Proximity in
Dimension Reduction,
Clustering, and Classification

Michael W. Trosset
Department of Statistics
Indiana University

December 16, 2018
Contents

I Preliminaries 9

1 Data 10
  1.1 Measurements ................................................................. 10
  1.2 Visualizing Low-Dimensional Data Matrices ................................. 11
  1.3 Proximities ................................................................. 12
    1.3.1 Dissimilarity .......................................................... 13
    1.3.2 Similarity .......................................................... 14
    1.3.3 A Provocative Example ............................................. 15
  1.4 Graphs ........................................................................ 17
    1.4.1 Examples .......................................................... 17
    1.4.2 Shortest Path Distance ............................................. 18
  1.5 Exercises .................................................................. 19

2 Euclidean Space 23
  2.1 Definition .................................................................. 23
  2.2 Orthogonality .............................................................. 24
  2.3 Spectral Decomposition ....................................................... 25
  2.4 Quadratic Forms .............................................................. 25
  2.5 Ellipsoids and the Singular Value Decomposition ............................. 27
  2.6 Euclidean Distance Geometry ................................................ 30
  2.7 Cosine Similarity .............................................................. 36
  2.8 Low-Distortion Embedding ..................................................... 37
  2.9 Exercises .................................................................. 39

3 Principal Component Analysis 42
  3.1 Principal Components as Coordinates ........................................... 42
    3.1.1 Hyperplanes .......................................................... 42
    3.1.2 Projection into Hyperplanes ............................................. 43
    3.1.3 Best Hyperplane Contains Centroid .................................... 44
    3.1.4 Centering the Data ..................................................... 45
    3.1.5 Rotating the Coordinate Axes ............................................. 46
    3.1.6 Best Linear Approximation ............................................. 47
    3.1.7 Squared Distances ..................................................... 48
  3.2 Principal Components as Variables ............................................... 50
    3.2.1 Covariance .......................................................... 50
    3.2.2 Linear Combinations ..................................................... 51
    3.2.3 Concentration Ellipsoids ................................................. 52
## CONTENTS

### 7 Manifold Learning

7.1 Manifolds .................................................. 96
  7.1.1 Differentiable Manifolds ............................... 96
  7.1.2 Riemannian Manifolds .................................. 98
  7.1.3 Manifold Learning ...................................... 98

7.2 Proximity Analysis of Data Profiles .......................... 98

7.3 Isomap .................................................. 99

7.4 Parametrization Recovery .................................. 99

7.5 Geodesic Representation ................................... 100
  7.5.1 Motivating Example ................................... 101
  7.5.2 Representing Swiss Rolls ............................... 102

7.6 Exercises ............................................ 102

### III Clustering

8 Agglomerative Hierarchical Clustering ....................... 108
  8.1 External Separation ...................................... 108
  8.2 Dendrograms ........................................... 109
  8.3 Exercises ............................................. 110

9 K-Means Clustering ........................................ 111
  9.1 Euclidean Formulation ................................... 111
  9.2 Algorithms .............................................. 112
    9.2.1 Alternating Least Squares ........................... 112
    9.2.2 Exchange ........................................... 113
    9.2.3 Stochastic Gradient Descent ......................... 115
  9.3 Self-Organizing Maps ................................... 118
  9.4 Kernel Formulation ...................................... 118
  9.5 Exercises ............................................. 119

10 Graph Partitioning ....................................... 122
  10.1 The Fiedler Vector ..................................... 122
  10.2 Several Eigenvectors .................................. 124
  10.3 Edge Cutting .......................................... 125
    10.3.1 Ratio Cuts ......................................... 126
    10.3.2 Normalized Cuts .................................... 131
  10.4 Exercises ............................................. 131

11 Probability Models for Clustering .......................... 132
  11.1 Label Model ........................................... 133
    11.1.1 Classification Likelihood Function .................. 133
    11.1.2 Model-Based Clustering ............................... 133
  11.2 Mixture Model ......................................... 134
    11.2.1 Finite Mixture Distributions ......................... 134
    11.2.2 Estimating Normal Mixture Parameters ............... 135
    11.2.3 Clustering Normal Mixture Samples ................... 137
  11.3 Exercises ............................................. 137
### IV Classification

12 Nearest Neighbor Classifiers
- 12.1 Nearest Neighbor Rules .......................... 144
- 12.2 Editing KNN Rules for Better Performance .......... 145
- 12.3 Editing KNN Rules for Greater Parsimony .......... 145
- 12.4 Distance Metric Learning ........................ 145

13 Linear Discriminant Analysis
- 13.1 Linear Classification .............................. 146
- 13.2 Fisher’s Best Linear Discriminator ................. 147
- 13.3 Discriminant Coordinates ........................ 147
- 13.4 Linear and Quadratic Discriminant Analysis ....... 150
- 13.5 Exercises ........................................ 152

14 Support Vector Machines

V Appendices

A Abstract Spaces ....................................... 155
B Matrices .................................................. 158
C Graphs .................................................... 159
List of Figures

1.1 An $\epsilon$-neighborhood graph. How well the graph approximates the underlying spiral depends on the choice of $\epsilon$. .................. 21
1.2 Two simple, undirected, weighted graphs. .................. 22

2.1 Embedding a Euclidean inner product matrix $B$. .................. 31
2.2 Two isometric configurations, designated $\heartsuit$ and $\spadesuit$. .................. 32

3.1 Constructing the $d$-dimensional principal component representation of $x_1, \ldots, x_n \in \mathbb{R}^q$. .................. 48
3.2 Latent semantic indexing finds the best linear subspace; principal component analysis finds the best affine linear subspace. .................. 56

4.1 Constructing a $d$-dimensional principal component representation from a Euclidean inner product matrix $C$. .................. 63
4.2 Constructing a $d$-dimensional principal component representation from a dissimilarity matrix $\Delta$ by classical multidimensional scaling (CMDS). .................. 67
4.3 The 2-dimensional principal component representation of 22 western states, constructed by CMDS from the dissimilarities in Table 4.1. The exact location of each state is the center of the rectangle that encloses the state’s postal abbreviation. .................. 72
4.4 An alternative 2-dimensional principal component representation of the 22 western states, identical to the representation in Figure 4.3 except that the orientation of the horizontal axis has been reversed. .................. 73
4.5 A slight rotation of the representation in Figure 4.4, resulting in a more familiar compass orientation. .................. 74
4.6 Out-of-sample embedding via classical multidimensional scaling. The objects $x_1, \ldots, x_n$ are embedded as $y_1, \ldots, y_n \in \mathbb{R}^d$. The out-of-sample object $x$ is then embedded as $y \in \mathbb{R}^d$. .................. 75
4.7 The out-of-sample embedding discussed in Example 4.3. Objects $\bullet$ and $\circ$ are fixed, with an out-of-sample object nearest the former. Out-of-sample embedding based on the $\bullet$ can produce $\bullet$ or $\bigcirc$. .................. 75
4.8 Two 1-dimensional principal component representations constructed by CMDS from the cosine similarities in Example 4.4. The upper representation was constructed by embedding geodesic distances (angle dissimilarities) and preserves equal spacing; the lower representation was constructed by embedding chordal distances and distorts equal spacing. .................. 76

5.1 Landmark MDS. .................. 81
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>R functions for Guttman majorization. The user calls <code>mds.gma()</code> with an initial configuration (X), a dissimilarity matrix (Delta), a weight matrix (W), and (optionally) a fixed number of iterations (k).</td>
</tr>
<tr>
<td>6.1</td>
<td>Imputing missing dissimilarities by shortest path distance. The embedding on the left fits Euclidean distances to observed edge weights with no regard for other pairs; the embedding on the right fits Euclidean distances to shortest path distances. See Example 6.1.</td>
</tr>
<tr>
<td>6.2</td>
<td>An embedding of the shortest path distances for the ε-neighborhood graph displayed in Figure 1.1(b). This construction exemplifies the Isomap procedure for manifold learning.</td>
</tr>
<tr>
<td>6.3</td>
<td>Constructing a d-dimensional principal component representation of a graph by a combinatorial Laplacian eigenmap.</td>
</tr>
<tr>
<td>6.4</td>
<td>Two embeddings of the ε-neighborhood graph displayed in Figure 1.1(b): (a) CMDS of shortest path distances derived from unit edge weights; (b) Combinatorial Laplacian eigenmap derived from unit edge weights.</td>
</tr>
<tr>
<td>7.1</td>
<td>Isomap, the manifold learning procedure proposed in [38].</td>
</tr>
<tr>
<td>7.2</td>
<td>CMDS embedding of 200 equally spaced points on a closed curve of length 2. The matrix of pairwise arc distances, hence the embedding, does not depend on the shape of the curve.</td>
</tr>
<tr>
<td>8.0</td>
<td>Unsupervised versus supervised learning. These n = 18 feature vectors in ( \mathbb{R}^2 ) have been labelled blue or red. Unsupervised: absent the labels, the horizontal axis explains more total variation and better separates the data into two clusters with good internal cohesion and external separation. Supervised: the vertical axis better discriminates red objects from blue objects.</td>
</tr>
<tr>
<td>9.1</td>
<td>Lloyd’s algorithm for k-means clustering.</td>
</tr>
<tr>
<td>9.2</td>
<td>Exchange algorithms for k-means clustering. Different ways of choosing x define different exchange algorithms.</td>
</tr>
<tr>
<td>9.3</td>
<td>SOM algorithms for k-means clustering. Different ways of updating ( \alpha ) define different SOM algorithms.</td>
</tr>
<tr>
<td>9.4</td>
<td>A scatter diagram of ( N = 1000 ) points in ( \mathbb{R}^2 ).</td>
</tr>
<tr>
<td>10.1</td>
<td>If ( \epsilon \in (0.75, 1) ) or ( \epsilon \in (0, 0.60) ), then applying 2-means clustering to the 3-dimensional combinatorial eigenmap produces the partition that optimizes the ratio cut criterion. If ( \epsilon \in (0.60, 0.75) ), then these techniques produce different partitions.</td>
</tr>
<tr>
<td>10.2</td>
<td>An undirected graph with similarity-weighted edges.</td>
</tr>
<tr>
<td>11.1</td>
<td>An algorithm for model-based clustering. Each cluster is identified with a multivariate normal distribution.</td>
</tr>
<tr>
<td>11.2</td>
<td>A finite mixture experiment. In the case of supervised learning (Part IV), both ( i ) and ( x ) are observed; in the case of unsupervised learning (Chapter 11), only ( x ) is observed.</td>
</tr>
<tr>
<td>11.3</td>
<td>The EM algorithm for normal mixture experiments.</td>
</tr>
<tr>
<td>12.1</td>
<td>The k nearest neighbor (KNN) classifier. The choice of k is often made by leave-one-out cross-validation on a training sample.</td>
</tr>
</tbody>
</table>
13.1 Representing multivariate data in discriminant coordinates. . . . . . . . . . . 151
List of Tables

4.1 Dissimilarities of $n = 22$ western states. Each $\delta_{ij}$ is the minimal number of state boundaries that must be crossed to drive from state $i$ to state $j$. . . . . . 68

9.1 Five-number summaries of 1000 values of $W$ returned by each of three algorithms for 20-means clustering the 1000 points in Figure 9.4. The $q_i$ are the sample quartiles. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
Part I

Preliminaries
Chapter 1

Data

This book describes a variety of methods used to explore the structure of high-dimensional data. The present chapter surveys the fundamental data structures with which we shall be concerned.

1.1 Measurements

In classical statistics, *multivariate analysis* is concerned with situations in which $q$ numerical measurements are taken on $n$ experimental units. We will variously refer to the experimental units as individuals, subjects, and objects. We will variously refer to what we measure as variables and features.

