},$$

and the square root is monotone.

We want to minimize

$$\sum_{i=1}^{n} \|y_i - Q^T x_i\|^2 = \operatorname{tr}[(Y - XQ)(Y - XQ)^T] = \operatorname{tr}(YY^T) + \operatorname{tr}(XX^T) - 2\operatorname{tr}(QY^T X),$$

therefore we want to maximize $\mathrm{tr}(QY^TX)$. The problem is solved by choosing $Q = UV^T$, where $U$ and $V$ are $d$-by-$d$ orthogonal matrices obtained by a singular value decomposition of $X^TY = UDV^T$, where $D$ is the diagonal matrix with the singular values on its diagonal. It follows from the next lemma, taking $A = Y^TX$.

**Lemma 5.** *Let $A$ be a $d \times d$ matrix with singular value decomposition $A = VDU^T$ where $V$ and $U$ are orthogonal matrices and $D = diag(\delta_1, \ldots, \delta_d)$, where each $\delta_j \geq 0$. Then for all orthogonal $Q$,*

$$\mathrm{tr}(QA) \leq \mathrm{tr}[(A^TA)^{1/2}]$$

*with equality if $Q = UV^T$.*

*Proof.*

$$\mathrm{tr}(QA) = \mathrm{tr}(QVDU^T) = \mathrm{tr}(U^TQVD) = \mathrm{tr}(ND),$$

where $N = U^TQV$ is an orthogonal matrix, being the product of orthogonal matrices. Now the elements of an orthogonal matrix cannot exceed 1, so that

$$\begin{aligned}\mathrm{tr}(ND) &= \sum_{j=1}^{d} n_{jj}\delta_j \leq \sum_{j=1}^{d} \delta_j = \mathrm{tr}(D) = \mathrm{tr}[(D^TD)^{1/2}] \\ &= \mathrm{tr}[(U^TA^TVV^TAU)^{1/2}] = \mathrm{tr}[(U^TA^TAU)^{1/2}] \\ &= \mathrm{tr}[(AUU^TA^T)] = \mathrm{tr}[(AA^T)^{1/2}] = \mathrm{tr}[(A^TA)^{1/2}].\end{aligned}$$

We have equality when $Q = UV^T$, as

$$\begin{aligned}QA &= UV^TVDU^T = UDU^T = (UDU^TUDU^T)^{1/2} \\ &= (UD^2U^T)^{1/2} = (UDV^TVDU^T)^{1/2} = (A^TA)^{1/2}.\end{aligned}$$

□

**Algorithm 1** Procrustes

```
Input: points sets $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ in $\mathbb{R}^d$
Output: an orthogonal transformation $Q$ of $\mathbb{R}^d$

1: store the point sets in $X = [x_1^T \cdots x_n^T]$ and $Y = [y_1^T \cdots y_n^T]$
2: compute $X^TY$ and its singular value decomposition $UDV^T$
Return: the matrix $Q = UV^T$
```

The following is a code in Matlab:

```
function Q = procrustes(X,Y)
Z = X'*Y;
[U,D,V] = svd(Z);
Q = U*V';
end
```

To show an example, we generated a data set of $n = 1000$ points of $\mathbb{R}^3$ using the matlab command `normrnd` to choose each vector. This command takes in input two matrices, a.k.a. $M$ and $\Sigma$, and returns an array of random numbers chosen from a normal distribution with mean $M$ and standard deviation $\Sigma$; we choose $M = 0 \in \mathbb{R}^{3\times3}$ and $\Sigma = I \in \mathbb{R}^{3\times3}$, where $I$ denotes the identity matrix. Taking the norm of these vectors and multiplying the resulting matrix by a diagonal matrix, we obtaine the ellipsoid stored in the matrix $X \in \mathbb{R}^{n\times d}$. Then we compute $Y$ multiplying $X$ by a random matrix. Finally we apply the algorithm `procrustes` to $X$ and $Y$ obtaining the matrix $Q$, and we plot the three point sets given by the rows of $X, Y$ and $XQ$.

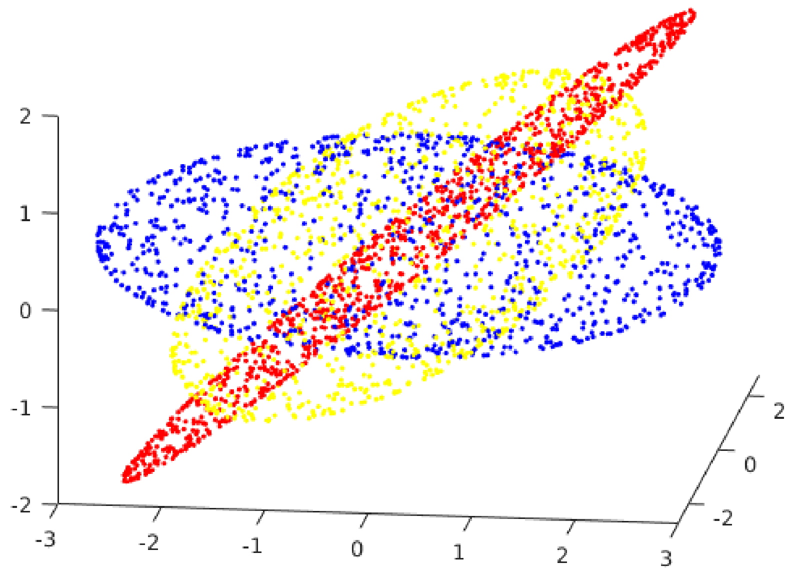


Figure 3.1: In blue the ellipsoid $X$, in red $Y$ and in yellow $XQ$, where $Q$ is obtained applying `procrustes` to $X$ and $Y$.

In [1] it is proved the following perturbation bound for procrustes, where the distance between two configurations of points $Y$ and $XQ$ is bounded in terms of the distance between $XX^T$ and $YY^T$.

**Theorem 5.** *Let $X, Y \in \mathbb{R}^{n\times d}$ be full rank matrices and set $\varepsilon^2 = \|YY^T - XX^T\|_p$, with $p > 0$. Then, we have*

$$\min_{Q\in\mathcal{O}} \|Y - XQ\|_p \leq \begin{cases} \|X^\ddagger\|\varepsilon^2 + ((1-\|X^\ddagger\|^2\varepsilon^2)^{-1/2}\|X^\ddagger\|\varepsilon^2) \wedge (d^{1/2p}\varepsilon) & \textit{if} \|X^\ddagger\|\varepsilon < 1, \\ \|X^\ddagger\|\varepsilon^2 + d^{1/2p}\varepsilon & \textit{otherwise.} \end{cases} \tag{3.1.1}$$

*Consequently, if* $\|X^\ddagger\|\varepsilon \leq \frac{1}{\sqrt{2}}$*, then*

$$\min_{Q\in\mathcal{O}} \|Y - XQ\|_p \leq (1+\sqrt{2})\|X^\ddagger\|\varepsilon^2. \tag{3.1.2}$$

*Proof.* Let $P \in \mathbb{R}^{n\times n}$ be the orthogonal projection onto the column space of $X$, which can be expressed as $P = XX^\ddagger$. Define $Y_1 = PY$ and $Y_2 = (I-P)Y$, and note that $Y = Y_1 + Y_2$ with $Y_2^TY_1 = 0$, and also $Y_2^TX = 0$.

Define $M = X^\ddagger Y \in \mathbb{R}^{d\times d}$, and apply a singular value decomposition to obtain $M = UDV^T$, where $U$ and $V$ are orthogonal matrices of size $d$, and $D$ is diagonal with nonnegative entries. Indeed columns of $U$ span the row space of $X$ and columns of $V$ span the row space of $Y$. Then define $Q = UV^T$, which is orthogonal. We show that the bound (3.1.2) holds for this orthogonal matrix.

We start with the triangle inequality.

$$\|Y - XQ\|_p = \|Y_1 - XQ + Y_2\|_p \leq \|Y_1 - XQ\|_p + \|Y_2\|_p. \tag{3.1.3}$$

Noting that $Y_1 = XX^\ddagger Y = XM$, we have

$$\begin{aligned}\|Y_1 - XQ\|_p = \|XM - XQ\|_p &= \|XUDV^T - XUV^T\|_p \\ &= \|XU(D-I)V^T\|_p \leq \|XU(D-I)\|_p.\end{aligned} \tag{3.1.4}$$

Now by Lemma 4, we have

$$\|XU(D-I)\|_p \leq \|XU(D-I)(D+I)\|_p = \|XU(D^2-I)\|_p. \tag{3.1.5}$$

Now by unitary invariance, we have

$$\|XU(D^2-I)\|_p = \|XU(D^2-I)U^T\|_p = \|XUD^2U^T - XUU^T\|_p = \|XUD^2U^T - X\|_p, \tag{3.1.6}$$

where in the last step we used the fact that columns of $U$ span the row space of $X$ and hence $UU^TX^T = X^T$. Combining (3.1.4), (3.1.5) and (3.1.6), we obtain

$$\begin{aligned}\|Y_1 - XQ\|_p &\leq \|XUD^2U^T - X\|_p \\ &= \|(XMM^T - X)(X^\ddagger X)^T\|_p \\ &\leq \|X^\ddagger\|\|XMM^TX^T - XX^T\|_p \\ &= \|X^\ddagger\|\|Y_1Y_1^T - XX^T\|_p,\end{aligned} \tag{3.1.7}$$

where the first equality holds since $X^\ddagger X = I$, given that $X$ has full column rank.

