Ordination – Low Dimensional Views

John Maindonald

August 29, 2010

Contents

1 Ordination

Ordination is a generic name for methods for providing a low-dimensional view of points in multi-dimensional space, such that "similar" objects are near each other and dissimilar objects are separated. The plot(s) from an ordination in 2 or 3 dimensions may provide useful visual clues on clusters in the data and on outliers. The methods described help all use some form of multi-dimensional scaling (MDS)

Distances may be already given, or it may be necessary to start by calculating distances between points. In either case, the distances are the starting point for an ordination. Similarities will be transformed into distances before starting the ordination calculations.

Examples are:

1. From Australian road travel distances between cities and larger towns, can we derive a plausible "map" showing the relative geographic locations?

2. Starting with genomic data, various methods are available for calculating genomic "distances " between, e.g., different insect species. The distance measures are based on evolutionary models that aim to give distances between pairs of species that are a monotone function of the time since the two species separated.

3. Given a matrix **X** of n observations by p variables, a low-dimensional representation is required, i.e., the hope is that a major part of the information content in the data can be summarized in a small number of constructed variables. There is typically no good model, equivalent to the evolutionary models used by molecular biologists, that can be used to motivate distance calculations. There is then a large element of arbritariness in the distance measure used.

If data can be separated into known classes that should be reflected in any ordination, then the scores from classification using `lda()` may be a good basis for an ordination. Plots in 2 or perhaps 3 dimensions may then reveal additional classes and/or identify points that may be misclassified and/or are in some sense outliers. It may indicate whether the classes that formed the basis for the ordination seem real and/or the effectiveness of the discrimination method in choosing the boundaries between classes.

The function `randomForest()` is able to return "proximities" that are measures of the closeness of any pair of points. These can be turned into rough distance measures that can then form the basis for an ordination. With Support Vector Machines, decision values are available from which distance measures can be derived and used as a basis for ordination.

1.1 Distance measures

1.1.1 Euclidean distances

Treating the rows of **X** (n by p) as points in a p-dimensional space, the squared Euclidean distance d_{ij}^2 between points i and j is

$$d_{ij}^2 = \sum_{k=1}^{p} (x_{ik} - x_{jk})^2$$

The distances satistfy the triangle inequality

$$d_{ij} \leq d_{ik} + d_{kj}$$

The columns of **X** can be arbitrarily transformed before calculating the d_{ij}. Where all elements of a column are positive, use of the logarithmic transformation is common. A logarithmic scale makes sense for biological morphometric data, and for other data that has similar characteristics. For morphometric data, the effect is to focus attention on relative changes in the various body proportions, ignoring the overall magnitude.

The columns may be standardized before calculating distances, i.e., scaled so that the standard deviation is one. The columns may be weighted differently. Use of an unweighted measure with all columns scaled to a standard deviation of one is equivalent to working with the unscaled columns and calculating d_{ij}^2 as

$$d_{ij}^2 = \sum_{k=1}^{p} w_{ij}(x_{ik} - x_{jk})^2$$

where $w_{ij} = (s_i s_j)^{-1}$ is the inverse of the product of the standard deviations for columns i and j. Results may depend strongly on the distance measure.

1.1.2 Non-Euclidean distance measures

Euclidean distance is one of many possible choices of distance measures, still satisfying the triangle inequality. As an example of a non-Euclidean measure, consider the Manhattan distance. This has

$$d_{ij} = \sum_{k=1}^{p} \mid x_{ik} - x_{jk} \mid$$

The Manhattan distance is the shortest distance for a journey that always proceeds along one of the co-ordinate axes. In Manhattan in New York, streets are laid out in a rectangular grid. This is then (with $k = 2$) the walking distance along one or other street. For other choices, see the help page for the function `dist()`.

The function `daisy()` in the *cluster* package offers a still wider range of possibilities, including distance measures that can be used when columns that are factor or ordinal. It has an argument `stand` that can be used to ensure standardization when distances are calculated. Unless measurements are comparable (e.g., relative growth, as measured perhaps on a logarithmic scale, for different body measurements), then it is usually desirable to standardize before using ordination methods to examine the data.

Irrespective of the method used for the calculation of the distance measure, ordination methods yield a representation in Euclidean space. Depending on the distance measure and the particular set of distances, an exact representation may or may not be possible.