Let $x_{ij}$ denote the numerical value of measurement $j$ on experimental unit $i$. We organize the measurements in the $n \times q$ rectangular array

$$X = [x_{ij}] = \begin{bmatrix} x_{11} & \cdots & x_{1q} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{nq} \end{bmatrix}.$$  

This array is the *data matrix*.

Rows of the data matrix correspond to objects; columns of the data matrix correspond to variables. A single row of $X$ is called a *subject profile* or *feature vector*; a single column of $X$ is called a *variable profile*. A feature vector is an ordered $q$-tuple of real numbers, hence a point in $q$-dimensional space: $x_i = (x_{i1}, \ldots, x_{iq}) \in \mathbb{R}^q$. By convention, vectors are represented in matrix notation as rectangular arrays with a single column, i.e., as column vectors. Thus, we represent feature vector $i$ as

$$x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{iq} \end{bmatrix}$$

and row $i$ of the data matrix is its transpose:

$$X = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix}.$$
Example 1.1 Suppose that we survey $n = 4$ film buffs, observing whether or not each individual has or has not seen each of the $q = 6$ feature films directed by Sergio Leone.\(^1\) Let $x_{ij} = 1$ if individual $i$ has seen film $j$ and let $x_{ij} = 0$ if s/he has not. Then we might observe the following data matrix:

$$X = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
\end{bmatrix}. \quad (1.1)$$

Classical multivariate analysis is usually concerned with situations in which $n \gg q$. Although Example 1.1 describes a small data set, the fact that $q > n$ evokes the nature of high-dimensional data.

1.2 Visualizing Low-Dimensional Data Matrices

Let $X$ denote a data matrix, with measurements $V_1, \ldots, V_q$ taken on each of $n$ objects. A 2-dimensional scatter diagram or scatterplot displays the data in Cartesian coordinates with respect to two of the variables (features), say $V_j$ and $V_k$. Scatterplots can be constructed using the R function `plot`. Suppose that the $n \times 2$ data matrix $X$ has columns $x$ and $y$:

```r
> X <- cbind(x,y)
```

Then either of the following commands construct a scatterplot with $x$ on the horizontal axis and $y$ on the vertical axis:

```r
> plot(X)
> plot(x,y)
```

Various options allow one to control the axis scales and labels, the plotting symbol, etc.

A scatterplot displays how the data appears after projection into the 2-dimensional subspace spanned by $V_j$ and $V_k$. Depending on what one is interested in learning about the data and the relevance of $V_j$ and $V_k$ to one’s concerns, this scatterplot may or may not reveal interesting structure in $X$. Often, the process of exploring multivariate data means the process of searching for interesting scatterplots.

If $q$ is not too large, then one can use the R function `pairs` to form a scatterplot matrix that simultaneously displays the scatterplots for each $(V_j, V_k)$. A scatterplot matrix may help one to discern interesting pairs of variables, but the best views of multivariate structure usually involve more than two of the original variables. Instead of projecting the data into the 2-dimensional subspaces spanned by pairs of original variables, it is often desirable to consider more general projections. For example, one might construct a new coordinate system by computing new variables $W_1, \ldots, W_q$, e.g., by

\[
\begin{align*}
W_1 &= a_{11}V_1 + \cdots + a_{1q}V_q \\
\vdots \\
W_q &= a_{q1}V_1 + \cdots + a_{qq}V_q,
\end{align*}
\]

\(^1\) A Fistful of Dollars (1964), For a Few Dollars More (1965), The Good, the Bad, and the Ugly (1966), Once Upon a Time in the West (1968), Duck, You Sucker! (1972), and Once Upon a Time in America (1984).
then represent the data with respect to two of the new variables. One of the fundamental challenges of multivariate data analysis is to find coordinate systems that most readily reveal the high-dimensional structure of one’s data.

One way to examine a great many 2-dimensional projections of the data is to perform a grand tour. Doing so requires specialized software, e.g., GGobi or CrystalVision. A grand tour follows a parameterized path through the space of possible projections, allowing the data analyst to observe representations of the data that continuously change. When an interesting representation is encountered, the grand tour is paused.

Grand tours are cumbersome when \( q \) is even moderately large. Various statistical procedures, e.g., principal component analysis and linear discriminant analysis, can be viewed as attempts to automate the search for interesting projections, an activity called projection pursuit.

Visualization need not be limited to 2-dimensional scatterplots. Exploiting the methods of projective geometry, one can represent multivariate data in parallel coordinates. However, while the use of parallel coordinates permits one to visualize data in several dimensions, it is easy to be overwhelmed if one tries to use too many dimensions at one time.

The standard tools for visualizing multivariate data work best when the number of dimensions is not too large. When \( q \) is large, we will not attempt to apply these tools directly. Instead, we will first apply techniques for dimension reduction. The basic idea of dimension reduction is to replace the observed \( n \times q \) data matrix with a derived \( n \times p \) data matrix. Typically, \( p \ll q \) and one hopes that the derived data matrix preserves (most of) the interesting structure in the observed data matrix.

So far we have emphasized the use of data matrices. Data matrices are the most common form of multivariate data and most available procedures operate directly on data matrices. If we observe (or compute from a data matrix) a proximity matrix or a graph, then we may need to extract a data matrix from it. The process of extracting data matrices from proximities and/or graphs is called embedding and is a topic to which we will devote considerable attention.

### 1.3 Proximities

Proximities are real numbers that convey pairwise information about the extent to which two objects resemble each other. If larger proximity indicates that objects are less alike, then we say that the proximities measure dissimilarity and denote the proximity of pair \( i \sim j \) by \( \delta_{ij} = \delta_{ji} \). If larger proximity indicates that objects are more alike, then we say that the proximities measure similarity and denote the proximity of pair \( i \sim j \) by \( \gamma_{ij} = \gamma_{ji} \). By convention, one usually requires proximities to be nonnegative.

A complete set of proximities comprises real numbers for each of the \( \binom{n}{2} = n(n - 1)/2 \) pairs for which \( i \neq j \), together with real numbers for each of the \( n \) pairs for which \( i = j \). We require that \( \delta_{ij} \geq \delta_{ii} = 0 \) and that \( \gamma_{ij} \leq \gamma_{ii} \). Some authors further require \( \gamma_{ii} = 1 \). Although doing so involves some redundancy, it is often convenient to organize proximities into matrices, \( \Delta = [\delta_{ij}] \) and \( \Gamma = [\gamma_{ij}] \).

**Example 1.2** We might measure the similarity of pairs of film buffs in Example 1.1 by computing simple matching coefficients. The matching coefficient for pair \( i \sim j \) is the proportion
of films that both \(i\) and \(j\) either have or have not seen. Thus,

\[
\gamma_{ij} = \frac{1}{6} \sum_{k=1}^{6} \left\{ \begin{array}{ll}
1 & \text{if } x_{ik} = x_{jk} \\
0 & \text{if } x_{ik} \neq x_{jk}
\end{array} \right.
\]

(1.2)

and

\[
\Gamma = [\gamma_{ij}] = \frac{1}{6} \begin{bmatrix}
6 & 5 & 4 & 2 \\
5 & 6 & 5 & 3 \\
4 & 5 & 6 & 2 \\
2 & 3 & 2 & 6
\end{bmatrix}.
\]

1.3.1 Dissimilarity

The notion of dissimilarity is motivated by the formal concept of distance. For example, the Euclidean distance between \(x_i\) and \(x_j\) is

\[
d_{ij} = d(x_i, x_j) = \left[ (x_{i1} - x_{j1})^2 + \cdots + (x_{iq} - x_{jq})^2 \right]^{1/2} = \left[ \sum_{k=1}^{q} (x_{ik} - x_{jk})^2 \right]^{1/2}.
\]

(1.3)

Given a data matrix \(X\), the matrix of pairwise Euclidean distances, \(D = [d_{ij}]\), can be computed in \(R\) as follows:

\[
> D \leftarrow \text{as.matrix(dist}(X))
\]

If \(\Delta\) is a matrix of pairwise distances, then the following properties necessarily obtain.

1. For each \(ij\), \(\delta_{ij} \geq 0\), i.e., \(\Delta\) is nonnegative.

2. For each \(i\), \(\delta_{ii} = 0\), i.e., the diagonal entries of \(\Delta\) vanish, i.e., \(\Delta\) is hollow.

3. For each \(ij\), \(\delta_{ij} = \delta_{ji}\), i.e., \(\Delta = \Delta^t\), i.e., \(\Delta\) is symmetric.

We shall refer to these properties as the delta properties and use them to provide a formal definition of dissimilarity.

**Definition 1.1** The \(n \times n\) matrix \(\Delta\) is a dissimilarity matrix iff it satisfies the delta properties, i.e., iff \(\Delta\) is hollow, nonnegative, and symmetric.

We emphasize that Euclidean distance is but one way to measure the dissimilarity of feature vectors. The mere fact that \(x_1, \ldots, x_n \in \mathbb{R}^q\) means no more than that the variables of interest are measured by real numbers. It does not follow that Euclidean distance is the most meaningful way to measure the dissimilarity of experimental units. However, Euclidean distance is a type of dissimilarity that most people find fairly intuitive, and a great many statistical techniques have been developed with Euclidean distance in mind. For these reasons, we will often endeavor to approximate non-Euclidean dissimilarities with Euclidean distances.
1.3.2 Similarity

The delta properties imply that $\delta_{ii} \leq \delta_{ij}$ for every $j$. To pass from dissimilarity ($\delta$) to similarity ($\gamma$), we reverse this requirement: $\gamma_{ii} \geq \gamma_{ij}$. Typically one also requires $\gamma_{ij} \geq 0$. Sometimes one fixes a common value of self-similarity, e.g., $\gamma_{ii} = 1$.

**Definition 1.2** The $n \times n$ matrix $\Gamma = [\gamma_{ij}]$ is a similarity matrix iff $\Gamma$ is symmetric and $\gamma_{ii} \geq \gamma_{ij} \geq 0$ for each $ij$.

**Example 1.3** For any $h > 0$, let

$$\gamma_{ij} = \exp\left(-hd_{ij}^2\right).$$

This measure of similarity has been widely used in manifold learning, a collection of techniques for nonlinear dimension reduction. The similarity matrix $\Gamma = [\gamma_{ij}]$ is sometimes called the heat kernel.

**Example 1.4** Let

$$\langle x_i, x_j \rangle = x_{i1}x_{j1} + \cdots + x_{iq}x_{jq}$$

denote the dot product of feature vectors $x_i$ and $x_j$. Assume that each $\langle x_i, x_j \rangle \geq 0$, e.g., because the features are nonnegative. Let

$$\|x_i\| = \left(x_{i1}^2 + \cdots + x_{iq}^2\right)^{1/2}$$

denote the length of feature vector $x_i$. Then

$$\gamma_{ij} = \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|}$$

is the cosine of the angle between the vectors $x_i$ and $x_j$, a measure of similarity that is widely used in text mining (and elsewhere).