Coming from the other end, so to speak, we have

$$\begin{aligned}\varepsilon^2 = \|YY^T - XX^T\|_p &= \|Y_1Y_1^T - XX^T + Y_1Y_2^T + Y_2Y_2^T\|_p & (3.1.8)\\ &\geq \|Y_1Y_1^T - XX^T + Y_1Y_2^T\| \vee \|Y_2Y_1^T + Y_2Y_2^T\|_p & (3.1.9)\\ &\geq \|Y_1Y_1^T - XX^T\|_p \vee \|Y_1Y_2^T\| \vee \|Y_2Y_1^T\|_p \vee \|Y_2Y_2^T\|_p, & (3.1.10)\end{aligned}$$

using Lemma 3 thrice, once based on the fact that

$$(Y_1Y_1^T - XX^T + Y_1Y_2^T)^T(Y_2Y_1^T + Y_2Y_2^T) = \underbrace{(Y_1Y_!^T - XX^T + Y_2Y_1^T)Y_2}_{=0}(Y_1^T + Y_2^T) = 0,$$

and then based on the fact that

$$(Y_1Y_1^T - XX^T)(Y_1Y_2^T)^T = \underbrace{(Y_1Y_1^T - XX^T)Y_2}_{=0}Y_1^T = 0,$$

and

$$(Y_2Y_1^T)(Y_2Y_2^T)^T = Y_2\underbrace{Y_1^TY_2}_{=0}Y_2^T.$$

From (3.1.10), we extract the bound $\|Y_1Y_1^T - XX^T\|_p \leq \varepsilon^2$, from which we get (based on the derivations above)

$$\|Y_1 - XQ\|_p \leq \|X^\ddagger\|\varepsilon^2. \qquad (3.1.11)$$

Recalling the inequality (3.1.3), we proceed to bound $\|Y_2\|_p$. From (3.1.10), we extract the bound $\|Y_2Y_2^T\|_p \leq \varepsilon^2$, and combine it with

$$\|Y_2Y_2^T\|_p = \|Y_2\|_{2p}^2 \geq d^{-1/p}\|Y_2\|_p^2,$$

where $d$ is the number of columns and the inequality is Cauchy-Schwarz's, to get

$$\|Y_2\|_p \leq d^{1/2p}\varepsilon.$$

We next derive another upper bound for $\|Y_2\|_p$, for the case that $\|X^\ddagger\|\varepsilon < 1$. Denote by $\lambda_1 \geq \cdots \geq \lambda_d$ the singular values of $X$ and by $\nu_1 \geq \cdots \geq \nu_d$ the singular values of $Y_1$. Given that $X$ has full column rank we have $\lambda_d > 0$ and so $\|X^\ddagger\| = 1/\lambda_d$. Further, by an application of Mirsky's inequality, we have

$$\max_i |\nu_i^2 - \lambda_i^2| \leq \|Y_1Y_1^T - XX^T\| \leq \|Y_1Y_1^T - XX^T\|_p \leq \varepsilon^2,$$

using (3.1.10). Therefore $\nu_d^2 > \lambda_d^2 - \varepsilon^2 > 0$ by our assumption that $\|X^\ddagger\|\varepsilon^2 < 1$, which implies that $Y_1$ has full column rank. Now, by an application of Lemma 1, we obtain

$$\|Y_2\|_p = \|Y_2^T\|_p \leq \|Y_1^\ddagger\|\|Y_1Y_2^T\|_p \leq \varepsilon^2\|Y_1^\ddagger\|, \qquad (3.1.12)$$

where we used (3.1.10) in the last step. Also,

$$\|Y_1^\ddagger\| = \frac{1}{\nu_d} \leq \frac{1}{(\lambda_d^2 - \varepsilon^2)^{1/2}} = \frac{\lambda_d^{-1}}{(1 - \varepsilon^2\lambda_d^{-2})^{1/2}} = \|X^\ddagger\|(1 - \varepsilon^2\|X^\ddagger\|^2)^{-1/2}. \qquad (3.1.13)$$

Combining (3.1.13) and (3.1.12) we obtain

$$\|Y_2\|_p \le \varepsilon^2 \|X^\ddagger\| (1 - \varepsilon^2 \|X^\ddagger\|^2)^{-1/2}, \quad \text{if } \|X^\ddagger\|\varepsilon < 1. \tag{3.1.14}$$

Combining the bounds (3.1.12) with (3.1.14) and (3.1.11) in the inequality (3.1.3), we get (3.1.1). The bound (3.1.2) follows readily from (3.1.1). □

We can verify numerically that the distance between $X$ and $Y$ scales linearly in $\varepsilon^2$. Indeed, given a matrix $X$ and a perturbation $\eta$, we can construct the matrix $Y = X + \eta R$, where $R$ is a randomly generated matrix. The theorem states that we can bound the distance with

$$\begin{aligned}\varepsilon^2 &= \|YY^T - XX^T\|_p = \|(X + \eta R)(X + \eta R)^T - XX^T\|_p \\ &= \|XX^T + \eta(XR^T + RX^T) + \eta^2 RR^T - XX^T\|_p \\ &= \|\eta(XR^T - RX^T) + \eta^2 RR^T\|_p\end{aligned}$$

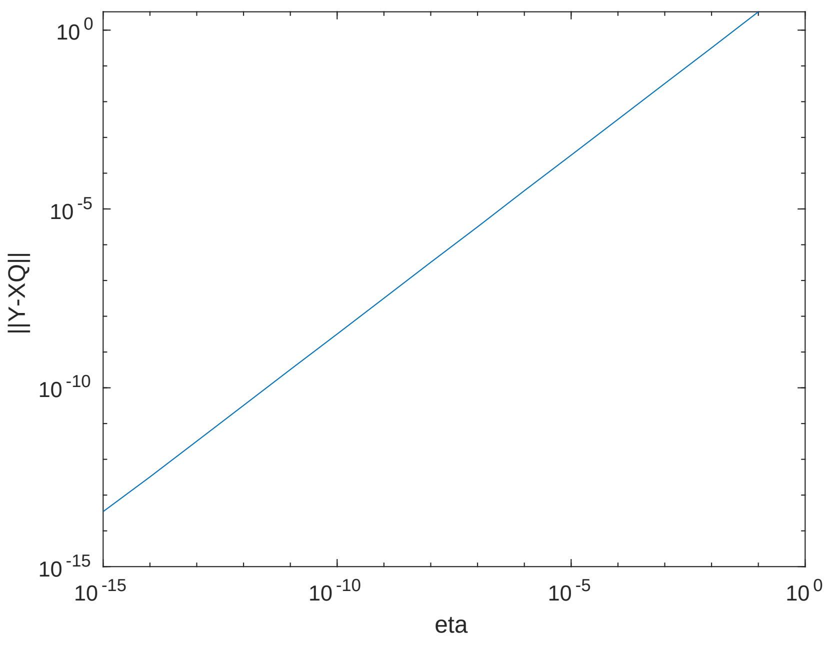


Figure 3.2: Distance between the configurations $Y$ and $XQ$, where $Y = X + \eta R$ and $R$ contains pseudorandom values drawn from the standard uniform distribution on the open interval $(0, 1)$, using the matlab command `rand`.

## 3.2 Classical Scaling

In multidimensional scaling, we are given a matrix, $\Delta = (\Delta_{ij}) \in \mathbb{R}^{n\times n}$, storing the dissimilarities between a set of $n$ items. A square matrix $\Delta$ is called dissimilarity matrix if it is symmetric, $\Delta_{ii} = 0$, and $\Delta_{ij} > 0$, for $i \neq j$ ($\Delta_{ij}$ gives the level of dissimilarity between items $i, j \in \{1, \ldots, n\}$).

In the following, we represent a data set $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^D$ in the matrix form $X = [x_1 \cdots x_n] \in \mathbb{R}^{D\times n}$, where each column of $X$ is a point of $\mathcal{X}$. Thus, the totality of all columns of the data matrix $X$ constitutes the data set

$\mathcal{X}$. For convenience, we shall identify the data matrix $X$ with the data set $\mathcal{X}$.

Let $Dist(i,j) = d_2(x_i, x_j)\ \forall i,j \in \{1,\ldots,n\}$, where $d_2$ denotes the distance induced by the Euclidean norm. Given $Dist$ and a number $d$, the CMDS algorithm returns a matrix $Y \in \mathbb{R}^{n\times d}$ whose rows $\{y_1,\ldots,y_n\}$ are vectors of $\mathbb{R}^d$ such that $d_2(y_i,y_j) \approx d_2(x_i,x_j)\ \forall i,j \in \{1,\ldots,n\}$. More precisely

$$Y = \arg\min_{A\in\mathbb{R}^{n\times d}} \sum_{i,j=1}^{n} (d_2^2(x_i,x_j) - d_2^2(a_i,a_j)),$$

where $a_1,\ldots,a_n$ are the rows of $A$.

## 3.2.1 Euclidean Metric and Gram Matrices

From the viewpoint of geometry, in a Euclidean space, the distance describes the dissimilarity of a pair of points while the inner product describes the similarity. They have a close relationship.