See Gower & Legendre (1986) for a detailed discussion of the netric and Euclidean propoerties of a wide variety of similarity coefficients.

1.2 From distances to a configuration in Euclidean space

Given a set of "distances" d_{ij} that satisfy the triangle inequality, there is in general no guarantee that it will be possible to derive a configuration X in Euclidean that exactly reproduces those distances. Where Euclidean distances are calculated between the rows of a matrix **X**, clearly the matrix **X** is itself one possible configuration in Euclidean space. So also is **XP**, where **P** is an orthogonal matrix.

Suppose however that non-metric distances are derived from a matrix **X**. For example, they may be Manhattan distances. For some distance measures, it is always possible to find a configuration X in Euclidean space (an embedding) that exactly reproduces those distances. For other choices of distance this is not always possible.

It is however always possible to find a configuration X in Euclidean space in which the distances are approximated, perhaps rather poorly. This is true whether ot not the triangle inequality is satisfied. It will become apparent in the course of seeking the configuration whether an exact embedding (matrix X) is possible, and how accurate this embedding is.

Given such a matrix X if it exists, we can write

$$\begin{aligned} d_{ij}^2 &= \sum_{k=1}^{p} (\mathsf{x}_{ik} - \mathsf{x}_{jk})^2 \\ &= \sum_{k=1}^{p} \mathsf{x}_{ik}^2 + \sum_{k=1}^{p} \mathsf{x}_{jk}^2 - 2\sum_{k=1}^{p} \mathsf{x}_{ik}\mathsf{x}_{jk} \end{aligned}$$

Thus

$$d_{ij}^2 = q_{ii} + q_{jj} - 2q_{ij}$$

where $q_{ii} = \sum_{k=1}^{p} \mathsf{x}_{ik}^2$; $q_{ij} = \sum_{k=1}^{p} \mathsf{x}_{ik}\mathsf{x}_{jk}$. Observe that q_{ij} is the (i, j)th element of the matrix $\mathbf{Q} = \mathsf{X}\mathsf{X}^T$. Thus, the matrix $\mathbf{Q}$ has all the information needed to derive the distances. Because $\mathbf{Q} = \mathsf{X}\mathsf{X}^T$, it is positive semidefinite.

The mapping from the q_{ij} to the d_{ij} is one to one. Given distances, it is possible to find such a matrix $\mathbf{Q}$, if it exists. The detailed derivation is in Section 2. This shows that it is always possible to derive a symmetric matrix $\mathbf{Q}$. If and only if $\mathbf{Q}$ is positive definite, there is an exact embedding X in Euclidean space.

Having thus recovered a symmetric matrix $\mathbf{Q}$, the spectral decomposition yields

$$\mathbf{Q} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

where $\mathbf{\Lambda}$ is a diagonal matrix. The diagonal elements λ_i are ordered so that

$$\lambda_1 \geq \lambda_2 \ldots \geq \lambda_n$$

Providing $\lambda_i \geq 0$, choose

$$\mathsf{X} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$$

As the rows and columns of $\mathbf{Q}$ sum to zero, $\mathbf{Q}$ is singular. Hence if $\mathbf{Q}$ is positive definite, as required for exact embedding in Euclidean space, $\lambda_i \geq 0$ for all i and $\lambda_n = 0$.

Important points are:

- Often, most of the information will be in the first few dimensions. We may for example be able to approximate $\mathbf{Q}$ by replacing $\mathbf{\Lambda}$ in $\mathbf{Q} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ by a version of $\mathbf{\Lambda}$ in which diagonal elements after the kth have been set to zero. If `cmdscale()` is called with `eig=TRUE`, it returns both the eigenvalue information (the λ_i) and a goodness of fit statistic, by default (assuming at least two non-zero λ_i) for the configuration with $k = 2$.

- If $\mathbf{Q}$ is not positive semidefinite, the ordination can still proceed. However one or more eigenvalues λ_i will now be negative. If relatively small, it may be safe to ignore dimensions that correspond to negative eigenvalues. It is then more than otherwise desirable to check that the ordination reproduces the distances with acceptable accuracy.