While dissimilarities are naturally modeled by distances, it is less obvious how to model similarity. Example 1.4 suggests the possibility of modeling similarities by dot products. In fact, this is the conventional mathematical model of similarity in machine learning. As we shall see, however, modeling similarities with dot products may prove problematic. Alternatively, we might transform the similarities to dissimilarities. The transformation of similarity to dissimilarity is a major concern of this book. That such transformations may require considerable care can be discerned from a careful study of our hypothetical film buffs.
1.3.3 A Provocative Example

In Example 1.2, we proposed measuring $\gamma_{ij}$, the similarity of individuals $i$ and $j$, by computing the proportion of films that either both have or both have not seen. With equal plausibility, we might measure $\delta_{ij}$, the dissimilarity of individuals $i$ and $j$, by computing the proportion of films that one of the individuals has seen and the other has not, i.e., $\delta_{ij} = 1 - \gamma_{ij}$. Doing so results in the dissimilarity matrix

$$\Delta_2 = \left[ 1 - \gamma_{ij} \right] = \frac{1}{6} \begin{bmatrix} 0 & 1 & 2 & 4 \\ 1 & 0 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ 4 & 3 & 4 & 0 \end{bmatrix}.$$ 

It would seem that we should be completely indifferent to describing the proximities between our film buffs by $\Gamma$ or by $\Delta_2$. For example, there is no evident difference between saying that individuals 1 and 3 match on 4 of 6 films ($\gamma_{13} = 4/6$) or that they mismatch on 2 of 6 films ($\delta_{13} = 2/6$). However, there are important differences between the distances used to model dissimilarities and the dot products used to model similarities. This section illustrates that distinction.

We begin by inquiring if it is possible to represent the $n = 4$ film buffs as $y_1, y_2, y_3, y_4$ in such a way that $\delta_{ij} = d(y_i, y_j)$. The only way for $y_1, y_2, y_3$ to have interpoint distances

$$\frac{1}{6} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

is for $y_1, y_2, y_3$ to be collinear. A second dimension may be required for $y_4$. If we set $y_1 = (-1/6, 0), y_2 = (0, 0),$ and $y_3 = (1/6, 0),$ then $\delta_{41} = \delta_{43}$ implies that $y_4 = (0, a)$. Now $a = \pm \delta_{42} = \pm 3/6$, whereas the Pythagorean Theorem entails

$$\left(\frac{4}{6}\right)^2 = \delta_{41}^2 = \delta_{42}^2 + \delta_{21}^2 = a^2 + \left(\frac{1}{6}\right)^2$$

and therefore $a^2 = 15/36 \neq (\pm 3/6)^2$. We conclude that there is no configuration of points whose interpoint Euclidean distances equal $\Delta_2$.

We could proceed by approximating $\Delta_2$ with Euclidean distances, losing information in the process. Instead, we observe that the matching coefficients defined by (1.2) satisfy

$$\gamma_{ij} = \frac{1}{6} \left[ 6 - \sum_{k=1}^{6} (x_{ik} - x_{jk})^2 \right] = 1 - \sum_{k=1}^{6} \left( \frac{x_{ik}}{\sqrt{6}} - \frac{x_{jk}}{\sqrt{6}} \right)^2 = 1 - d^2 \left( \frac{x_i}{\sqrt{6}}, \frac{x_j}{\sqrt{6}} \right).$$

This observation invites us to transform similarity to dissimilarity not by $\delta_{ij} = 1 - \gamma_{ij}$ but rather by $\delta_{ij}^2 = 1 - \gamma_{ij}$, thereby ensuring that the dissimilarity matrix

$$\Delta_1 = \left[ \sqrt{1 - \gamma_{ij}} \right] = \frac{1}{\sqrt{6}} \begin{bmatrix} 0 & 1 & \sqrt{2} & 2 \\ 1 & 0 & 1 & \sqrt{3} \\ \sqrt{2} & 1 & 0 & 2 \\ 2 & \sqrt{3} & 2 & 0 \end{bmatrix}.$$
is a matrix of Euclidean interpoint distances. In fact, $\Delta_1$ contains the interpoint distances of the points $x_i/\sqrt{6}$, but we need not resort to $q = 6$ dimensions to represent $n = 4$ points. For example, $\Delta_1$ is also generated by the rows of

$$Y = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sqrt{3} \end{bmatrix}.$$ 

Now we return to the similarities and attempt to represent the $\gamma_{ij}$ as dot products of $z_1, z_2, z_3, z_4$. Although 3 dimensions suffice to represent 4 points, in this case it is convenient to work in 4 dimensions.

First, assuming that $z_1 = (a, 0, 0, 0)$, we solve

$$\langle z_1, z_1 \rangle = a^2 = \gamma_{11} = 1$$

to obtain and $z_1 = (1, 0, 0, 0)$.

Next, assuming that $z_2 = (a, b, 0, 0)$, we solve

$$\langle z_2, z_1 \rangle = a = \gamma_{21} = 5/6$$

and then

$$\langle z_2, z_2 \rangle = a^2 + b^2 = \gamma_{22} = 1$$

to obtain $z_2 = (5/6, \sqrt{11}/6, 0, 0)$.

Continuing in this manner, we obtain

$$Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{5}{6} & \frac{\sqrt{11}}{6} & 0 & 0 \\ \frac{5}{3} & \frac{\sqrt{11}}{3} & \frac{\sqrt{11}}{3} & 0 \\ \frac{3}{3} & \frac{\sqrt{11}}{3} & -\frac{3}{110} & \frac{7}{10} \end{bmatrix},$$

where $z_i$ is row $i$ of $Z$. The matrix $Z$ is a Cholesky factor of the matrix $\Gamma$ and can be computed (in finite precision arithmetic) by the following R command:

```
> Z <- t(chol(Gamma))
```

We have succeeded in representing the similarities $\gamma_{ij}$ as dot products of $z_1, z_2, z_3, z_4$, i.e.,

$$\Gamma = [\gamma_{ij}] = [\langle z_i, z_j \rangle] = [z_i^t z_j] = ZZ^t.$$ 

Furthermore, some calculation reveals that

$$D_2(Z) = \left[ d^2 (z_i, z_j) \right] = 2\Delta_2,$$

suggesting a relationship between dot products and squared Euclidean distances. We will formalize that relationship in Section 2.6.

The preceding discussion was intended to pose questions, not provide answers. Ideally, it should leave one feeling uncertain, prepared to think critically about the relationship between and the interpretations of similarity and dissimilarity.
1.4 Graphs

Graphs are data structures that may be observed (as in the case of social networks) or derived from other data structures. Informally, a graph is a pair $G = (V, E)$ of vertices and edges. See Appendix C for formal definitions of graphs and related concepts.

We will use vertices to denote objects and edge weights to record and/or construct proximities between objects. For this reason, it often will be imperative to distinguish between edge weights that represent similarity (larger $w_{ij}$ indicate greater similarity of $v_i$ and $v_j$) and edge weights that represent dissimilarity (larger $w_{ij}$ indicate greater dissimilarity of $v_i$ and $v_j$). If $G$ is unweighted, then we usually will interpret $G$ as a weighted graph with unit edge weights. In such cases, we can reasonably interpret $w_{ij} = 1$ as measuring either similarity or dissimilarity.

1.4.1 Examples

Example 1.5 Let $V$ denote a set of $n$ Facebook users. Include an undirected edge between $v_i$ and $v_j$ iff users $i$ and $j$ are friends. Then $G$ is a simple graph.

Example 1.6 Let $V$ denote a set of Verizon cell phone users. Include a directed edge between $v_i$ and $v_j$, $e_{ij}$, iff $i$ called $j$ within the last month. Weight $e_{ij}$ by the number of times that $i$ called $j$ in the last month. Then $G$ is a weighted digraph and $\gamma_{ij} = w_{ij} + w_{ji}$ is a measure of similarity between $i$ and $j$.

Example 1.7 Let $V$ denote a set of $n$ objects and let $E$ denote the set of unordered pairs of objects for which the proximity is known. Let $w_{ij}$ denote the proximity between objects $i$ and $j$. (Thus, any proximity matrix can be represented as a weighted graph.) For example, suppose that the vertices are heavy (not hydrogen) atoms in a molecule. Include an edge between $v_i$ and $v_j$ iff atoms $i$ and $j$ are bonded to each other and set $w_{ij}$ equal to the length of the bond.

Example 1.8 Let $V$ denote a set of feature vectors. Given $\epsilon > 0$, include an undirected edge between distinct $v_i$ and $v_j$ iff $d(x_i, x_j) \leq \epsilon$. Then $G$ is a simple graph, sometimes called an $\epsilon$-neighborhood graph. We might weight the edges of $G$ with a measure of proximity, e.g., $w_{ij} = \delta_{ij} = d(x_i, x_j)$ or $w_{ij} = \gamma_{ij} = \exp(-td^2(x_i, x_j))$.

Consider, for example, the $n = 201$ feature vectors plotted in Figure 1.1(a). These data were sampled (with error) from a spiral in $\mathbb{R}^2$. For $\epsilon = 0.5$, an $\epsilon$-neighborhood graph of these data is displayed in Figure 1.1(b). The graph neatly captures the shape of the underlying spiral. Notice, however, that too small a value of $\epsilon$ would result in a disconnected graph, whereas too large a value of $\epsilon$ would result in a graph that obscures the spiral shape.
Example 1.9 Let $V$ denote a set of feature vectors. Given a measure of proximity and a natural number $K$, include a directed edge from $v_i$ to $v_j$ if $x_j$ is one of the $K$ nearest neighbors of $x_i$. Then $\mathcal{G}$ is a simple digraph, sometimes called a $K$-nearest neighbor graph. As in Example 1.8 we might weight each directed edge by $w_{ij} = d(x_i, x_j)$ or $w_{ij} = \exp(-td^2(x_i, x_j))$, but $W$ so defined need not be a proximity matrix because it need not be symmetric.

1.4.2 Shortest Path Distance

The concept of shortest path distance implicitly assumes that we are interpreting edge weights as dissimilarities. If $G = (V, E)$ is a weighted graph, then the length of a path in $\mathcal{G}$ is the sum of its edge weights. Because there are finitely many paths from $v_i$ to $v_j$, there must be a shortest path from $v_i$ to $v_j$. If $\mathcal{G}$ is undirected, then the length of the shortest path from $v_i$ to $v_j$ is also the length of the shortest path from $v_j$ to $v_i$ and this common quantity is the shortest path distance between $v_i$ and $v_j$. If $\mathcal{G}$ is directed, then the shortest path distance between $v_i$ and $v_j$ is the average (or sum) of the shortest path distance from $v_i$ to $v_j$ and the shortest path distance from $v_j$ to $v_i$. The following result justifies the terminology.

**Theorem 1.1** Let $G = (V, E)$ be a simple connected graph with strictly positive edge weights. Define $d : V \times V \to \mathbb{R}$ by letting $d(v_1, v_2)$ equal the length of the shortest path from $v_1$ to $v_2$ if $v_1 \neq v_2$ and setting $d(v_1, v_2) = 0$ if $v_1 = v_2$. Define $\bar{d} : V \times V \to \mathbb{R}$ by

$$\bar{d}(v_1, v_2) = \left[ d(v_1, v_2) + d(v_2, v_1) \right] / 2.$$ 

Then $\bar{d}$ is a distance function on $V$, called the shortest path distance on $\mathcal{G}$.

**Proof:** Because the edge weights are strictly positive, $\bar{d}(v_1, v_2) \geq 0$ with equality iff $v_1 = v_2$. By construction, $\bar{d}(v_1, v_2) = d(v_2, v_1)$. It remains only to verify the triangle inequality.