**Definition 1.** An $n\times n$ symmetric matrix $D$ is called a Euclidean distance matrix (or Euclidean metric) if there exists an integer $m > 0$ and a vector set $\mathcal{Z} = \{z_1,\cdots,z_n\} \subset \mathbb{R}^m$ so that

$$D = [D_{ij}] = [d_2(z_i,z_j)]_{i,j=1}^n.$$

The vector set $\mathcal{Z}$ is called a configurative point set (or a configuration) of $D$.

We define the Euclidean square-distance matrix as

$$S = [S_{ij}] = [d_2^2(x_i,x_j)]_{i,j=1}^n.$$

The Gram matrix on the data set $\mathcal{X}$ is defined by

$$G = [G_{ij}] = [\langle x_i,x_j\rangle]_{i,j=1}^n.$$

It is clear that $G$ is a positive semi-definite (psd) matrix. On the other hand, each psd matrix represents a Gram matrix of a certain data set. Indeed, if an $n\times n$ psd matrix $G$ has rank $m$, then it has a Cholesky decomposition

$$G = X^T X, \tag{3.2.1}$$

where $X = [x_1,\cdots,x_n]$ is an $m\times n$ matrix. By (3.2.1), $G$ is the Gram matrix of the data set $\mathcal{X} \subset \mathbb{R}^m$. Therefore, we can identify a Gram matrix with a psd matrix.

We now reveal the relation between the Gram matrix $G$ and the Euclidean distance matrix $D$ of a data set $\mathcal{X}$. Let $x,y \in \mathbb{R}^D$. By the Law of Cosines,

$$d_2(x,y) = \sqrt{\langle x,x\rangle + \langle y,y\rangle - 2\langle x,y\rangle},$$

which yields

$$D_{ij} = d_2(x_i, x_j = \sqrt{G_{ii} + G_{jj} - 2G_{ij}}. \tag{3.2.2}$$

The entries of a Gram matrix $G$ are vector inner products which are not shift invariant. In order to establish a relation between $G$ and $D$, we shift the data $\mathcal{X}$ by its center. Let $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the $\bar{x}$-shift of $\mathcal{X}$ be denoted by

$$\hat{\mathcal{X}} = \{\hat{x}_1, \dots, \hat{x}_n\},$$

where $\hat{x}_i = x_i - \bar{x}$. We call $\hat{\mathcal{X}}$ a *centered data set* and call the corresponding data matrix $\hat{X}$ the *centered data matrix*.

**Definition 2.** For a given data set $\mathcal{X}$, let $\hat{\mathcal{X}}$ be the centered data set of $\mathcal{X}$. Then the Gram matrix of $\hat{\mathcal{X}}$

$$G^c = [\langle \hat{x}_i, \hat{x}_j \rangle]_{i,j=1}^{n} = \hat{X}^T \hat{X}, \tag{3.2.3}$$

is called the centering Gram matrix of $\mathcal{X}$.

It is easy to verify that the centering Gram matrix $G^c$ and the Gram matrix $G$ have the same relationship with the Euclidean distance matrix $D$ descibed in (3.2.2), namely,

$$D_{ij} = \sqrt{G^c_{ii} + G^c_{jj} - 2G^c_{ij}}. \tag{3.2.4}$$

**Definition 3.** Write $e = [1, \dots, 1]^T \in \mathbb{R}^n$, $E = ee^T$, and let $I$ denote the $n \times n$ identity matrix. Then the $n \times n$ matrix $H = I - \frac{1}{n}E$ is called the $n$-centralizing matrix.

If the dimension $n$ is understood, we shall simplify the term "$n$-centralizing" to "centralizing".

**Lemma 6.** *The centralizing matrix $H$ has the following properties.*

1. *$H^2 = H$;*
2. *$e^T H = He = 0$;*
3. *$\mathcal{X}$ is a centered data set $\iff$ $XH = X$;*
4. *A psd matrix $C$ is a centering Gram matrix $\iff$ $HCH = C$.*

*Proof.* Since $e^T e = n$, we have $Ee = ne$, so that

$$H^2 = \Big(I - \frac{1}{n}E\Big)^2 = I - \frac{2}{n}E + \frac{1}{n^2}Eee^T = I - \frac{1}{n}E = H,$$

which yields (1). Furthermore, (2) follows from the fact that $Ee = ne$, (3) can be derived from the definition of centered data, and (4) is a direct consequence of (3). □

By applying these properties directly, we have the following

**Lemma 7.** *Let $X$ be a data matrix and $G$ be its Gram matrix. Then the centered data set of $X$ is $XH$, and the centering Gram matrix of $X$ is $G^c = HGH$.*

In general, the *centering matrix* of a symmetric matrix $A$ (not necessary psd) is defined as $A^c = HAH$. It is obvious that a symmetric matrix $S$ is centering if and only if $S = HSH$. The notion of centering symmetric matrix enables us to represent the centering Gram matrix in terms of Euclidean square-distance matrix, reducing the relation (3.2.2) to a very simple form.

**Theorem 6.** *Euclidean square-distance matrix $S$ and the centering Gram matrix $G^c$ of a data set $\mathcal{X}$ have the following relation.*

$$G^c = -\frac{1}{2} S^c.$$

*Proof.* It follows from Lemma 6 that $G^c$ has the property $\sum_{i=1}^n G^c_{ij} = 0$. Hence, the relation in (3.2.4) immediately yields both

$$\sum_{i=1}^n D^2_{ij} = nG^c_{jj} + \sum_{i=1}^n G^c_{ii}$$

and

$$\sum_{j=1}^n D^2_{ij} = nG^c_{ii} + \sum_{j=1}^n G^c_{jj}.$$

Therefore, the $(i,j)$-entry of $S^c$ is given by

$$\begin{aligned}(S^c)_{ij} &= D^2_{ij} - \frac{1}{n}\left(\sum_{i=1}^n D^2_{ij} + \sum_{j=1}^n D^2_{ij} - \frac{1}{n}\sum_{i,j=1}^n D^2_{ij}\right) \\ &= D^2_{ij} - G^c_{ii} - G^c_{jj} \\ &= -2G^c_{ij},\end{aligned}$$

completing the proof of the theorem. $\square$

The following is a consequence of Theorem 6 and (3.2.1).

**Theorem 7.** *Let $A$ be a symmetric matrix. Then*

1. *$A$ is a Gram matrix of a data set $\iff$ $A$ is a psd matrix;*
2. *$A$ is a centering Gram matrix $\iff$ $A$ is a centering psd matrix;*
3. *$A$ is a Euclidean square-distance matrix $\iff$ $-\frac{1}{2}A^c$ is a centering psd matrix.*

*Proof.* The first and the second statements are trivial. We prove the third one. Write $G = -\frac{1}{2}A^c$. If $G$ is a centering psd matrix, by (3.2.3), there is an $m \times n$ centered matrix $V = [v_1 \cdots v_n]$ with $m \leq n$ such that

$$G = V^T V.$$

By Theorem 6, the matrix $A$, defined by $A^c = -2G^c$, is a Euclidean square-distance matrix. On the other hand, if $A$ is the Euclidean square-distance matrix of a data set $\mathcal{X}$, then $G^c = -\frac{1}{2}A^c$ is the centering Gram matrix of $\mathcal{X}$ so that it is a centering psd matrix. □

**Lemma 8.** *Assume that an $n \times n$ matrix $D = [d_{ij}]$ is a Euclidean metric and $S = [d_{ij}^2]$ is the corresponding square-distance matrix. Let $G^c = -\frac{1}{2}S^c$. If the rank of $G^c$ is $r$, then there is an $r$-dimensional centered vector set $\mathcal{X} = \{x_1, \dots, x_n\} \subset \mathbb{R}^r$ such that*

$$d_2(x_i, x_j) = d_{ij}, \quad 1 \leq i, j \leq n. \tag{3.2.5}$$

*Proof.* By Theorem 7, $G^c$ is a centering Gram matrix. Since the rank of $G^c$ is $r$, there exists an $r \times n$ centered data matrix $X$ such that $G^c = X^T X$. Then the centered data set $\mathcal{X}$ satisfies (3.2.5). The lemma is proved. □

### 3.2.2 CMDS Algorithm and perturbation bounds

```
Algorithm 2 Classical Scaling
Input: square-distance matrix $S \in \mathbb{R}^{n\times n}$, embedding dimension $d$
Output: set of points $y_1, \dots, y_n \in \mathbb{R}^d$
1: compute the matrix $G^c = -\frac{1}{2}HSH$
2: let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $G^c$, with corresponding eigenvectors $u_1, \dots, u_n$
3: compute $Y \in \mathbb{R}^{n\times d}$ as $Y = [\sqrt{\lambda_{1,+}}u_1, \dots, \sqrt{\lambda_{d,+}}u_d]$
Return: the row vectors $y_1, \dots, y_n$ di $Y$
```

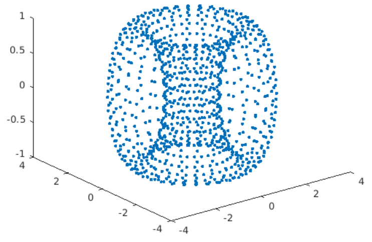
```
function [Y,ev] = cmds(Dist,d)
sz = size(Dist);
n = sz(1);
E = ones(n);
H = eye(n) - E/n;
S = Dist.*Dist;
Gc = -(H*S*H)/2;
[Ud,Dd] = eigs(Gc,d);
Zd = sqrt(max(Dd,zeros(d)));
Y = Ud*Zd;
```

```
ev = diag(Zd);
end
```

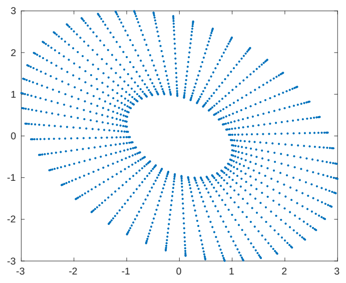
We tried this algorithm with many manifolds. The torus is one of them: we take a parameterization

$$\begin{cases} x(u,v) = (R + r\cos u)\cos v \\ y(u,v) = (R + r\cos u)\sin v \\ z(u,v) = r\sin u \end{cases}$$

with $R = 2, r = 1$ and $u, v$ chosen uniformly from the interval $[0, 2\pi]$. Then we apply `cmds` to scale it in 2 dimension. The distance we use to make the distance matrix is the Euclidean one.