1.2.1 The connection with principal components

Let $\mathbf{X}$ be a matrix that is the basis for the calculation of Euclidean distances, after any transformations and/or weighting. Then metric p-dimensional ordination, applied to Euclidean distances between the rows of $\mathbf{X}$, yields an orthogonal transformation of the space spanned by the columns of $\mathbf{X}$. If the successive dimensions are chosen to "explain" successively larger proportions of the trace of $\mathbf{X}\mathbf{X}^T$, it is equivalent to the principal components transformation. Thus `cmdscale()` yields, by a different set of matrix manipulations, a principal components decomposition.

1.3 Non-metric scaling

These methods all start from "distances", but allow greater flexibility in their use to create an ordination. The aim is to represent the "distances" in some specified number of dimensions, typically two dimensions. As described here, a first step is to treat the distances as Euclidean, and determine a configuration in Euclidean space. These Euclidean distances are then used as a starting point for a representation in which the requirement that these are Euclidean distances, all determined with equal accuracy, is relaxed. The methods that will be noted here are:

Sammon scaling: A configuration with distances $\tilde{d}$ is chosen to minimize a weighted squared "stress"

$$\frac{1}{\sum_{i \neq j} d_{ij}} \sum_{i \neq j} \frac{(d_{ij} - \tilde{d}_{ij})^2}{d_{ij}}$$

Kruskal's non-metric multidimensional scaling: This aims to minimize

$$\frac{\sum_{i\neq j}(\theta(d_{ij}) - \tilde{d}_{ij})^2}{\sum_{i\neq j}\tilde{d}_{ij}^2}$$

with respect to the configuration of points and an increasing function θ of the distance d_{ij}.

Often, it makes sense to give greater weight to small distances than to large distances. The distance scale should perhaps not be regarded as rigid. Larger distances may not be measured on the same Euclidean scale as shorter distances. The ordination should perhaps preserve relative rather than absolute distances.

1.4 Examples

1.4.1 Australian road distances

The distance matrix that will be used is in the matrix `audists`, in the image file **audists.Rdata**. Consider first the use of classical multi-dimensional scaling, as implemented in the function `cmdscale()`:

```
> library(DAAGxtras)
> aupoints <- cmdscale(audists)
> plot(aupoints)
> text(aupoints, labels=paste(rownames(aupoints)))
```

An alternative to `text(aupoints, labels=paste(rownames(aupoints)))`, allowing better placement of the labels, is `identify(aupoints, labels=rownames(aupoints))`. We can compare the distances in the 2-dimensional representation with the original road distances:

```
> audistfits <- as.matrix(dist(aupoints))
> misfit <- as.matrix(dist(aupoints)) - as.matrix(audists)
> for (j in 1:9)for (i in (j+1):10){
    lines(aupoints[c(i,j), 1], aupoints[c(i,j), 2], col="gray")
    midx <- mean(aupoints[c(i,j), 1])
    midy <- mean(aupoints[c(i,j), 2])
    text(midx, midy, paste(round(misfit[i,j])))
    }
> colnames(misfit) <- abbreviate(colnames(misfit),6)
> print(round(misfit))
```

```
          Adelad Alice Brisbn Broome Cairns Canbrr Darwin Melbrn Perth Sydney
Adelaide       0   140   -792   -156    366     20     11     82   482   -273
Alice        140     0  -1085   -175    -41     76   -118    106   -26   -314
Brisbane    -792 -1085      0    198    319    -25   -233   -471   153    -56
Broome      -156  -175    198      0    527     -7      6    -65   990     70
Cairns       366   -41    319    527      0    277    -31    178     8    251
Canberra      20    76    -25     -7    277      0     -1   -241   372     -8
Darwin        11  -118   -233      6    -31     -1      0    -12    92    -58
Melbourne     82   106   -471    -65    178   -241    -12      0   301   -411
Perth        482   -26    153    990      8    372     92    301     0    271
Sydney      -273  -314    -56     70    251     -8    -58   -411   271      0
```

The graph is a tad crowded, and for detailed information it is necessary to examine the table.