Let $P(v_1, v_3)$ denote the set of paths from $v_1$ to $v_3$ and let $P(v_1, v_3|v_2)$ denote the set of paths from $v_1$ to $v_3$ that contain $v_2$ as an intermediate vertex. Then $d(v_1, v_3)$ is the minimum length of all paths in $P(v_1, v_3)$ and $d(v_1, v_2) + d(v_2, v_3)$ is the minimum length of all paths in $P(v_1, v_3|v_2)$. Because $P(v_1, v_3|v_2) \subset P(v_1, v_3)$,

$$d(v_1, v_3) \leq d(v_1, v_2) + d(v_2, v_3). \tag{1.4}$$

By the same reasoning,

$$d(v_3, v_1) \leq d(v_3, v_2) + d(v_2, v_1). \tag{1.5}$$

Summing (1.4) and (1.5), we obtain

$$\bar{d}(v_1, v_3) \leq \bar{d}(v_1, v_2) + \bar{d}(v_2, v_3).$$

The following examples illustrate two important facts about shortest path distance. For simplicity, assume that $\mathcal{G}$ is undirected.
CHAPTER 1. DATA

Example 1.10 Consider the graph displayed on the left side of Figure 1.2 and notice that its edge weights do not obey the triangle inequality. The shortest path from $v_1$ to $v_3$ is $v_1 \leftrightarrow v_2 \leftrightarrow v_2$ (length $1 + 1 = 2$), not $v_1 \leftrightarrow v_3$ (length 3). Thus, if we begin with a partial set of pairwise dissimilarities and attempt to construct a complete set of dissimilarities by computing shortest path distance, the process may overwrite some of the original dissimilarities.

Example 1.11 Consider the graph displayed on the right side of Figure 1.2. The matrix of pairwise shortest path distances is

$$D = \begin{bmatrix} 0 & 1 & 2 & 4 \\ 1 & 0 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ 4 & 3 & 4 & 0 \end{bmatrix}.$$  

Notice that $D$ is a scalar multiple of the dissimilarity matrix $\Delta_2$ in Section 1.3.3, which we proved is not a matrix of Euclidean interpoint distances. In general, therefore, approximating shortest path distances by Euclidean distances will entail a loss of information.

1.5 Exercises

1. A basic R installation contains a variety of data sets. The iris data set contains $q = 4$ measurements (sepal length, sepal width, petal length, petal width) on $n = 150$ irises, 50 each of three different species. These measurements were made by the American botanist Edgar Anderson and famously analyzed by Sir Ronald Fisher in his pioneering study of linear discriminant analysis. The following R command produces a scatterplot matrix in which iris species is coded by color (red = setosa, green = versicolor, blue = virginica).

   ```r
   > pairs(iris[1:4], main="Anderson’s Iris Data", pch=21, 
   + bg=c("red","green3","blue")[unclass(iris$Species)])
   ```

Examine the scatterplot matrix and discuss the extent to which one might distinguish iris species on the basis of these four measurements.

2. Consider the following $n = 11$ states:

   1. Illinois (IL)
   2. Indiana (IN)
   3. Kentucky (KY)
   4. Maryland (MD)
   5. Michigan (MI)
   6. Ohio (OH)
   7. Pennsylvania (PA)
   8. Tennessee (TN)
   9. Virginia (VA)
  10. West Virginia (WV)
  11. Wisconsin (WI)
Define a measure of dissimilarity between states by setting \( \delta(i, j) \) equal to the minimum number of state boundaries that must be crossed to drive from state \( i \) to state \( j \).

Compute the dissimilarity matrix \( \Delta = [\delta(i, j)] \).

Hint: You may need a map to complete this assignment!

3. Verify that the function \( d : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R} \) defined by (1.3) is a distance function in the formal sense of Appendix A.

4. For each of the following dissimilarity matrices, and for both \( d = 2 \) and \( d = 3 \), explain why it is or is not possible to find \( x_1, \ldots, x_4 \in \mathbb{R}^d \) such that each \( \delta_{ij} = \| x_i - x_j \|_2 \).

   \[
   (a) \, \Delta = \begin{bmatrix}
   0 & 2 & 2 & 1.1 \\
   2 & 0 & 2 & 1.1 \\
   2 & 2 & 0 & 1.1 \\
   1.1 & 1.1 & 1.1 & 0
   \end{bmatrix}
   
   (b) \, \Delta = \begin{bmatrix}
   0 & 1 & 2 & \sqrt{2} \\
   1 & 0 & \sqrt{3} & 1 \\
   2 & \sqrt{3} & 0 & \sqrt{5} \\
   \sqrt{2} & 1 & \sqrt{5} & 0
   \end{bmatrix}
   \]

5. The similarity matrix

   \[
   \Gamma = \begin{bmatrix}
   1.0000 & 0.1054 & 0.0019 & 0.0183 \\
   0.1054 & 1.0000 & 0.0183 & 0.0019 \\
   0.0019 & 0.0183 & 1.0000 & 0.1054 \\
   0.0183 & 0.0019 & 0.1054 & 1.0000
   \end{bmatrix}
   \]

was derived from feature vectors \( x_1, x_2, x_3, x_4 \in \mathbb{R}^2 \). Consider the following two transformations from similarity to dissimilarity:

   (a) \( \delta_{ij} = 1 - \gamma_{ij} \)

   (b) \( \delta_{ij}^2 = -\log(\gamma_{ij}) \)

For each of the resulting \( \Delta = [\delta_{ij}] \), investigate whether or not there exists a 2-dimensional configuration of points whose interpoint distance matrix equals (or closely approximates) \( \Delta \).
Figure 1.1: An $\epsilon$-neighborhood graph. How well the graph approximates the underlying spiral depends on the choice of $\epsilon$. 
Figure 1.2: Two simple, undirected, weighted graphs.
Chapter 2

Euclidean Space

Many of the methods discussed in this book attempt to construct low-dimensional Euclidean representations of high-dimensional Euclidean or non-Euclidean data. Many more methods assume that data has been so represented. For these reasons, a thorough understanding of Euclidean space is essential to our program of study.

2.1 Definition

Let $\mathbb{R}^k$ denote the real vector space of ordered $k$-tuples of real numbers. The origin of $\mathbb{R}^k$ is the zero vector, $\vec{0} = (0, \ldots, 0)$, and $e = (1, \ldots, 1)$. Addition of vectors is defined by

$$(x_1, \ldots, x_k) + (y_1, \ldots, y_k) = (x_1 + y_1, \ldots, x_k + y_k)$$

and multiplication of a vector by a scalar $\alpha \in \mathbb{R}$ is defined by

$$\alpha (x_1, \ldots, x_k) = (\alpha x_1, \ldots, \alpha x_k).$$

For $i = 1, \ldots, k$, let $e_i \in \mathbb{R}^k$ denote the $k$-tuple defined by specifying entry $i$ to be 1 and all other entries to be 0. Then

$$\alpha_1 e_1 + \cdots + \alpha_k e_k = \vec{0} \text{iff } \alpha_1 = \cdots = \alpha_k = 0,$$

i.e., the vectors $e_1, \ldots, e_k$ are linearly independent, and any $x = (x_1, \ldots, x_k) \in \mathbb{R}^k$ can be expressed as a linear combination of $e_1, \ldots, e_k$:

$$x = x_1 e_1 + \cdots + x_k e_k \quad (2.1)$$

Thus, the vectors $e_1, \ldots, e_k$ are a basis for $\mathbb{R}^k$ and the coefficients of $x$ with respect to this basis are the Cartesian coordinates of $x$. In matrix notation, the Cartesian coordinates of $x$ are represented as the entries in a $k \times 1$ column vector and (2.1) becomes

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + x_k \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

Notice that $e = e_1 + \cdots + e_k$. 

23
By Euclidean space we mean $\mathbb{R}^k$ together with a function $\langle \cdot, \cdot \rangle$ that assigns real numbers to pairs of vectors as follows:

$$\langle x, y \rangle = x^t y = \begin{bmatrix} x_1 & \cdots & x_k \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = x_1y_1 + \cdots + x_ky_k.$$ 

The function $\langle \cdot, \cdot \rangle$ is the *dot product*. It is easily checked that the dot product is an inner product; hence, the real-valued function $d(\cdot, \cdot)$ defined by

$$d(x, y) = \|x - y\|_2 = \langle x - y, x - y \rangle^{1/2} = \left( (x - y)^t (x - y) \right)^{1/2}$$

is a metric/distance function. The quantity $d(x, y)$ is the Euclidean distance between $x$ and $y$ and the length of $x$ is its Euclidean distance from the origin:

$$d(x, 0) = \|x - 0\|_2 = \|x\|_2.$$ 

Finally, the angle $\theta$ between nonzero vectors $x$ and $y$ is defined implicitly by the equation

$$\cos \theta = \left( \frac{x}{\|x\|_2}, \frac{y}{\|y\|_2} \right) = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2}.$$ 

### 2.2 Orthogonality

Two vectors $x, y \in \mathbb{R}^k$ are *orthogonal* iff $\langle x, y \rangle = 0$. If two nonzero vectors are orthogonal, then the cosine of the angle $\theta$ between them equals zero and $\theta = \pi/2$, i.e., the vectors are perpendicular. A set of vectors is orthogonal iff each pair of vectors in the set is orthogonal. A set of vectors is orthonormal iff it is orthogonal and each vector has unit length. For example, $e_1, \ldots, e_k$ is an orthonormal basis for $\mathbb{R}^k$.

The square matrix $Q$ is orthogonal iff its column vectors are orthonormal. Because the $ij$ entry of $Q^t Q$ is the dot product of columns $i$ and $j$, $Q$ is orthogonal iff $Q^t Q = I$. Furthermore, because the columns of $Q$ are linearly independent, $Q$ is invertible. By the uniqueness of inverses, it follows that $Q$ is orthogonal iff $Q^{-1} = Q^t$. Reversing this chain of reasoning, $I = QQ^{-1} = QQ^t$, so $Q$ is orthogonal iff $QQ^t = I$ iff the row vectors of $Q$ are orthonormal.

If $Q$ is orthogonal, then the transformation $x \mapsto Q^t x$ has several appealing properties:

1. It is invertible. If $y = Q^t x$, then $Qy = QQ^t x = x$.

2. It preserves inner products. If $y_i = Q^t x_i$, then

$$\langle y_1, y_2 \rangle = \langle Q^t x_1, Q^t x_2 \rangle = (Q^t x_1)^t (Q^t x_2) = x_1^t QQ^t x_2 = x_1^t x_2 = \langle x_1, x_2 \rangle.$$ 

3. It preserves lengths. If $y = Q^t x$, then

$$\|y\|_2^2 = \langle y, y \rangle = \langle x, x \rangle = \|x\|_2^2.$$ 

4. It preserves angles. Suppose that $y_i = Q^t x_i$. Let $\theta$ denote the angle between $x_1$ and $x_2$; let $\phi$ denote the angle between $y_1$ and $y_2$. Then

$$\cos \phi = \frac{\langle y_1, y_2 \rangle}{\|y_1\|_2 \|y_2\|_2} = \frac{\langle x_1, x_2 \rangle}{\|x_1\|_2 \|x_2\|_2} = \cos \theta.$$ 

In consideration of these properties, orthogonal matrices are often called rotation matrices.
2.3 Spectral Decomposition

Let $B$ be a symmetric $k \times k$ matrix. If $\lambda \in \mathbb{R}$ and a nonzero $v \in \mathbb{R}^k$ satisfy the matrix equation $Bv = \lambda v$, then $\lambda$ is an eigenvalue of $B$ and $v$ is an eigenvector of $B$ that corresponds to $\lambda$. The set of all eigenvectors that correspond to $\lambda$ is a linear subspace of $\mathbb{R}^k$ and the dimension of this subspace is the (geometric) multiplicity of $\lambda$. If $(\lambda, v)$ and $(\mu, w)$ are two eigenpairs, then

$$\lambda v^t w = (\lambda v)^t w = (Bv)^t w = v^t Bw = v^t (\mu w) = \mu v^t w.$$ 

It follows that, if $\lambda \neq \mu$, then $\langle v, w \rangle = 0$.