(a) Torus in $\mathbb{R}^3$

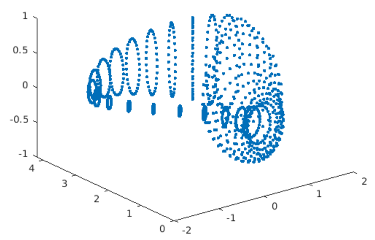


(b) Representation of the torus in $\mathbb{R}^2$, obtained applying `cmds`

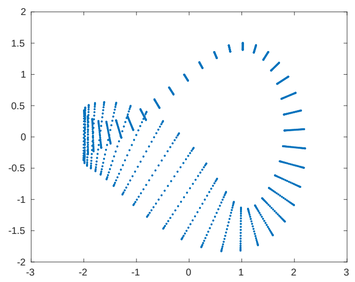
We have done the same thing with the Klein bottle:

$$\begin{cases} x(u,v) = -\frac{2}{15}\cos u(3\cos v - 30\sin u + 90\cos^4 u\sin u - 60\cos^6 u\sin u + 5\cos u\cos v\sin u) \\ y(u,v) = -\frac{1}{15}\sin u(3\cos v - 3\cos^2 u\cos v - 48\cos^4 u\cos v + 48\cos^6 u\cos v - 60\sin u \\ +5\cos u\cos v\sin u - 5\cos^3 u\cos v\sin u - 80\cos^5 u\cos v\sin u + 80\cos^7 u\cos v\sin u) \\ z(u,v) = \frac{2}{15}(3 + 5\cos u\sin u)\sin v; \end{cases}$$

with $u$ chosen uniformly from the interval $[0, \pi]$ and $v = 2u$.



(c) Klein bottle in $\mathbb{R}^3$



(d) Representation of the Klein bottle in $\mathbb{R}^2$, obtained applying `cmds`

In the description, we use the notation $a_+ = \max(a, 0)$ for a scalar $a$. The basic idea of classical scaling is to assume that the dissimilaries are Euclidean distances and then find coordinates that explain them.

For a general dissimilarity matrix $\Delta$, the doubly centered matrix $\Delta^c$ may have negative eigenvalues and that is why in the construction of $Y$, we use the positive part of the eigenvalues. However, if $\Delta$ is an Euclidean dissimilarity matrix, namely $\Delta_{ij} = \|x_i - x_j\|^2$ for a set points $\{x_1, \dots, x_n\}$ in some ambient Euclidean space, then $\Delta^c$ is a positive semi-definite matrix. This follows from the following identity relating a configuration $X$ with the corresponding squared distance matrix $\Delta$:

$$-\frac{1}{2}H\Delta H = HXX^TH. \tag{3.2.6}$$

We perform a perturbation analysis of classical scaling, by studing the effect of perturbing the dissimilarities on the embedding that the algorithm returns. This sort of analysis helps quantify the degree of robustness of a method to noise, and is particularly important in applications where the dissimilarities are observed with some degree of inaccuracy, which is the case in the context of manifold learning.

**Definition 4.** We say that $\Delta \in \mathbb{R}^{m\times m}$ is a $d$-Euclidean dissimilarity matrix if there exists a set of points $\{x_1, \dots, x_m\} \subset \mathbb{R}^d$ such that $\Delta_{ij} = \|x_i - x_j\|^2$.

**Corollary 1.** *Let $\Lambda, \Delta \in \mathbb{R}^{m\times m}$ denote two $d$-Euclidean dissimilarity matrices, with $\Delta$ corresponding to a centered and full rank configuration $Y \in \mathbb{R}^{m\times d}$. Set $\varepsilon^2 = \frac{1}{2}\|H(\Lambda - \Delta)H\|_p$. If it holds that $\|Y^\ddagger\|\varepsilon \le \frac{1}{\sqrt{2}}$, then classical scaling with input dissimilarity matrix $\Lambda$ and dimension $d$ returns a centered configuration*

$Z \in \mathbb{R}^{m \times d}$ *satisfying*

$$\min_{Q \in \mathcal{O}} \|Z - YQ\|_p \leq (1 + \sqrt{2}) \|Y^{\ddagger}\| \varepsilon^2.$$

Since $H$ has one zero eigenvalue and $d-1$ eigenvalues equal to one, $\|H\|_p = (d-1)^{1/p}$. It follows that $\varepsilon^2 \leq \frac{1}{2} d^{2/p} \|\Lambda - \Delta\|_p$.

*Proof.* We have

$$\|\Lambda^c - \Delta^c\|_p = \frac{1}{2} \|H(\Lambda - \Delta)H\|_p = \varepsilon^2.$$

Note that since $\Delta$ and $\Lambda$ are both $d$-Euclidean dissimilarity matrices, using identity (3.2.6), the doubly centered matrices $\Delta^c$ and $\Lambda^c$ are both positive semi-definite and of rank at most $d$. Indeed, since $Y$ is full rank (rank $d$) and centered, then (3.2.6) implies that $\Delta^c$ is of rank $d$. Therefore, for the underlying configuration $Y$ and the configuration $Z$, returned by classical scaling, we have $\Delta^c = YY^T$ and $\Lambda^c = ZZ^T$. We next simply apply Theorem 5, which we can do since $Y$ has full rank by assumption, to conclude. □

We try to appreciate the variation of the configuration that the algorithm returns if we perturb the matrix $\Delta$. So we take $\Lambda$ as $\Delta + \eta R$, where $R$ contains pseudorandom values drawn from the standard uniform distribution on the open interval $(0, 1)$, and we apply `cmds` to both, obtaining respectively the configurations $Y$ and $Z$. Then, to quantify the difference between $Y$ and $Z$, we use the algorithm `procrustes`.

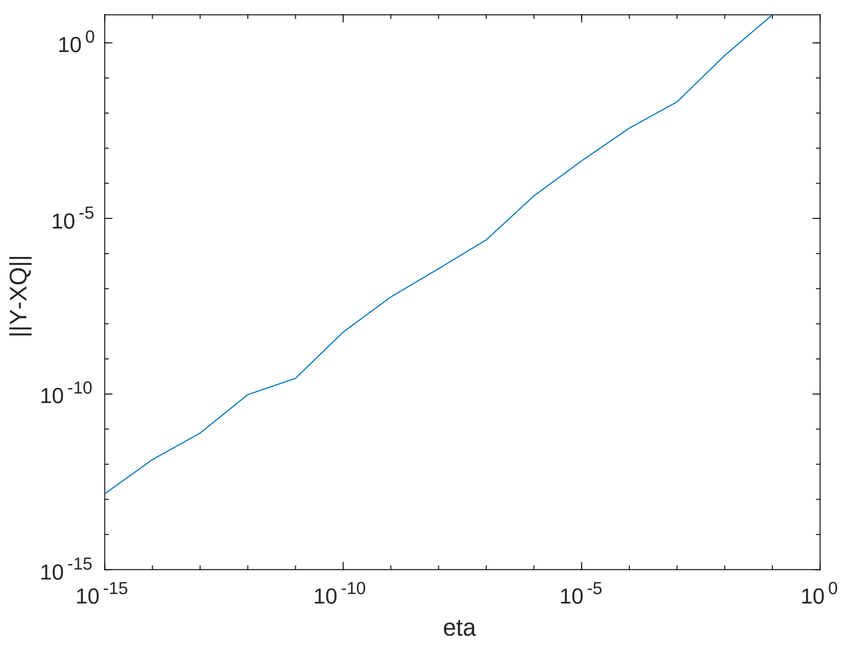


Figure 3.3: Distance between the configurations $Y$ and $ZQ$.

### 3.2.3 Computational cost and the Lanczos algorithm

Classical scaling amounts to performing an eigen-decomposition of the dissimilarity matrix after double-centering. Only the top $d$ eigenvectors are needed if an embedding in dimension $d$ is desired. Using iterative methods such as the Lanczos algorithm, classical scaling can be implemented with a complexity of $O(dn^2)$, where $n$ is the number of items (and therefore also the dimension of the dissimilarity matrix).