It is interesting to overlay this "map" on a physical map of Australia.

```
> library(oz)
> oz()
> points(aulatlong, col="red", pch=16, cex=1.5)
> comparePhysical <- function(lat=aulatlong$latitude, long=aulatlong$longitude,
                              x1=aupoints[,1], x2 = aupoints[,2]){
    ## Get best fit in space of (latitude, longitude)
    fitlat <- predict(lm(lat ~ x1+x2))
    fitlong <- predict(lm(long ~ x1+x2))
    x <- as.vector(rbind(lat, fitlat, rep(NA,10)))
    y <- as.vector(rbind(long, fitlong, rep(NA,10)))
    lines(x, y, col=3, lwd=2)
    }
> comparePhysical()
```

An objection to `cmdscale()` is that it gives long distances the same weight as short distances. It is just as prepared to shift Canberra around relative to Melbourne and Sydney, as to move Perth. It makes more sense to give reduced weight to long distances, as is done by `sammon()` (*MASS*).

```
> aupoints.sam <- sammon(audists)

Initial stress        : 0.01573
stress after  10 iters: 0.00525, magic = 0.500
stress after  20 iters: 0.00525, magic = 0.500

> oz()
> points(aulatlong, col="red", pch=16, cex=1.5)
> comparePhysical(x1=aupoints.sam$points[,1], x2 = aupoints.sam$points[,2])
```

Notice how Brisbane, Sydney, Canberra and Melbourne now maintain their relative positions much better.

Now try full non-metric multi-dimensional scaling (MDS). This preserves only, as far as possible, the relative distances. A starting configuration of points is required. This might come from the configuration used by `cmdscale()`. Here, however, we use the physical distances.

```
> oz()
> points(aulatlong, col="red", pch=16, cex=1.5)
> aupoints.mds <- isoMDS(audists, as.matrix(aulatlong))

initial  value 11.875074
iter   5 value 5.677228
iter  10 value 4.010654
final  value 3.902515
converged

> comparePhysical(x1=aupoints.mds$points[,1], x2 = aupoints.mds$points[,2])
```

Notice how the distance between Sydney and Canberra has been shrunk quite severely.

1.4.2 Genetic Distances – Hasegawa's selected primate sequences

Here, matching genetic DNA or RNA or protein or other sequences are available from each of the different species. Distances are based on probabilistic genetic models that describe how gene sequences change over time. The package *ape* implements a number of alternative measures. For details see `help(dist.dna)`.

Hasegawa's sequences were selected to have as little variation in rate, along the sequence, as possible. The sequences are available either from the *DAAGxtras* package or from the wepage `http://evolution.genetics.washington.edu/book`. They can be read into R as:

```
> ## Obtain data from the web page on the next line, calculate distances
> url <- "http://evolution.genetics.washington.edu/book/primates.dna"
> library(ape)
> primates.dna <- read.dna(url)
> ## Alternative - download and then read in data
> # download.file(webpage, destfile="primates.txt")   # Alternative
> # primates.dna <- read.dna("primates.txt")
> ## Now calculate distances, using Kimura's K80 model
> primates.dist <- dist.dna(primates.dna, model="K80")
```

The *DAAGxtras* package has the dataset `primateDNA`. These are the same data, but stored in character format. For use with `dist.dna()` use the function `dist.dna()` to convert the data to a binary format. The following is an alternative to the code given above:

```
> ## Use dataset primateDNA from the DAAGbio package
> library(DAAGbio)
> library(ape)
> ## Calculate distances, using Kimura's K80 model
> primates.dist <- dist.dna(as.DNAbin(primateDNA), model="K80")
```

We now try for a two-dimensional representation, using `cmdscale()` from the *MASS* package:

```
> primates.cmd <- cmdscale(primates.dist)
> eqscplot(primates.cmd, xlab="Axis 1", ylab="Axis 2")
> lefrt <- 2+2*(primates.cmd[,1] < mean(par()$usr[1:2]))
> text(primates.cmd[,1], primates.cmd[,2], row.names(primates.cmd), pos=lefrt)
```

```
initial  value 19.892084
iter   5 value 13.849956
iter  10 value 13.553589
final  value 13.527427
converged
```