It is not obvious that eigenpairs actually exist, but in fact we have the following result.

**Theorem 2.1** Let $B$ be a symmetric $k \times k$ matrix. Then there exists an orthonormal basis for $\mathbb{R}^k$, $v_1, \ldots, v_k$, such that each $v_i$ is an eigenvector of $B$.

Let $v_1, \ldots, v_k$ denote an orthonormal basis of eigenvectors and let $\lambda_1, \ldots, \lambda_k$ denote their corresponding eigenvalues. Following convention, we assume that the eigenpairs have been labelled so that $\lambda_1 \geq \cdots \geq \lambda_k$. Letting

$$V = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix} \quad \text{and} \quad \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix},$$

we obtain the spectral decomposition of $B$:

$$V \Lambda V^t = \sum_{i=1}^{k} \lambda_i v_i v_i^t = \sum_{i=1}^{k} B v_i v_i^t = B \left( \sum_{i=1}^{k} v_i v_i^t \right) = BVV^t = B.$$ 

If $B$ has $k$ distinct eigenvalues, i.e., each eigenvalue of $B$ has multiplicity one, then the only unit eigenvectors associated with $\lambda_i$ are $\pm v_i$ and the spectral decomposition of $B$ is unique up to sign changes in the columns of $V$.

2.4 Quadratic Forms

The symmetric $k \times k$ matrix $B$ is **positive semidefinite** iff $x^t B x \geq 0$ for every $x \in \mathbb{R}^k$. This definition extends the concept of nonnegativity from $1 \times 1$ matrices (real numbers) to $k \times k$ matrices, motivating the notation $B \geq 0$. If $x^t B x > 0$ when $x \neq 0$, then $B$ is **positive definite** and we write $B > 0$.

Suppose that $\lambda < 0$ is an eigenvalue of $B$ with unit eigenvector $v$. Then

$$v^t B v = v^t (\lambda v) = \lambda v^t v = \lambda < 0,$$

so $B$ cannot be positive semidefinite. Conversely, suppose that $B$ has nonnegative eigenvalues $\lambda_1, \ldots, \lambda_k$ and let $v_1, \ldots, v_k$ be an orthonormal basis of eigenvectors of $B$. Then

$$x^t B x = (c_1 v_1 + \cdots + c_k v_k)^t B (c_1 v_1 + \cdots + c_k v_k)$$

$$= (c_1 v_1 + \cdots + c_k v_k)^t (c_1 \lambda_1 v_1 + \cdots + c_k \lambda_k v_k)$$

$$= c_1^2 \lambda_1 + \cdots + c_k^2 \lambda_k$$

$$\geq 0,$$
so $B$ must be positive semidefinite. Thus, $B \geq 0$ iff $B$ has nonnegative eigenvalues. Analogously, $B > 0$ iff $B$ has strictly positive eigenvalues.

The symmetric $k \times k$ matrix $B = [b_{ij}]$ is an inner product matrix (or Gram matrix) iff there exists $a_1, \ldots, a_k \in \mathbb{R}^p$ such that $b_{ij} = \langle a_i, a_j \rangle = a_i^t a_j$. Writing $A = [a_1 \cdots a_k]$, $B$ is an inner product matrix iff there exists a $p \times k$ matrix $A$ such that $B = A^t A$. If $B = A^t A$, then

$$x^t B x = x^t A^t A x = (Ax)^t (Ax) = \|Ax\|_2^2 \geq 0,$$

must be positive semidefinite. Conversely, let $B = V \Lambda V^t$ be a spectral decomposition of $B$. If $B \geq 0$, then $\Lambda = \Sigma$ and $B = V \Sigma^2 V^t = (V \Sigma)(V \Sigma)^t = A^t A$.

Alternatively,

$$B = V \Sigma^2 V^t = V \Sigma V^t V \Sigma V^t = (V \Sigma V^t)^t (V \Sigma V^t) = A^t A,$$

where $A = V \Sigma V^t$ is a symmetric square root of $B$. Either way, we conclude that $B \geq 0$ iff $B$ is an inner product matrix. Furthermore, $B > 0$ iff $B = A^t A$ and the columns of $A$ are linearly independent.

A quadratic form is a function $f : \mathbb{R}^k \to \mathbb{R}$ of the form $f(q) = q^t B q$ for $B$ symmetric. If $B \geq 0$, then $f$ is a positive semidefinite form. The Rayleigh-Ritz theorem and the following generalization thereof\(^1\) relates the optimization of positive semidefinite forms to the eigenstructure of $B$.

**Theorem 2.2** Suppose that the symmetric positive semidefinite matrix $B = [b_{ij}]$ has eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and corresponding orthonormal eigenvectors $v_1, \ldots, v_n$. Then

$$\lambda_1 = \max_{q^t q = 1} q^t B q \quad \text{and} \quad v_1 = \arg \max_{q^t q = 1} q^t B q,$$

$$\lambda_2 = \max_{q^t v_1 = 0} q^t B q \quad \text{and} \quad v_2 = \arg \max_{q^t v_1 = 0} q^t B q,$$

$$\lambda_3 = \max_{q^t v_1 = 0, q^t v_2 = 0} q^t B q \quad \text{and} \quad v_3 = \arg \max_{q^t v_1 = 0, q^t v_2 = 0} q^t B q,$$

$$\vdots \quad \vdots \quad \vdots$$

If $B > 0$, then $f$ is a positive definite form and it is easily checked that

$$\langle x, y \rangle_B = x^t B y$$

defines an inner product on $\mathbb{R}^k$. Then

$$f(q) = q^t B q = \langle q, q \rangle_B = \|q\|_B^2$$

is a strictly convex function with a unique minimizer at $q = \vec{0}$ and

$$d_B(x, y) = \sqrt{f(x - y)} = \|x - y\|_B$$

defines a metric/distance function on $\mathbb{R}^k$.

\(^1\)See Section 4.2 of *Matrix Analysis*, by R. A. Horn and C. R. Johnson.
2.5 Ellipsoids and the Singular Value Decomposition

By a centered ellipsoid we mean a linear transformation of the unit ball,
\[ \mathcal{B}(1) = \{ x \in \mathbb{R}^\ell : \| x \|_2 = 1 \} . \]

The matrix \( A \in \mathbb{R}^{k \times \ell} \) defines the centered ellipsoid
\[ \mathcal{E}(A) = \{ y \in \mathbb{R}^k : y = Ax \text{ for some } x \in \mathcal{B}(1) \} . \]

An affine ellipsoid is an affine linear transformation of \( \mathcal{B}(1) \), i.e., a set of the form \( \bar{y} + \mathcal{E}(A) \).

Given \( A \), define
\[ v_1 = \text{argmax} \left\{ \| Ax \|_2 : x \in \mathcal{B}(1) \right\} , \]
i.e., \( v_1 \in \mathcal{B}(1) \) satisfies \( \| Av_1 \|_2 \geq \| Ax \| \) for every \( x \in \mathcal{B}(1) \). Let \( y_1 = Av_1 \) and \( \sigma_1 = \| y_1 \|_2 \).

If \( \sigma_1 > 0 \), then set \( u_1 = y_1 / \sigma_1 \) and define
\[ v_2 = \text{argmax} \left\{ \| Ax \|_2 : x \in \mathcal{B}(1), \langle x, v_1 \rangle = 0 \right\} . \]
Let \( y_2 = Av_2 \) and \( \sigma_2 = \| y_2 \|_2 \). If \( \sigma_2 > 0 \), then set \( u_2 = y_2 / \sigma_2 \) and define
\[ v_3 = \text{argmax} \left\{ \| Ax \|_2 : x \in \mathcal{B}(1), \langle x, v_1 \rangle = \langle x, v_2 \rangle = 0 \right\} . \]
We continue in this manner until we obtain \( \sigma_{r+1} = 0 \), so that
\[ \sigma_1 \geq \cdots \geq \sigma_r > 0 = \sigma_{r+1} . \]
We say that \( r \) is the dimension of \( \mathcal{E}(A) \). Of course, \( r \) is also the number of linearly independent columns of \( A \), i.e., \( r = \text{rank}(A) \).

Because maximizing \( \| Ax \|_2 \) is equivalent to maximizing
\[ \| Ax \|_2^2 = (Ax)^t(Ax) = x^tA^tAx , \]
it follows from Theorem 2.2 that \( v_1, \ldots, v_r \) are orthonormal eigenvectors of \( A^tA \) with corresponding eigenvalues \( \sigma_1^2, \ldots, \sigma_r^2 \), i.e.,
\[ A^tAv_j = \sigma_j^2v_j, \ j = 1, \ldots, r . \]
In matrix form,
\[ A^tAV_1 = A^tA \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}^2 = V_1\Sigma^2. \]

Now consider the ellipsoid, \( \mathcal{E}(A) \). First,
\[ \langle y_i, y_j \rangle = \langle Av_i, Av_j \rangle = v_i^tA^tAv_j = v_i^t\sigma_j^2v_j = \begin{bmatrix} \sigma_i^2 & \ i = j \\ 0 & \ i \neq j \end{bmatrix} , \]
so \( y_1, \ldots, y_r \) are mutually orthogonal and \( u_1, \ldots, u_r \) form an orthonormal basis for \( \text{col}(A) \).
The orthogonal lines through \( \pm y_i \), and also the orthogonal line segments \( [-y_i, y_i] \), are called
the axes of $\mathcal{E}(A)$. The line segments $[0, \pm y_i]$ are called the semiaxes of $\mathcal{E}(A)$ and the positive real numbers $\sigma_i$ are the lengths of the semiaxes.

Notice that the $(\sigma_i, u_i, v_i)$ are completely determined by the matrix $A$. The $\sigma_i$ are singular values of $A$, the $u_i$ are left singular vectors of $A$, and the $v_i$ are right singular vectors of $A$. The designations left and right are not intuitive, but will make sense after we organize the preceding constructions in the form of the celebrated singular value decomposition of $A$.

Let us write the equations

$$\begin{align*}
Av_1 &= y_1 = \sigma_1 u_1 \\
&\vdots \\
Av_r &= y_r = \sigma_r u_r
\end{align*}$$

in matrix form:

$$AV_1 = A \begin{bmatrix} v_1, \ldots, v_r \end{bmatrix} = \begin{bmatrix} u_1, \ldots, u_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\
& \ddots & \\
& & \sigma_r \end{bmatrix} = U_1 \Sigma.$$

Let

$$A^- = V_1 \Sigma^{-1} U_1^t,$$

so that

$$A^- \begin{bmatrix} y_1, \ldots, y_r \end{bmatrix} = A^- U_1 \Sigma = V_1 \Sigma^{-1} U_1^t U_1 \Sigma = V_1.$$

Just as $A$ maps $v_1, \ldots, v_r$ to $y_1, \ldots, y_r$, its pseudoinverse, $A^-$, maps $y_1, \ldots, y_r$ to $v_1, \ldots, v_r$.

Instead of defining ellipsoids via linear transformations, we might (more conventionally) define ellipsoids via quadratic forms. Given $A$, define the symmetric positive semidefinite matrix

$$B = \left( A^- \right)^t \left( A^- \right) = \left( V_1 \Sigma^{-1} U_1^t \right)^t \left( V_1 \Sigma^{-1} U_1^t \right) = U_1 \Sigma^{-1} V_1^t V_1 \Sigma^{-1} U_1^t = U_1 \Sigma^{-2} U_1^t.$$

We then obtain the following equivalence.