The matlab command `eigs` that we have used in `cmds`, is an implementation of the Lanczos algorithm that returns only the top $d$ eigenvalues and eigenvectors.

This algorithm allows us to decompose a Hermitian matrix $A \in \mathbb{R}^{n\times n}$ in the product $VTV^H$ where $V \in \mathbb{R}^{n \times s}$ is a matrix whose columns are orthonormal and $T \in \mathbb{R}^{d \times d}$ is a tridiagonal matrix. If $\lambda$ is an eigenvalue of $A$ and $x$ is an eigenvector of $T$ corresponding to $\lambda$ then $y = Vx$ is an eigenvector of $A$ corresponding to $\lambda$. Indeed $Ay = AVx = VTV^H Vx = VTx = V\lambda x = \lambda Vx = \lambda y$. Therefore the Lanczos algorithm trasforms the decomposition problem of $A$ in the decomposition problem of $T$. As $T$ is tridiagonal, there are specialized algorithms, whose complexity is generally lower, to compute its decomposition.

## 3.3 Trilateration

In applications, particularly if the intent is visualization, the embedding dimension $d$ tends to be small. Even then, the resulting complexity of `cmds` is quadratic in the number of items $n$ to be embedded. There has been some effort in bringing this down to a complexity that is linear in the number of items. We introduce a new algorithm in order to decrease the computational cost of the scaling.

The procedure proposed by de Silva and Tenenbaum (2004), which they called landmark MDS (LMDS) works by selecting a small number of items, perhaps uniformly at random from the itemset, and embedding them via classical scaling. These items are used as landmark points to enable the embedding of the remaining items. The second phase consists in performing trilateration, which aims at computing the location of a point based on its distances to known (landmark) points. Note that this task is closely related to, but distinct, from triangulation, which is based on angles instead. If $l$ items are chosen as landmarks in the first step (out of $n$ items in total), then the procedure has complexity $O(dl^2 + dln)$. Since $l$ can in principle be chosen on the order of $d$, and $d \leq n$ always, the complexity is effectively $O(d^2 n)$, which is linear in the number of items. A good understanding of the robustness properties of LMDS necessitates a good understanding of the robustness properties of not only classical scaling (used to embed the landmark items), but also of trilateration (used to embed the remaining items).

The problem of trilateration is that of positioning a point, or a set of points, based on its (or their) distances to a set of points, which in this context serve as landmarks. In detail, given a set of landmark points $y_1, \dots, y_m \in \mathbb{R}^d$ and a set of dissimilarities $\tilde{\delta}_1, \dots, \tilde{\delta}_m$, the goal is to find $\tilde{y} \in \mathbb{R}^d$ such that $\|\tilde{y} - y_i\|^2$ is close to $\tilde{\delta}_i$ over all $i \in \{1, \dots, m\}$. The following algorithm describes the trilateration method simultaneously applied to multiple points to be located.

**Algorithm 3** Trilateration

```
Input: centered point set $y_1, \dots, y_m \in \mathbb{R}^d$, dissimilarities $\tilde{\Delta} = (\tilde{\delta}_{i,j}) \in \mathbb{R}^{n\times m}$
Output: points $\tilde{y}_1, \dots, \tilde{y}_n \in \mathbb{R}^d$
1: compute $\bar{a} = \frac{1}{n}\sum_{i=1}^n a_i$, where $a_i = (\tilde{\delta}_{i1}, \dots, \tilde{\delta}_{im})$
2: compute the pseudo-inverse $Y^\ddagger$ of $Y = [y_1 \cdots y_m]^T$
3: compute $\tilde{Y}^T = \frac{1}{2}Y^\ddagger(\bar{a}1^T - \Delta^T)$
Return: the row vectors of $\tilde{Y}$, denoted $\tilde{y}_1, \dots, \tilde{y}_n \in \mathbb{R}^d$
```

```
function Yt = trilateration(Y,D2)
sz = size(D2);
n = sz(1);
a = sum(D2)/n;
piY = pinv(Y);
e = ones(1,n);
Yt = piY*(a'*e-D2')/2;
Yt = Yt';
end
```

**Algorithm 4** Landmark Multidimensional Scaling

```
Input: data points $x_1, \dots, x_n \in \mathbb{R}^D$, embedding dimension $d$, number of landmarks $l$
Output: embedding points $\{z_i : i \in \mathcal{L}\} \cup \{\tilde{z}_i : i \notin \mathcal{L}\} \subseteq \mathbb{R}^d$
1: select $\mathcal{L} \subset [n]$ of size $l$
2: compute the square-distance matrix $S_l \in \mathbb{R}^{l\times l}$ of the points in $\{x_i : i \in \mathcal{L}\}$
3: apply classical scaling with input $S_l$ and $d$, resulting in (landmark) points $z_i \in \mathbb{R}^d, i \in \mathcal{L}$
4: for each $i \notin \mathcal{L}$, apply trilateration based on $\{z_i : i \in \mathcal{L}\}$ e $\tilde{\Delta} = (\tilde{\delta}_{i,j}) = \|x_i - x_j\|^2 \in \mathbb{R}^{l\times(n-l)}$, with $i \in \mathcal{L}$ and $j \notin \mathcal{L}$ to obtain $\{\tilde{z}_j : j \notin \mathcal{L}\} \subset \mathbb{R}^d$
Return: points $\{z_i : i \in \mathcal{L}\} \cup \{\tilde{z}_i : i \notin \mathcal{L}\}$
```

```
function Ytot = lmds(X,d,l)
sz = size(X);
D = sz(1);
n = sz(2);
vl = sort(randperm(n,l));
L = zeros(D,l);
L = X(:,vl);
DistL = zeros(l);
for i=1:l
    for j=1:l
        if j>i
            DistL(i,j) = norm(L(:,i)-L(:,j));
        end
    end
end
DistL = DistL + DistL';
[Y,~] = cmds(DistL,d);
vnl = 1:n-l;
j = 1;
k = 1;
for i=1:n-l
    while k<=l && j==vl(k)
       k = k + 1;
       j = j + 1;
    end
    vnl(1,i) = j;
    j = j + 1;
end
D2 = zeros(n-l,l);
for i=1:n-l
    for j=1:l
        D2(i,j) = norm(X(:,vnl(1,i))-L(:,j))^2;
    end
end
Yt = trilateration(Y,D2);
Ytot = zeros(n,d);
Ytot(vl,:) = Y;
Ytot(vnl,:) = Yt;
end
```

We generated a data set of $n = 1000$ points of $\mathbb{R}^{10}$ using the matlab command `normrnd` to choose each vector. Taking the norm of these vectors, we obtain a 10-dimensional sphere. Instead of using directly `cmds` on all the $n$ points, we randomly selecte $l = 4$ landmarks and we apply the algorithm to the Euclidean distance matrix of these points. Then the entire representation is reconstructed thanks to the algorithm `trilateration`.

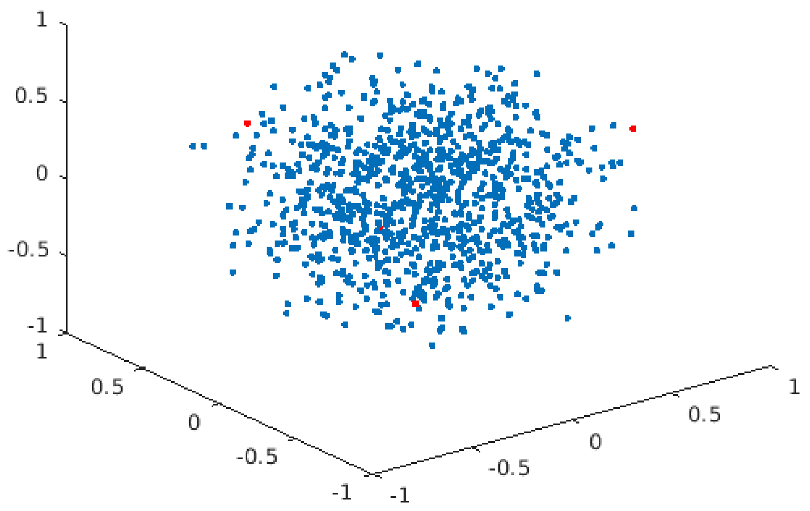


Figure 3.4: 10-dimensional sphere represented in 3 dimensions thanks to the algorithm `lmds`. The red points are the embedding of the 4 landmarks.

Then, in order to analyze how the difference between the representations of the same point set changes when varying the landmarks considered, we make the following experiment. We take as landmarks, in one case the first $j$ points, in the other the last $j$. Increasing the number of landmarks, we observe that the difference between the configurations decrease, until it gets to 0 when $j = n$.