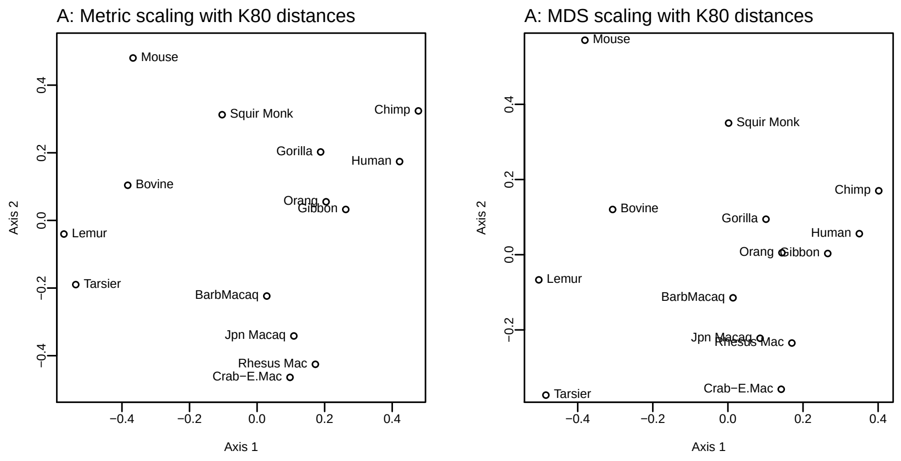

Figure 1: The plot on the left has used classical metric scaling, i.e., calculations seek a Euclidean space representation of the distances. The plot on the right has used the `isoMDS()` function to show results from Kruskal's non-metric multidimensional scaling, i.e., the "distances" provide an ordering in Euclidean space.

Now see how well Figure 1A reproduces the distances:

```
> d <- dist(primates.cmd)
> sum((d-primates.dist)^2)/sum(primates.dist^2)
```

```
[1] 0.101
```

With only around 5% of the sum of squared distances unaccounted for, it is hardly worth examining a 3-dimensional representation. Here, however, is the code:

```
> library(lattice)
> primates.cmd <- cmdscale(primates.dist, k=3)
> cloud(primates.cmd[,3] ~ primates.cmd[,1]*primates.cmd[,2])
> d <- dist(primates.cmd)
> sum((d-primates.dist)^2)/sum(primates.dist^2)
```

Now repeat the above with `sammon()` and `mds()`.

```
> primates.sam <- sammon(primates.dist, primates.cmd, k=2)
> eqscplot(primates.sam$points)
> text(primates.sam$points[,1], primates.sam$points[,2],
         row.names(primates.sam$points), pos=lefrt)
```

There is no harm in asking for three dimensions, even if only two of them will be plotted.

The following code is used to for the multidimensional scaling representation in Figure 1B:

```
> primates.mds <- isoMDS(primates.dist, primates.cmd, k=2)
> eqscplot(primates.mds$points, xlab="Axis 1", ylab="Axis 2")
> text(primates.mds$points[,1], primates.mds$points[,2],
         row.names(primates.mds$points), pos=lefrt)
```

1.4.3 Pacific rock art

Here, the the 614 features were all binary – the presence or absence of specific motifs in each of 98 Pacific sites. (Actually, there were 103 sites, but 5 were omitted because they had no motifs in common with any of the other sites.) Data are from Meredith Wilson's PhD thesis at Australian National University.

The binary measure of distance was used – the number of locations in which only one of the sites had the marking, as a proportion of the sites where one or both had the marking. Here then is the calculation of distances:

```
> pacific.dist <- dist(x = as.matrix(rockArt[-c(47,54,60,63,92), 28:641]),
                          method = "binary")
> sum(pacific.dist==1)/length(pacific.dist)

[1] 0.631

> plot(density(pacific.dist, to = 1))
> ## Now check that all columns have some distances that are less than 1
> symmat <- as.matrix(pacific.dist)
> table(apply(symmat, 2, function(x) sum(x==1)))

13 21 27 28 29 32 33 35 36 38 40 41 42 43 44 45 46 47 48 49 51 52 53 54 55 56
 1  1  1  1  2  1  2  1  2  2  1  2  4  3  1  3  1  2  1  1  2  2  3  2  2  2
57 58 61 62 64 65 66 67 68 69 70 71 73 75 76 77 79 81 83 84 85 90 91 92 93 94
 1  3  3  1  2  1  1  1  3  3  1  1  4  1  2  1  1  1  2  1  1  3  1  1  3  1
95 96 97
 1  3  4
```

It turns out that 63% of the distances were 1. This has interesting consequences, for the plots we now do.

```
> pacific.cmd <- cmdscale(pacific.dist)
> plot(pacific.cmd)
> pacific.mds <- isoMDS(pacific.dist, pacific.cmd)
```

```
initial  value 54.388728
iter   5 value 40.556391
iter  10 value 37.297430
iter  15 value 36.120966
iter  20 value 35.291828
iter  25 value 34.785333
iter  30 value 34.259107
iter  35 value 33.771381
iter  35 value 33.739070
iter  35 value 33.723549
final  value 33.723549
converged
```

```
> plot(pacific.mds$points)
```