**Theorem 2.3** Suppose that $y \in \text{col}(A)$. Then $y \in \mathcal{E}(A)$ iff $y^t By \leq 1$.

**Proof** Given $y \in \text{col}(A)$, write $y = Ax$ and $x = V_1 c$. Note that $x \in B^t(1)$ iff $c^t c \leq 1$. Then

$$y^t By = x^t A^t U_1 \Sigma^{-2} U_1^t A x = c^t V_1^t A^t U_1 \Sigma^{-2} U_1^t A V_1 c = c^t \left( U_1 \Sigma \right)^t U_1 \Sigma^{-2} U_1^t \left( U_1 \Sigma \right) c = c^t \Sigma^t U_1^t U_1 \Sigma^{-2} U_1^t U_1 \Sigma c = c^t \Sigma^t \Sigma c = c^t c;$$

hence, $y^t By \leq 1$ is equivalent to $x \in B^t(1)$ and therefore to $y \in \mathcal{E}(A)$. $\square$
Thus far we have relied on the orthonormal vectors \(v_1, \ldots, v_r\). If \(r < \ell\), we now extend \(v_1, \ldots, v_r\) to an orthonormal basis for \(\mathbb{R}^\ell\), \(v_1, \ldots, v_r, v_{r+1}, \ldots, v_\ell\) and rewrite the matrix equation \(AV_1 = U_1 \Sigma\) as

\[
AV = A \left[ \begin{array}{c|c} V_1 & V_2 \end{array} \right] = A \left[ \begin{array}{c} v_1 \cdots v_r \mid v_{r+1} \cdots v_\ell \end{array} \right] = \left[ \begin{array}{c|c} U_1 \Sigma & 0 \end{array} \right] = U_1 \left[ \begin{array}{c|c} \Sigma & 0 \end{array} \right].
\]

Because \(V\) is orthogonal, we can right-multiply this equation by \(V^{-1} = V^t\) to obtain

\[
A = U_1 \left[ \begin{array}{c|c} \Sigma & 0 \end{array} \right] V^t,
\]

the reduced (or thin) singular value decomposition of \(A\). If \(r < k\), then we can also extend \(u_1, \ldots, u_r\) to an orthonormal basis for \(\mathbb{R}^k\), \(u_1, \ldots, u_r, u_{r+1}, \ldots, u_k\), then write \(U_2 = [u_{r+1} \cdots u_k]\) and

\[
A = \left[ \begin{array}{c|c} U_1 & U_2 \end{array} \right] \left[ \begin{array}{c} \Sigma \mid 0 \end{array} \right] V^t = U \left[ \begin{array}{c} \Sigma \mid 0 \end{array} \right] V^t,
\]

the singular value decomposition (SVD) of \(A\).

**Theorem 2.4 (Singular Value Decomposition)** Let \(r\) denote the rank of the \(k \times \ell\) matrix \(A\). Then \(A\) can be factored as

\[
A = U \left[ \begin{array}{c|c} \Sigma & 0 \end{array} \right] V^t,
\]

where \(U\) is a \(k \times k\) orthogonal matrix, \(V\) is a \(\ell \times \ell\) orthogonal matrix, \(\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)\), and \(\sigma_1 \geq \cdots \geq \sigma_r > 0\) are the singular values of \(A\).

The volume of the ellipsoid \(E(A)\) is proportional to the product of the lengths of its semiaxes, i.e., to the product of the strictly positive singular values of \(A\). Suppose that we flatten \(E(A)\) by retaining the \(d\) semiaxes of greatest length. The resulting ellipsoid is \(E(A_d)\), where

\[
A_d = U \left[ \begin{array}{c|c} \Sigma_d & 0 \end{array} \right] V^t \quad \text{and} \quad \Sigma_d = \left[ \begin{array}{c} \sigma_1 \quad \cdots \quad \sigma_d \end{array} \right].
\]

The following result states that \(A_d\) is the best (in the sense of squared error) rank-\(d\) approximation of \(A\). Given an \(n \times q\) matrix \(E = [e_{ij}]\), let

\[
\|E\|_F^2 = \left( \sum_{i=1}^n e_{ij}^2 \right)^2
\]

denote the Frobenius norm of \(E\).

**Theorem 2.5 (Eckart & Young, 1936)** Let \(\Sigma_d = \text{diag}(\sigma_1, \ldots, \sigma_d)\) and

\[
A_d = U \left[ \begin{array}{c|c} \Sigma_d & 0 \end{array} \right] V^t.
\]

Then \(A_d\) minimizes \(\|X - A\|_F^2\) among all \(k \times \ell\) matrices \(X\) of rank \(\leq d\) and

\[
A_d V = \left[ \begin{array}{c|c} U_d \mid \cdot \end{array} \right] \left[ \begin{array}{c} \Sigma_d \mid 0 \end{array} \right] V^t V = \left[ \begin{array}{c|c} U_d \Sigma_d \mid 0 \end{array} \right]
\]

represents \(A_d\) with respect to the \(d\) orthogonal axes of \(E(A_d)\).
Notice that the approximations provided by Theorem 2.5 are nested, e.g., the two axes that coordinatize the best 2-dimensional approximation of $A$ are the first two of the three axes that coordinatize the best 3-dimensional approximation of $A$.

## 2.6 Euclidean Distance Geometry

Given an $n \times n$ matrix $\Delta = [\delta_{ij}]$, the fundamental problem of Euclidean distance geometry is to determine whether or not there exist $p$ and $x_1, \ldots, x_n \in \mathbb{R}^p$ such that each $\delta_{ij} = d(x_i, x_j) = \|x_i - x_j\|_2$. If such a configuration of points exists, then we say that $\Delta$ is a Type 1 Euclidean distance matrix (EDM-1). The configuration itself is an embedding of $\Delta$ in $\mathbb{R}^p$, and the smallest $p$ for which embedding is possible is the embedding dimension of $\Delta$. If there exists a configuration such that each $\delta_{ij} = \|x_i - x_j\|_2^2$, then we say that $\Delta$ is a Type 2 Euclidean distance matrix (EDM-2).

Obviously, $\Delta$ is EDM-1 iff $\Delta^2 = [\delta_{ij}^2]$ is EDM-2.

Several easily checked conditions that are necessary for $\Delta$ to be EDM-1 are readily inferred from the definition of distance. If $\Delta$ is EDM-1, then $\delta_{ij} = \delta_{ji}$ ($\Delta$ is symmetric), $\delta_{ij} \geq 0$ ($\Delta$ is nonnegative), and $\delta_{ii} = 0$ ($\Delta$ is hollow). A symmetric, nonnegative, hollow matrix—a matrix that might plausibly be EDM-1 (or might naturally be approximated by a matrix that is EDM-1)—is a dissimilarity matrix. The challenge that we consider in this section is to determine whether or not a dissimilarity matrix is EDM-1. This challenge was first addressed by a young Arthur Cayley [3], in 1841, for the case $n = 5$. Cayley’s approach culminated in a general characterization provided by Karl Menger in the late 1920s and early 1930s. Menger’s conditions are not easily checked, nor are they constructive. If $\Delta$ is EDM-1, then we would like to find $x_1, \ldots, x_n \in \mathbb{R}^p$ such that $\delta_{ij} = \|x_i - x_j\|_2$. A constructive approach, pioneered in the 1930s by I. J. Schoenberg [33] and by Gale Young and A. S. Householder [43], exploits a connection between (squared) Euclidean distances and Euclidean inner products.

Given $x_1, \ldots, x_n \in \mathbb{R}^p$, let

$$X = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix}.$$ 

The $n \times p$ matrix $X$ is a data matrix; in the present context, we also call it a configuration matrix. The pairwise dot products of $x_1, \ldots, x_n$ are the entries of the $n \times n$ inner product matrix

$$B = B(X) = XX^t = [x_i^t x_j] = [b_{ij}].$$

Evidently, any $B \geq 0$ can be embedded by first computing a factorization $B = XX^t$, then setting $x_i^t$ equal to row $i$ of $X$. Although any square root of $B$ suffices to embed $B$, we emphasize the construction in Figure 2.1, which relies on the spectral decomposition of $B$. One virtue of this construction is that the number of strictly positive eigenvalues, $r = \text{rank}(B)$, is the smallest dimension in which $B$ can be embedded.

Because embedding inner products is straightforward, it is natural to ask if the problem of embedding distances can be reduced to the problem of embedding inner products. Toward that end, we define $D_2(X) = D_2 = [d_{ij}^2]$ by

$$d_{ij}^2 = \|x_i - x_j\|_2^2 = \langle x_i - x_j, x_i - x_j \rangle = \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle = b_{ii} - 2b_{ij} + b_{jj} \quad (2.2)$$

In the literature on Euclidean distance geometry, EDM conventionally means EDM-2.
1. Compute the spectral decomposition of $B \geq 0$ and write

$$B = \sum_{i=1}^{r} \lambda_i v_i v_i^t,$$

where $\lambda_1 \geq \cdots \geq \lambda_r > 0$ are the strictly positive eigenvalues of $B$ and $v_1, \ldots, v_r$ are corresponding orthonormal eigenvectors.

2. Let $\lambda_i = \sigma_i^2$ and

$$X = \begin{bmatrix} \sigma_1 v_1 & \cdots & \sigma_r v_r \end{bmatrix} = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix}.$$ 

Figure 2.1: Embedding a Euclidean inner product matrix $B$.

and note that these relations are summarized by the linear matrix equation

$$D_2 = \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} - 2B + \begin{bmatrix} b_{11} & b_{nn} \\ \vdots & \ddots & \vdots \\ b_{11} & b_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} b_{11} \\ \vdots \\ b_{nn} \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} - 2B + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{nn} \end{bmatrix}$$

$$= \text{diag}(B) e^t - 2B + e \text{diag}(B)^t$$

$$= \kappa(B).$$

Thus, (squared) Euclidean distances can be deduced from Euclidean inner products. If we can also deduce Euclidean inner products from Euclidean distances, then we can embed the latter by embedding the former.

Converting distances to inner products requires special care. Consider Figure 2.2, which displays two configurations of $n = 4$ points in $\mathbb{R}^2$,

$$X = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 1 & 2 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} 6 & 4 \\ 4 & 4 \\ 5 & 5 \\ 5 & 3 \end{bmatrix}.$$

One configuration is a translation of the other, e.g., we can obtain $Y$ from $X$ by adding $a = (4, 3)$ to each $x_i$. Translation is isometric, i.e., it does not affect the pairwise interpoint distances:

$$D_2(Y) = D_2(X) = \begin{bmatrix} 0 & 4 & 2 & 2 \\ 4 & 0 & 2 & 2 \\ 2 & 2 & 0 & 4 \\ 2 & 2 & 4 & 0 \end{bmatrix}.$$
The pairwise inner products, however, are strikingly different:

\[
B(X) = \begin{bmatrix}
5 & 1 & 4 & 2 \\
1 & 1 & 2 & 0 \\
4 & 2 & 4 & 1 \\
2 & 0 & 1 & 1
\end{bmatrix}
\quad \text{versus} \quad B(Y) = \begin{bmatrix}
52 & 40 & 50 & 42 \\
40 & 32 & 40 & 32 \\
50 & 40 & 50 & 40 \\
42 & 32 & 40 & 34
\end{bmatrix}.
\]

The dot products, e.g.,

\[
b_{ij}(X) = \langle x_i, x_j \rangle = \cos \theta_{ij} \cdot \|x_i\|_2 \cdot \|y_j\|_2,
\]

are determined by the lengths of \(x_i\) and \(x_j\), and by the angle between them. These quantities, in turn, are affected by the location of the origin in the coordinate system used to represent them. Translating the configuration is equivalent to translating the origin.