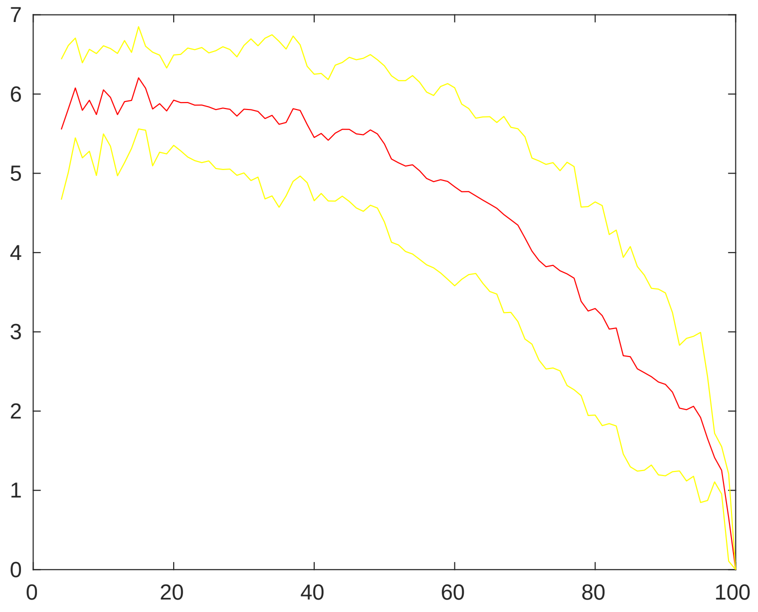


Figure 3.5: Difference of representation when varying the number of lankmarks considered.

Actually, article [1] does not investigate this aspect, but it prefers to focus on other stability result, like the following.

We perturb both the dissimilarities and the landmark points, and quantitatively characterize how it will affect the returned position by trilateration. For a configuration $Y = [y_1, \cdots, y_m]^T$, define its max-radius as

$$\rho_\infty(Y) = \max_{i\in\{1,\ldots,m\}} \|y_i\|,$$

and note that $\rho(Y) \leq \rho_\infty(Y)$. We limit ourselves with a bound in Frobenius norm[1]

[1]All Schatten norms are equivalent here up to a multiplicative constant that depends on $d$, since the matrices that we consider have rank of order $d$.

**Theorem 8.** *Consider a centered configuration $Y \in \mathbb{R}^{m\times d}$ that spans the whole space $\mathbb{R}^d$, and for a given configuration $\tilde{Y} \in \mathbb{R}^{n\times d}$, let $\tilde{\Delta} \in \mathbb{R}^{n\times m}$ denote the matrix of dissimilarities between $\tilde{Y}$ and $Y$, namely $\tilde{\Delta}_{ij} = \|\tilde{y}_i - y_j\|^2$. Let $Z \in \mathbb{R}^{m\times d}$ be another centered configuration that spans the whole space, and let $\tilde{\Lambda} \in \mathbb{R}^{n\times m}$ be an arbitrary matrix. Then, trilateration with inputs $Z$ and $\tilde{\Lambda}$ returns $\tilde{Z} \in \mathbb{R}^{n\times d}$ satisfying*

$$\|\tilde{Z} - \tilde{Y}\|_2 \le \frac{1}{2}\|Z^\ddagger\|\|\tilde{\Lambda} - \tilde{\Delta}\|_2 + 2\|\tilde{Y}\|\|Z^\ddagger\|\|Z - Y\|_2 + 3\sqrt{m}(\rho_\infty(Y) + \rho_\infty(Z))\|Z^\ddagger\|\|Z - Y\|_2 + \|Y\|\|\tilde{Y}\|\|Z^\ddagger - Y^\ddagger\|_2. \tag{3.3.1}$$

We see that the first term captures the effect of the error in the dissimilarity matrix, i.e. $\|\tilde{\Delta} - \tilde{\Lambda}\|$, while the other three terms reflect the impact of the error in the landmark positions, i.e. $\|Z - Y\|$. As we expected, we have a more accurate embedding as these two terms get smaller and in particular, when $\tilde{\Delta} = \tilde{\Lambda}$ and $Y = Z$ (no error in inputs), we have exact recovery.

*Proof.* Let $\bar{a}$ denote the average dissimilarity vector defined in Algorithm 3 based on $Y$ and define $\bar{b}$ similarly on $Z$. Let $\Theta$ denote the matrix of dissimilarities between $\tilde{Y}$ and $Z$ and let $\hat{Y}$ denote the result of Algorithm 3 with inputs $Z$ and $\Theta$. From Algorithm 3, we have

$$\tilde{Y}^T = \frac{1}{2}Y^\ddagger(\bar{a}1^T - \tilde{\Delta}^T), \quad \hat{Y}^T = \frac{1}{2}Z^\ddagger(\bar{b}1^T - \Theta^T), \quad \tilde{Z}^T = \frac{1}{2}Z^\ddagger(\bar{b}1^T - \tilde{\Lambda}^T),$$

due to the fact that the algorithm is exact.

We have

$$\|\tilde{Z} - \tilde{Y}\|_2 \le \|\tilde{Z} - \hat{Y}\|_2 + \|\hat{Y} - \tilde{Y}\|_2.$$

On the one hand,

$$2\|\tilde{Z} - \hat{Y}\|_2 \le \|Z^\ddagger\|\|\tilde{\Lambda} - \Theta\|_2 \le \|Z^\ddagger\|(\|\tilde{\Lambda} - \tilde{\Delta}\|_2 + \|\tilde{\Delta} - \Theta\|_2).$$

On the other hand, starting with the triangle inequality,

$$\begin{aligned} 2\|\hat{Y} - \tilde{Y}\|_2 &= \|Z^\ddagger(\bar{b}1^T - \Theta^T) - Y^\ddagger(\bar{a}1^T - \tilde{\Delta}^T)\|_2 \\ &\le \|Z^\ddagger(\bar{b}1^T - \Theta^T) - Z^\ddagger(\bar{a}1^T - \tilde{\Delta}^T)\|_2 + \|Z^\ddagger(\bar{a}1^T - \tilde{\Delta}^T) - Y^\ddagger(\bar{a}1^T - \tilde{\Delta}^T)\|_2 \\ &\le \|Z^\ddagger\|(\|\bar{b}1^T - \bar{a}1^T\|_2 + \|\Theta - \tilde{\Delta}\|_2) + \|\bar{a}1^T - \tilde{\Delta}^T\|\|Z^\ddagger - Y^\ddagger\|_2. \end{aligned}$$

Together, we find that

$$2\|\tilde{Z} - \tilde{Y}\|_2 \le \|Z^\ddagger\|(\|\tilde{\Lambda} - \tilde{\Delta}\|_2 + 2\|\Theta - \tilde{\Delta}\|_2 + \sqrt{m}\|\bar{b} - \bar{a}\|) + \|\bar{a}1^T - \tilde{\Delta}^T\|\|Z^\ddagger - Y^\ddagger\|_2.$$

In the following, we bound the terms $\|\bar{a}1^T - \tilde{\Delta}^T\|$, $\|\Theta - \tilde{\Delta}\|_2$ and $\|\bar{b} - \bar{a}\|$, separately.

First, using Lemma 1 and the fact that $(Y^\ddagger)^\ddagger = Y$ has full rank,

$$\|\tilde{Y}\| = \frac{1}{2}\|Y^\ddagger(\bar{a}1^T - \tilde{\Delta}^T)\| \geq \frac{1}{2}\|Y\|^{-1}\|\bar{a}1^T - \tilde{\Delta}^T\|.$$

Therefore,

$$\|\bar{a}1^T - \tilde{\Delta}^T\| \leq 2\|Y\|\|\tilde{Y}\|.$$

Next, set $Y = [y_1, \cdots, y_m]^T$ and $Z = [z_1, \cdots, z_m]^T$, as well as $\tilde{Y} = [\tilde{y}_1, \cdots, \tilde{y}_n]^T$. Since

$$(\Theta - \tilde{\Delta})_{ij} = 2\tilde{y}_i^T(y_j - z_j) + \|z_j\|^2 - \|y_j\|^2,$$

we have

$$\|\Theta - \tilde{\Delta}\|_2 = \|2\tilde{Y}(Y^T - Z^T) + 1c^T\|_2 \leq 2\|\tilde{Y}\|\|Y - Z\|_2 + \sqrt{m}\|c\|,$$

with $c = (c_1, \ldots, c_m)$ and $c_j = \|z_j\|^2 - \|y_j\|^2$. Note that

$$\begin{aligned}\|c\|^2 &= \sum_{j\in[m]} (\|z_j\|^2 - \|y_j\|^2)^2 \\ &\leq \sum_{j\in[m]} \|z_j - y_j\|^2(\|z_j\| + \|y_j\|)^2 \\ &\leq (\rho_\infty(Y) + \rho_\infty(Z))^2\|Z - Y\|_2^2,\end{aligned}$$

so that

$$\|\Theta - \tilde{\Delta}\|_2 \leq 2\|\tilde{Y}\|\|Y - Z\|_2 + \sqrt{m}(\rho_\infty(Y) + \rho_\infty(Z))\|Z - Y\|_2.$$

Finally, recall that $\bar{a}$ and $\bar{b}$ are respectively the average of the columns of the dissimilarity matrix for the landmark $Y$ and the landmark $Z$. Using the fact that the $y$'s are centered and that the $z$'s are also centered, we get

$$\bar{b} - \bar{a} = c + c_{avg}1,$$

where $c_{avg} = \frac{1}{m}\sum_{j\in[m]} c_j$ and therefore

$$\|\bar{b} - \bar{a}\|^2 \leq \sum_{j\in[m]} (c_j + c_{avg})^2 = \|c\|^2 + 3mc_{avg}^2 \leq 4\|c\|^2,$$

using the Cauchy-Schwarz inequality at the last step.