2 Theory – From Distances to Representation in Euclidean Space

Given an embedding X in Euclidean space, if it exists, the squared Euclidean distance between points i and j can be written

$$\begin{aligned} d_{ij}^2 &= \sum_{k=1}^{p}(\mathsf{x}_{ik} - \mathsf{x}_{jk})^2 \\ &= \sum_{k=1}^{p}\mathsf{x}_{ik}^2 + \sum_{k=1}^{p}\mathsf{x}_{jk}^2 - 2\sum_{k=1}^{p}\mathsf{x}_{ik}\mathsf{x}_{jk} \end{aligned}$$

Thus

$$d_{ij}^2 = q_{ii} + q_{jj} - 2q_{ij} \tag{1}$$

where $q_{ii} = \sum_{k=1}^{p}\mathsf{x}_{ik}^2$; $q_{ij} = \sum_{k=1}^{p}\mathsf{x}_{ik}\mathsf{x}_{jk}$.

Observe that q_{ij} is the (i, j)th element of the matrix $\mathbf{Q} = \mathsf{X}\mathsf{X}'$. Thus, the matrix $\mathsf{X}\mathsf{X}'$ has all the information needed to construct distances.

Now require that columns of X are centered, i.e.

$$\sum_{i=1}^{n}\mathsf{x}_{ik} = 0, i = 1, \dots p$$

This implies that

$$\begin{aligned} \sum_{i=1}^{n} q_{ij} &= \sum_{i=1}^{n}(\sum_{k=1}^{p}\mathsf{x}_{ik}\mathsf{x}_{jk}) \\ &= \sum_{k=1}^{p}(\sum_{i=1}^{n}\mathsf{x}_{ik}\mathsf{x}_{jk}) \\ &= \sum_{k=1}^{p}(\mathsf{x}_{jk}\sum_{i=1}^{n}\mathsf{x}_{ik}) \\ &= 0 \end{aligned}$$

i.e., that the rows and columns of $\mathbf{Q}$ sum to zero.

2.1 An exact representation?

It will now be shown that given distances d_{ij}, then equation 1 uniquely determines a matrix $\mathbf{Q}$ whose rows and columns sum to zero. The demand that the d_{ij} satisfy the triangle inequality is unfortunately not enough to guarantee that this matrix will be positive definite, as is required to yield a configuration that can be exactly embedded in Euclidean space.

Set $A = \sum_{i=1}^{n} q_{ii}$. Summing $d_{ij} = q_{ii} + q_{jj} - 2q_{ij}$ over i, it follows that

$$\sum_{i=1}^{n} d_{ij}^2 = A + nq_{jj} \tag{2}$$

$$\sum_{j=1}^{n} d_{ij}^2 = A + nq_{ii} \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 = 2nA \tag{4}$$

From equation 4

$$A = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 \tag{5}$$

From equation 1, substituting for q_{ii} and q_{jj} from equations 2 and 3 above, and then for A from equation 5 above

$$\begin{aligned} q_{ij} &= -\frac{1}{2}d_{ij}^2 + \frac{1}{2n}\left(\sum_{i=1}^{n} d_{ij}^2 + \sum_{j=1}^{n} d_{ij}^2 - 2A\right) \\ &= -\frac{1}{2}d_{ij}^2 + \frac{1}{2n}\left(\sum_{i=1}^{n} d_{ij}^2 + \sum_{j=1}^{n} d_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2\right) \end{aligned}$$

Having thus recovered a symmetric matrix $\mathbf{Q}$, the spectral decomposition yields

$$\mathbf{Q} = \mathbf{U\Lambda U}^T$$

where $\mathbf{\Lambda}$ is a diagonal matrix. The diagonal elements λ_i are ordered so that

$$\lambda_1 \geq \lambda_2 \ldots \geq \lambda_n$$

An exact embedding is possible if and only if $\lambda_i \geq 0$ for all i. For this, set

$$\mathsf{X} = \mathbf{U\Lambda}^{\frac{1}{2}}$$

3 References

References

Gower, J. C. & Legendre, P. 1986. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification* **3:** 5-48.