We see that a single distance matrix is associated with infinitely many inner product matrices. Converting the former to one of the latter requires us to specify additional information. The most common specification is that the centroid of the configuration lie at the origin, i.e.,

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n} \bar{X}'e = \overline{\mathbf{0}}.
\]

If this is the case, then double centering \(D_2\) leads to

\[
d_{ij}^2 - \frac{1}{n} \sum_{k=1}^{n} d_{kj}^2 - \frac{1}{n} \sum_{\ell=1}^{n} d_{i\ell}^2 + \frac{1}{n^2} \sum_{k,\ell=1}^{n} d_{k\ell}^2 = \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle
\]
CHAPTER 2. EUCLIDEAN SPACE

\[-\frac{1}{n} \sum_{k=1}^{n} [\langle x_k, x_k \rangle - 2 \langle x_k, x_j \rangle + \langle x_j, x_j \rangle] \]
\[-\frac{1}{n} \sum_{\ell=1}^{n} [\langle x_\ell, x_\ell \rangle] - 2 \langle x_\ell, x_i \rangle + \langle x_i, x_i \rangle] \]
\[+ \frac{1}{n} \sum_{k,\ell=1}^{n} [\langle x_k, x_k \rangle - 2 \langle x_k, x_\ell \rangle + \langle x_\ell, x_\ell \rangle] \]
\[= \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle \]
\[-\frac{1}{n} \sum_{k=1}^{n} \langle x_k, x_k \rangle + 2 \langle \bar{x}, x_j \rangle - \langle x_j, x_j \rangle \]
\[-\langle x_i, x_i \rangle + 2 \langle x_i, \bar{x} \rangle - \frac{1}{n} \sum_{\ell=1}^{n} \langle x_\ell, x_\ell \rangle \]
\[+ \frac{1}{n} \sum_{k=1}^{n} \langle x_k, x_k \rangle - 2 \langle \bar{x}, x \rangle + \frac{1}{n} \sum_{\ell=1}^{n} \langle x_\ell, x_\ell \rangle \]
\[= -2 \langle x_i, x_j \rangle + 2 \langle \bar{x}, x_j \rangle + 2 \langle x_i, \bar{x} \rangle - 2 \langle \bar{x}, \bar{x} \rangle \]
\[= -2 \langle x_i, x_j \rangle \]
\[= -2 b_{ij}. \]

We see that we can obtain $B$ from $D_2$ by first double centering $D_2$, then dividing by $-2$.

To obtain a matrix representation of double centering, let $P = I - ee^t/n$, where $I$ denotes the $n \times n$ identity matrix and $e = (1, \ldots, 1) \in \mathbb{R}^n$. The $n \times n$ matrix $P$ is symmetric ($P^t = P$) and idempotent ($P^2 = P$), hence a projection matrix. Left-multiplication by $P$, i.e.,

$$PA = A - \frac{ee^t}{n} A = A - e \left( \frac{e^t}{n} A \right),$$

subtracts from each entry of $A$ the average of the column in which it appears, resulting in a column-centered matrix:

$$e^t PA = e^t A - e^t e \left( \frac{e^t}{n} A \right) = e^t A - e^t A = \vec{0}.$$

Similarly, right-multiplication by $P$ results in a row-centered matrix. Double centering is accomplished by

$$PAP = \left( I - \frac{ee^t}{n} \right) A \left( I - \frac{ee^t}{n} \right)$$

$$= A - \frac{ee^t}{n} A - A \frac{ee^t}{n} + \frac{ee^t}{n} A \frac{ee^t}{n}$$

$$= A - e \left( \frac{e^t}{n} A \right) - \left( A \frac{e}{n} \right) e^t + e \left( \frac{e^t}{n} A \frac{e}{n} \right) e^t.$$

Thus, Euclidean inner products can be obtained from (squared) Euclidean distances by

$$B = -\frac{1}{2} PD_2 P = \tau(D_2)$$

and we have derived the following result.
CHAPTER 2. EUCLIDEAN SPACE

Theorem 2.6 An $n \times n$ dissimilarity matrix $A = [a_{ij}]$ is EDM-2 iff

$$B = \tau(A) = -\frac{1}{2}PAP \geq 0.$$ 

If $B = [b_{ij}] \geq 0$, then $\text{rank}(B)$ is the embedding dimension of $A$. If $x_1, \ldots, x_n \in \mathbb{R}^p$ are such that $\langle x_i, x_j \rangle = b_{ij}$, then $\|x_i - x_j\|_2^2 = a_{ij}$.

Notice that $Pe = \bar{0}$. Hence, if $B = \tau(A)$, then $Be = \bar{0}$ and therefore $e^t Be = 0$. If $B = XX^t$, then

$$0 = e^tXX^t e = (X^t e)^t (X^t e)$$

and it follows that $X^t e = \bar{0}$. Hence, double centering $D_2$ results in configurations with centroid $\bar{x} = X^t e/n = \bar{0}$. If we restrict the domain of $\kappa$ to $B$ for which $Be = \bar{0}$, then $\kappa$ and $\tau$ are mutually inverse linear maps. The properties of these maps were investigated by Critchley [5].

Theorem 2.6 allows us to prove a familiar fact, that any two points in Euclidean space lie on a line, any three points lie in a plane, etc. Given $x_1, \ldots, x_n \in \mathbb{R}^d$, let

$$A = \left[\|x_i - x_j\|^2\right] \quad \text{and} \quad B = \tau(A).$$

Then $B$ is positive semidefinite and the smallest number of dimensions needed to represent $x_1, \ldots, x_n$ is $r = \text{rank}(B) \leq n$. Furthermore, because $Be = \bar{0}$, $r \leq n - 1$.

We can easily replace the unweighted centroid

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{n} X^t e = X^t e \frac{e}{n}$$

with a weighted centroid

$$\bar{x}_w = \sum_{i=1}^n x_i x_i = X^t w,$$

for any $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$ such that $\sum_{i=1}^n w_i = w^t e = 1$. If we specify that the weighted centroid must lie at the origin, then we obtain

$$d_{ij}^2 - \sum_{k=1}^n w_k d_{kj}^2 - \sum_{\ell=1}^n w_\ell d_{i\ell}^2 + \sum_{k,\ell=1}^n w_k w_\ell d_{k\ell}^2 = \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle$$

$$- \sum_{k=1}^n w_k \langle x_k, x_k \rangle - 2 \langle x_k, x_j \rangle + \langle x_j, x_j \rangle$$

$$- \sum_{\ell=1}^n w_\ell \langle x_i, x_i \rangle - 2 \langle x_i, x_\ell \rangle + \langle x_\ell, x_\ell \rangle$$

$$+ \sum_{k,\ell=1}^n w_k w_\ell \langle x_k, x_k \rangle - 2 \langle x_k, x_\ell \rangle + \langle x_\ell, x_\ell \rangle$$

$$= \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle$$

$$- \sum_{k=1}^n w_k \langle x_k, x_k \rangle + 2 \langle x_k, x_j \rangle - \langle x_j, x_j \rangle$$

$$- \langle x_i, x_i \rangle + 2 \langle x_i, \bar{x}_w \rangle - \sum_{\ell=1}^n w_\ell \langle x_\ell, x_\ell \rangle$$
\[
+ \sum_{k=1}^{n} w_k \langle x_k, x_k \rangle - 2 \langle \bar{x}_w, \bar{x}_w \rangle + \sum_{\ell=1}^{n} w_\ell \langle x_\ell, x_\ell \rangle \\
= -2 \langle x_i, x_j \rangle + 2 \langle \bar{x}_w, x_j \rangle + 2 \langle x_i, \bar{x}_w \rangle - 2 \langle \bar{x}_w, \bar{x}_w \rangle \\
= -2 \langle x_i, x_j \rangle \\
= -2b_{ij}.
\]

In matrix form,
\[
\left( I - ew^t \right) D_2 \left( I - we^t \right) = D_2 - e \left( w^t D_2 \right) - \left( D_2 w \right) e^t + e \left( w^t D_2 w \right) e^t = -2B,
\]
or
\[
\tau_w (D_2) = -\frac{1}{2} \left( I - ew^t \right) D_2 \left( I - we^t \right) = B,
\]
and we obtain a more general result due to Gower [15, 16]. The special case of \( w = e_n \) is due to Schoenberg [33] and Young and Householder [43]. Torgerson [39] popularized the use of \( w = e/n \).

**Theorem 2.7 (Gower, 1982, 1985)** Let \( w \in \mathbb{R}^n \) satisfy \( w^t e = 1 \). An \( n \times n \) dissimilarity matrix \( A = [a_{ij}] \) is EDM-2 iff
\[
B = \tau_w (A) = -\frac{1}{2} \left( I - ew^t \right) A \left( I - we^t \right) \geq 0.
\]
If \( B = [b_{ij}] \geq 0 \), then \( \text{rank}(B) \) is the embedding dimension of \( A \). If \( x_1, \ldots, x_n \in \mathbb{R}^p \) are such that \( \langle x_i, x_j \rangle = b_{ij} \), then \( \|x_i - x_j\|_2^2 = a_{ij} \).

It is natural to inquire what effect the choice of \( w \) has on the configuration constructed by factoring \( \tau_w (D_2) \). Suppose that \( X \) is a \( v \)-centered configuration matrix, i.e., \( X^t v = 0 \), and let \( B = XX^t \) and \( b = \text{diag}(B) \). Let us construct a \( w \)-centered configuration matrix \( Y \) by factoring
\[
\tau_w (\kappa (B)) = -\frac{1}{2} \left( I - ew^t \right) \left( be^t - 2B + eb^t \right) \left( I - we^t \right) \\
= -\frac{1}{2} \left( I - ew^t \right) \left( be^t + eb^t \right) \left( I - we^t \right) + \left( I - ew^t \right) B \left( I - we^t \right).
\]
Some arithmetic reveals that
\[
\left( I - ew^t \right) \left( be^t + eb^t \right) \left( I - we^t \right) = 0,
\]
and
\[
\left( I - ew^t \right) B \left( I - we^t \right) = \left( I - ew^t \right) XX^t \left( I - we^t \right)^t = [(I - ew^t) X] [(I - ew^t) X]^t.
\]
Thus, it suffices to obtain \( Y = (I - ew^t) X \) by simply recentering \( X \).
2.7 Cosine Similarity

The tools that we have developed in preceding sections allow us to explicate a popular measure of similarity, briefly noted in Example 4.4. Imagine a corpus of \( n \) documents about dogs and/or wolves. Let \( x_{i1} \) denote the number of times the word dog or dogs appears in document \( i \) and let \( x_{i2} \) denote the number of times the word wolf or wolves appears in document \( i \). These word counts impart some information about document content. Suppose, for example, that I am searching for documents about how dogs evolved from wolves. Such documents may have comparable values of \( x_{i1} \) and \( x_{i2} \), whereas documents about training dogs to heel will tend to have \( x_{i1} \gg x_{i2} \) and documents about reintroducing wolves to Yellowstone National Park will tend to have \( x_{i2} \gg x_{i1} \).

What is important for distinguishing document content in our imaginary corpus are word counts relative to document size. Suppose, for example, that we observe the following counts:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( x_{i1} )</th>
<th>( x_{i2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

While document 3 may be much longer than documents 1 or 2, what interests us is that documents 2 and 3 both contain equal numbers of dog/dogs and wolf/wolves. For this application, it follows that Euclidean distance is not an appropriate measure of document dissimilarity:

\[
d(x_2, x_3) = \sqrt{1800} \approx 42.43 > 14.14 = \sqrt{200} = d(x_2, x_1),
\]

but we want to identify documents 2 and 3 while distinguishing both from document 1.