Combining all these bounds, we obtain the bound stated in (3.3.1). The last part comes from the triangle inequality and an application of Lemma 2. □

## 3.4 Isomap

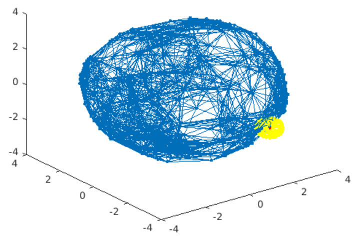
When the points belong to a manifold $M$, we would like to use distances on the manifold instead of the Euclidean ones to better preserve the properties of the set considered. For this reason, we introduce a variation in the distance matrix $D \in \mathbb{R}^{n \times n}$:

$$D = [D_{ij}] = [d_2(x_i, x_j)\mathbb{I}_{\{\|x_i - x_j\| \leq r\}}]_{i,j=1}^n,$$

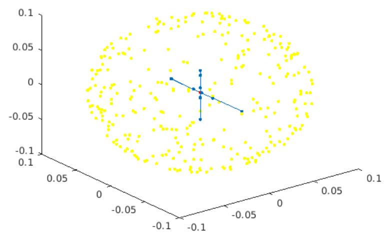
with $r \in \mathbb{R}_+$ a fixed parameter. Then we construct a weighted graph in which the nodes are the points $\{x_i\}$ and there is an edge from the node $i$ to the node $j$ of weight $w$ if and only if $D_{ij} = w$ (if $D_{ij} = 0$ there is not the edge). From this graph, we compute the shortest-path distances graph, i.e. a weighted graph with the points on the manifold as nodes and the shortest-path distance (calculated on the previous graph) as edge weights.

The value $r$ plays a basic role: it is the radius of the ball of each point $m$ within which the others are visible by $m$. Outside this ball, points are at infinite distance from the point $m$. The more we take $r$ small, the more the distance is similar to the one on the manifold. On the other hand, if we take $r$ too small, the shortest-path distances graph could be disconnected, since we consider a finite number of points.

To underline this dependence of the representation on the radius $r$, we plot the configurations obtained applying the algorithm `isomap` on the same 3-dimensional sphere of $n = 300$ points, the first time with $r = 0.5$ and the second time with $r = 0.1$. The lines link two points if their Euclidean distance on the sphere is less than $r$. In yellow, a sphere of radius $r$ centered in a random point of the representation. The points of the sphere are obtained taking the norm of the vectors genereted by the matlab command `normrnd`.



(a) 3-dimensional sphere represented through the algorithm `isomap` with $r = 0.5$.



(b) 3-dimensional sphere represented through the algorithm `isomap` with $r = 0.1$: there is only the plot of the connected component of the random point and the sphere centered on it.

Isomap is composed by two parts: first, it computes the shortest-path distances from the $r$-ball neighborhood graph based on the data points; then it passes the obtained distance matrix to classical scaling (together with the desires embedding dimension) to obtain an embedding. The algorithm is known to work well when the underlying manifold is isometric to a convex domain in $\mathbb{R}^d$. Indeed, assuming an infinite sample size, so that the data points are in fact all the points of the manifold, as $r \to 0$, the shortest-path distances will converge to the geodesic distances on the manifold, and thus, in that asymptote (infinite sample size and infinitesimal radius), an isometric embedding in $\mathbb{R}^d$ is possible under the stated condition. It is interesting however to consider more general manifolds.

**Algorithm 5** Isomap

```
Input: data points $x_1, \dots, x_n \in \mathbb{R}^D$, embedding dimension $d$, neighborhood radius $r$
Output: embedding points $z_1, \dots, z_n \in \mathbb{R}^d$
1: construct the graph on $[n]$ with edge weights $w_{ij} = \|x_i - x_j\| \mathbb{I}_{\{\|x_i - x_j\| \leq r\}}$
2: compute the shortest-path distances in that graph $\Gamma = (\gamma_{ij})$
3: apply classical scaling with inputs $\Gamma^{\circ 2}$ and $d$, resulting in points $z_1, \dots, z_n \in \mathbb{R}^d$
Return: the points $z_1, \dots, z_n$
```

```
function Z = isomap(X,d,r)
```

```
sz = size(X);
D = sz(1);
n = sz(2);
G = zeros(n);
k = 1;
for i=1:n
    for j=1:n
        if j>i
            nrm = norm(X(:,i)-X(:,j));
            if nrm<=r
                G(i,j) = nrm;
            end
        end
    end
end
G = G + G';
Gamma = dijkstra(G);
[Z,~] = cmds(Gamma.^2,d);
end
```

To calculate the shortest-path distances of the nodes on the graph, we have used a variant of the Dijkstra's algorithm.

The Dijkstra's algorithm finds the shortest path between two given nodes or between a given "source" node of the graph and the others, forming a shortest-path tree. It runs in time $O((|V|+|E|)\log|V|)$ (where $|V|$ is the number of nodes and $|E|$ is the number of edges) if implemented with a priority queue, in $O(|V|^2)$ if implemented with an array and in $O(|E|+|V|\log|V|)$ if implemented with a Fibonacci heap priority queue.

Actually, we don't need the shortest path, but the length of it. So we have not saved all the nodes crossed by the algorithm, but just the sum of the weights of the edges crossed. At each step, the algorithm select the minimum $m$ of these shortest-path distances, initialized with the distances of the $r$-ball neightborhood graph, and the node $p$ that realize it. Then it looks at the distances between this node and the others; if the sum of $m$ and the distance between $p$ and a node $q$, a.k.a. $d(p,q)$, is less then the distance between the source node and $q$, it updates the value of this distance with $m+d(p,q)$. It repeats this procedure $n$ times, one per node. In totally, in our case, the complexity is $O(n^3)$.

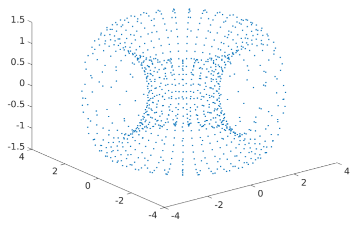
```
function Gamma = dijkstra(G)
sz = size(G);
n = sz(1);
l = sz(2);
M = intmax;
Gamma = G;
for i=1:n
```

```
    dist = Gamma(i,:);
    for j=1:l
        if j~=i && dist(j)==0
            dist(j) = M;
        end
    end
    tocheck = 1:l;
    st = l;
    [m,t] = min(dist);
    while st>0 && m<M
        p = tocheck(t);
        for j=1:st
            q = tocheck(j);
            if G(p,q)~=0 && m+G(p,q)<dist(q)
                dist(q) = m + G(p,q);
            end
        end
        if st>1
            temp = zeros(1,st-1);
            if t~=1
                temp(1,1:(t-1)) = tocheck(1:(t-1));
            end
            temp(1,t:(st-1)) = tocheck((t+1):st);
            tocheck = temp;
            [m,t] = min(dist(tocheck));
        end
        st = st - 1;
    end
    for j=1:l
        if dist(j)==M
            dist(j) = 0;
        end
    end
    Gamma(i,:) = dist;
end
end
```

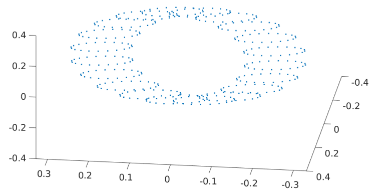
In the next example, we see two different configurations returned by the algorithm applied on two different tori; the first one, is the torus described above (Section 3.2.2). The second one is obtained looking at the torus as the quotient $\mathbb{R}^2/\mathbb{Z}^2$: we uniformly take points in the square $[0,1]^2$ and we compute the distance as follows:

$$d(x_i, x_j) = \min_{k\in\mathbb{Z}^2} \|x_i - x_j + k\|.$$

This is the so-called flat distance.

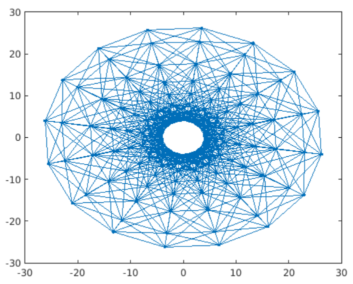


(c) Torus in 3 dimensions achieved through the algorithm `isomap` with $r = 0.5$, using the Euclidean distance.

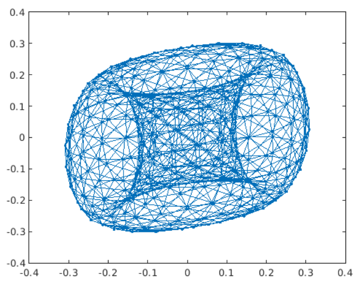


(d) Torus in 3 dimensions achieved through the algorithm `isomap` with $r = 0.08$, using the flat distance.

As above, we link the points whose distance is less than $r$.



(e) Torus in 2 dimensions achieved through the algorithm `isomap` with $r = 1.4$, using the Euclidean distance.



(f) Torus in 2 dimensions achieved through the algorithm `isomap` with $r = 0.08$, using the flat distance.

Also in this case, in order to reduce the computational cost, it is preferable to use the landmark version of the algorithm, i.e. the landmark isomap: instead of applying the algorithm `cmds` on the shortest-path distances graph, it is applied to a subgraph of it, formed by a small number $l$ of nodes. Finally, using the trilateration, the entire representation is reconstructed.

**Algorithm 6** Landmark Isomap

```
Input: data points $x_1, \dots, x_n \in \mathbb{R}^D$, embedding dimension $d$, neighborhood radius $r$, number of landmarks $l$
Output: embedding points $\{z_i : i \in \mathcal{L}\} \cup \{\tilde{z}_i : i \notin \mathcal{L}\} \subseteq \mathbb{R}^d$ for a choice of $|\mathcal{L}| = l$ landmarks
1: construct the graph on $[n]$ with edge weights $w_{ij} = \|x_i - x_j\| \mathbb{I}_{\{\|x_i - x_j\| \leq r\}}$
2: select $\mathcal{L} \subset [n]$ of size $l$
3: compute the shortest-path distances in that graph $\Gamma = (\gamma_{ij})$ for $(i,j) \in [n] \times \mathcal{L}$
4: apply classical scaling with inputs $\Gamma^{\circ 2}_{\mathcal{L} \times \mathcal{L}}$ and $d$, resulting in (landmark) points $z_i \in \mathbb{R}^d, i \in \mathcal{L}$
5: for each $i \notin \mathcal{L}$, apply trilateration based on $\{z_j : j \in \mathcal{L}\}$ and $\Gamma^{\circ 2}_{i \times \mathcal{L}}$ to obtain $\tilde{z}_i \in \mathbb{R}^d$
Return: points $\{z_i : i \in \mathcal{L}\} \cup \{\tilde{z}_i : i \notin \mathcal{L}\}$
```

```matlab
function Z = landisomap(X,d,r,l)
sz = size(X);
D = sz(1);
n = sz(2);
G = zeros(n);
for i=1:n
    for j=1:n
        if j>i
            nrm = norm(X(:,i)-X(:,j));
            if nrm<=r
                G(i,j) = nrm;
            end
        end
    end
end
G = G + G';
vl = sort(randperm(n,l));
GL = G(:,vl);
Gamma = dijkstra(GL);
GammaL = Gamma(vl,:);
[Y,~] = cmds(GammaL.^2,d);
vnl = 1:n-l;
j = 1;
k = 1;
for i=1:n-l
    while k<=l && j==vl(k)
      k = k + 1;
      j = j + 1;
    end
    vnl(1,i) = j;
    j = j + 1;
end
```

```
D2 = zeros(n-l,l);
for i=1:n-l
    for j=1:l
        D2(i,j) = Gamma(vnl(1,i),j)^2;
    end
end
Yt = trilateration(Y,D2);
Z = zeros(n,d);
Z(vl,:) = Y;
Z(vnl,:) = Yt;
end
```

# Chapter 4

# Further experiments on graphs

We have modified the algorithm `cmds`, in order to apply MDS also to data sets with non-Euclidean distances. We have taken the absolute values of the eigenvalues of $G^c$; indeed, if the distance is not Euclidean, they could be negative. The first example is the Hamming cube. Its vertex are the strings of 0 and 1 of length $D$ and the distance between two vertex is the number of positions at which the corresponding symbols are different. This cube cannot be embedded in an Euclidean space. Indeed, for example, if we take $D = 2$ and we compute the spectrum of $G^c = -\frac{1}{2}S^c$, where $S$ is the square-distance matrix of the cube, we obtain $Spec(G^c) = \{-1, 0, 2\}$ and therefore $G^c$ is not a centering psd matrix. Thus, for the theorem 7 (Section 3.2.1), $S$ cannot be an Euclidean square-distance matrix.

In the following examples, we will link two nodes if their distance is equal to 1.

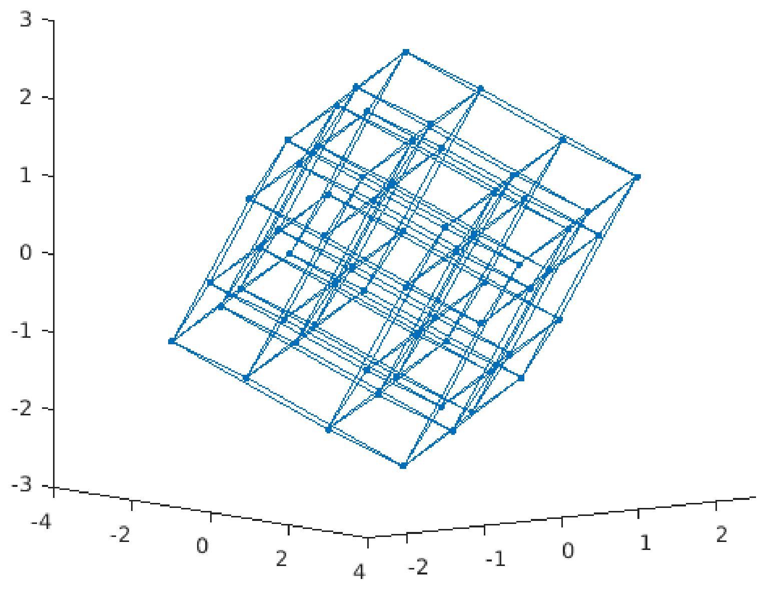


Figure 4.1: Hamming cube with 64 vertices

An other examples of non-Euclidean distance is the geodesic one: the distance between the node $n$ and the node $m$ is the number of edges of a shortest path between $n$ and $m$. We will see two graphs and their representation obtained with this distance: a complete binary tree and a complete graph.

A complete binary tree is a tree in which all the nodes, apart from the leaves, has two sons.

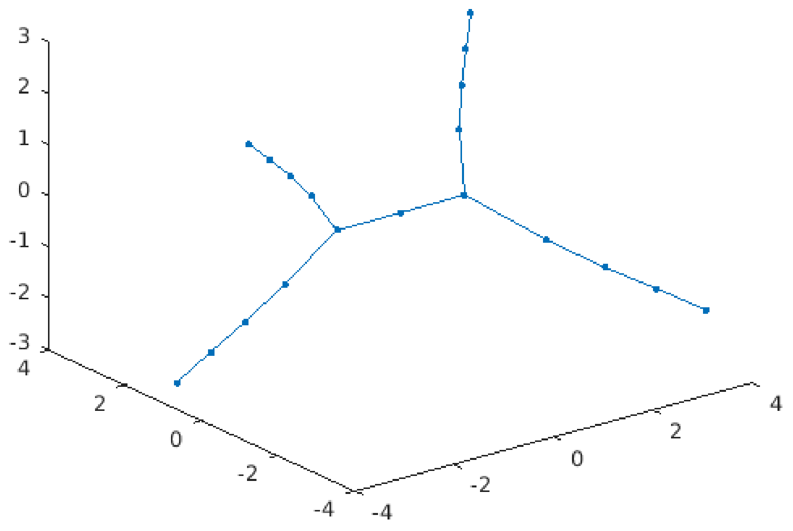


Figure 4.2: Complete binary tree of height 5

In the representation there are not many nodes and edges; this is due to the fact that the algorithm `cmds` returns coincident points.

A complete graph is a graph in which every node is linked by an edge with each of the others.

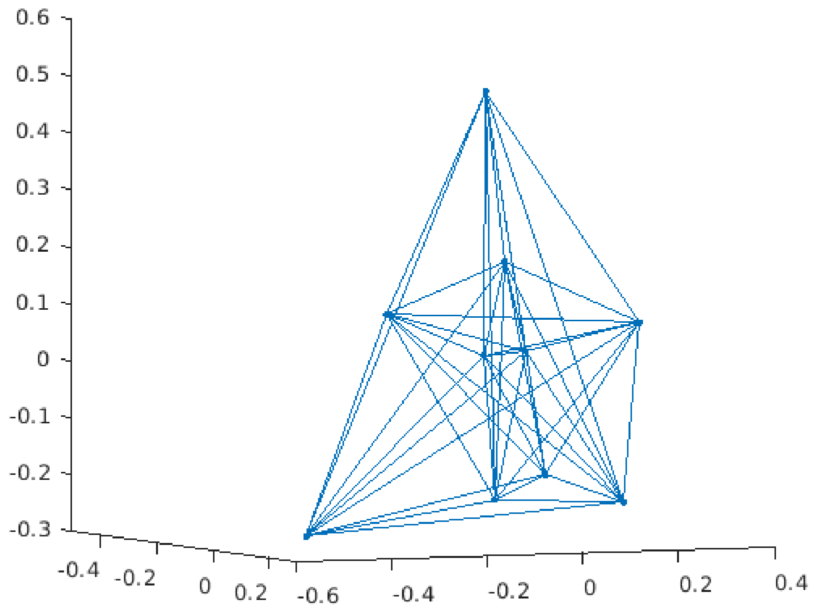


Figure 4.3: Complete graph formed by 10 nodes

# Bibliography

[1] Ery Arias-Castro, Adel Javanmard, Bruno Pelletier, *Perturbation Bounds for Procrustes, Classical Scaling, and Trilateration, with Applications to Manifold Learning*, (2019)

[2] Jianzhong Wang, *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*, (2011)

[3] G. W. Stewart, Ji-guang Sun, *Matrix Perturbation Theory*, (1990)

[4] George A. F. Seber, *Multivariate observations*, (1984)

[5] `https://en.wikipedia.org/wiki/Lanczos_algorithm`

[6] `https://scikit-learn.org/stable/modules/manifold.html`