An obvious solution to this difficulty is to normalize the word counts by the transformation

\[
x_i \mapsto z_i = \frac{x_i}{\|x_i\|_2}.
\]

This transformation identifies each \( x \in \mathbb{R}^q \) with a corresponding point \( z \) in the unit sphere,

\[
S = \{ z \in \mathbb{R}^q : \|z\|_2 = 1 \}.
\]

Geometrically, \( x \neq \vec{0} \) determines a ray that originates from \( \vec{0} \) and passes through \( x \). The point \( z \) is the point at which the ray intersects \( S \).

The standard measure of distance on a surface is geodesic distance, the shortest distance that must be travelled on the surface to pass from one point to another. Geodesic distance on a sphere is sometimes called great circle distance and transcontinental flights typically approximate great circle routes on our roughly spherical planet. The fact that the shortest routes on a sphere are circles of the same radius means that we can deduce geodesic distance on the unit sphere by studying arcs on the unit circle.

Let \( z_i \) and \( z_j \) denote two points on the unit circle in \( \mathbb{R}^2 \). The arc distance \( \alpha_{ij} \) between \( z_i \) and \( z_j \) equals the angle (measured in radians) between \( z_i \) and \( z_j \). Equivalently, \( \alpha_{ij} \) is the angle between the rays determined by \( x_i \) and \( x_j \). For the application that we have described, these angles are far more satisfying measures of dissimilarity than are Euclidean distances.

The angles in question are defined implicitly by the equation

\[
\cos \alpha_{ij} = \frac{\langle x_i, x_j \rangle}{\|x_i\|_2 \|x_j\|_2} = \frac{\langle x_i, \frac{x_j}{\|x_j\|_2} \rangle}{\|x_i\|_2 \|x_j\|_2} = \langle z_i, z_j \rangle.
\]
In our hypothetical application (and many actual applications), each \( x_{ik} \geq 0 \) and therefore each \( \alpha_{ij} \in [0, \pi/2] \). Hence, the proposed measure of dissimilarity can be computed as
\[
\alpha_{ij} = \arccos(z_i, z_j).
\]

Curiously, this measure is rarely used in practice. Notice that \( \cos() \) is strictly decreasing on \([0, \pi/2]\); hence, we can forgo computing arccosines and use \( \gamma_{ij} = \cos \alpha_{ij} = (z_i, z_j) \) as a measure of similarity—the cosine similarity of objects \( i \) and \( j \).

But how should we convert cosine similarities \( \Gamma = [\gamma_{ij}] \) to dissimilarities \( \Delta = [\delta_{ij}] \)? Obviously the transformation that respects the motivation for cosine similarity is \( \delta_{ij} = \arccos \gamma_{ij} \), but this transformation is rarely used. Instead one often sees \( \delta_{ij} = 1 - \gamma_{ij} \), and we proceed to study this popular transformation.

Recall the trigonometric power series expansion
\[
1 - \cos \alpha = \frac{\alpha^2}{2!} - \frac{\alpha^4}{4!} + \frac{\alpha^6}{6!} - \cdots,
\]
which reveals that \( 2(1 - \cos \alpha) \sim \alpha^2 \) for small \( \alpha \). This revelation suggests that \( 1 - \cos \alpha \) is not expressed in the appropriate units for measuring distance in \( \mathbb{R}^q \) or \( S \), somewhat as though one attempted to measure the length of a road race in \( \text{km}^2 \) rather than \( \text{km} \). The obvious remedy for this difficulty is to take square roots, resulting in dissimilarities
\[
\delta_{ij} = \sqrt{2(1 - \cos \alpha_{ij})}.
\] (2.5)

In fact, (2.5) has a nice geometric interpretation. Some elementary trigonometric calculation shows that
\[
\|z_i - z_j\|^2 = (1 - \cos \alpha_{ij})^2 + \sin^2 \alpha_{ij} = 1 - 2 \cos \alpha_{ij} + \cos^2 \alpha_{ij} + \sin^2 \alpha_{ij} = 2 - 2 \cos \alpha_{ij},
\]
so \( \delta_{ij} = d(z_i, z_j) \) is the chordal distance between \( z_i, z_j \in S \).

If \( \gamma_{ij} = (z_i, z_j) \), then the transformation \( \gamma_{ij} \mapsto 2(1 - \gamma_{ij}) \) converts Euclidean inner products to squared Euclidean distances. Because \( z_i, z_j \in S \), \( \gamma_{ii} = \|z_i\|^2 = 1 \); hence, \( \text{diag}(\Gamma) = e \) and
\[
\kappa(\Gamma) = \text{diag}(\Gamma)e^t - 2\Gamma + e\text{diag}(\Gamma)^t = 2ee^t - 2\Gamma = 2[1 - \gamma_{ij}].
\]

Thus, the linear transformation \( \Delta_2 = \kappa(\Gamma) \) computes squared chordal distances.

Recalling our discussion of the simple matching coefficient in Section 1.3.3, the present discussion of cosine similarity is the second occasion on which we have concluded that the transformation \( \Gamma \mapsto 1 - \Gamma \) constructs squared Euclidean distances. Some authors, e.g., Mardia, Kent, and Bibby [27] and Seber [34], refer to \( \Gamma \mapsto \kappa(\Gamma) \) as the standard transformation from similarity to dissimilarity. We emphasize that the elements of \( \kappa(\Gamma) \) should be interpreted as squared dissimilarities. Furthermore, if the \( \gamma_{ii} \) are not identical, then \( \kappa \) may transform individual \( \gamma_{ij} \) nonmonotonically.

### 2.8 Low-Distortion Embedding

Given \( x_1, \ldots, x_n \in \mathbb{R}^q \), the embedding dimension of \( D(X) = [\|x_i - x_j\|_2] \) may be as large as \( \min(q, n - 1) \). If, as is typically the case, the embedding dimension of \( D(X) \) is larger than desired (for visualization and/or subsequent analysis), then it is natural to search for...
\( \hat{x}_1, \ldots, \hat{x}_n \in \mathbb{R}^p, \ p \ll q \), for which \( D(\hat{X}) \) approximates \( D(X) \). In this context, various approximation criteria may be considered. An especially strong criterion, which requires that each \( \| \hat{x}_i - \hat{x}_j \|_2 \) approximate the corresponding \( \| x_i - x_j \|_2 \), is stated in Definition 2.1. This criterion is the subject of the present section; weaker criteria are studied in Chapters 4 and 5.

**Definition 2.1** Given \( \epsilon \in (0,1) \), the configuration \( \hat{x}_1, \ldots, \hat{x}_n \in \mathbb{R}^q \) if and only if

\[
(1 - \epsilon) D(X) \leq D(\hat{X}) \leq (1 + \epsilon) D(X),
\]

i.e.,

\[
(1 - \epsilon) \| x_i - x_j \|_2 \leq \| \hat{x}_i - \hat{x}_j \|_2 \leq (1 + \epsilon) \| x_i - x_j \|_2
\]

for each \( ij \) pair.

For fixed \( \epsilon, n, \) and \( q \), a celebrated lemma of Johnson and Lindenstrauss [19] guarantees the existence of a dimension \( p = p(n, \epsilon) = O(\epsilon^{-2} \log n) \) for which there exists a mapping \( \phi : \mathbb{R}^q \rightarrow \mathbb{R}^p \) such that \( \phi(x_1), \ldots, \phi(x_n) \in \mathbb{R}^p \) is guaranteed to be an \( \epsilon \)-distorted embedding of \( x_1, \ldots, x_n \in \mathbb{R}^q \). This lemma is remarkable in several respects. First, a single mapping suffices for every configuration of \( n \) points in \( \mathbb{R}^q \). Second, the embedding dimension \( p \) does not depend on the nominal dimension \( q \) and increases quite slowly as \( n \) increases. This observation offers hope that permitting some distortion may allow high-dimensional Euclidean configurations to be approximately embedded in a Euclidean space of much lower dimension. Third, the proof of existence is constructive, offering hope that low-distortion embeddings can be constructed in practice.

The proof technique in [19] is probabilistic. Since its publication, a number of researchers have proposed randomized algorithms that find low-distortion embeddings with high probability by randomly generating orthonormal (or nearly orthonormal) vectors \( q_1, \ldots, q_p \in \mathbb{R}^q \), then setting

\[
\hat{X} = \begin{bmatrix} \hat{x}_1^t \\ \vdots \\ \hat{x}_n^t \end{bmatrix} = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix} \begin{bmatrix} q_1 & \cdots & q_q \end{bmatrix} = XQ. \tag{2.6}
\]

For example:

- Indyk and Motwani [18] generated each entry in \( Q \) by drawing \( z \sim \text{Normal}(0,1/q) \). The resulting \( q_1, \ldots, q_p \) are not orthonormal, but

\[
E(q_i q_i) = \sum_{k=1}^{q} E(Z_k^2) = q(1/q) = 1
\]

and

\[
E(q_i q_j) = \sum_{k=1}^{q} E(Z_1 Z_2) = 0.
\]

- Achlioptas [1] replaced normal distributions with discrete distributions, considering two \( Q = R/\sqrt{p} \). Draw the entries in \( R \) from either (i) \( \{-1,1\} \), each with probability 1/2, or (ii) \( \{-\sqrt{3},0,\sqrt{3}\} \), with probabilities \( 1/6,2/3,1/6 \). In either case, \( Eq_i q_i = 1 \) and \( Eq_i q_j = 0 \).
Different algorithms for finding low-distortion embeddings provide different guarantees, e.g.,

**Theorem 2.8** Achlioptas [1] Fix \( n \) and \( q \). Given \( \epsilon, \beta > 0 \), let

\[
\bar{p} = \frac{4 + 2\beta}{\epsilon^2 / 2 + \epsilon^3 / 3 \log n}.
\]

For integer \( p \geq \bar{p} \), let \( R = [r_{ij}] \) be a \( q \times p \) random matrix, each \( r_{ij} \) drawn independently from either (i) \( \{-1, 1\} \) with probabilities \( (1/2, 1/2) \), or (ii) \( \{-\sqrt{3}, 0, \sqrt{3}\} \) with probabilities \( (1/6, 2/3, 1/6) \). For any \( x_1, \ldots, x_n \in \mathbb{R}^q \), let

\[
X = \begin{bmatrix}
  x_1^t \\
  \vdots \\
  x_n^t
\end{bmatrix}
\text{ and } \hat{X} = XR / \sqrt{q}.
\]

Then the probability that \( \hat{X} \) is an \( \epsilon \)-distorted embedding of \( X \) is at least \( 1 - n^{-\beta} \).

**Example 2.1** Set \( \epsilon = 1/3 \), so that an \( \epsilon \)-distorted embedding will ensure that \( \|\hat{x}_i - \hat{x}_j\|_2 \geq \|\hat{x}_r - \hat{x}_s\|_2 \) if \( \|x_i - x_j\|_2 \geq 2\|x_r - x_s\|_2 \). To apply Theorem 2.8 with confidence \( 1 - n^{-\beta} = 1 - \alpha = 0.9 \), set \( \beta = -\log \alpha / \log n = \log 10 / \log n \). We then obtain the following:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \bar{p}(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^4 )</td>
<td>746.04</td>
</tr>
<tr>
<td>( 10^4 )</td>
<td>959.19</td>
</tr>
<tr>
<td>( 10^5 )</td>
<td>1172.34</td>
</tr>
<tr>
<td>( 10^6 )</td>
<td>1385.50</td>
</tr>
</tbody>
</table>

How useful are low-distortion embeddings? Example 2.1 illustrates that, even allowing distortion as high as \( \epsilon = 1/3 \), a fairly large dimension is needed to ensure that each interpoint distance of every \( n \)-point configuration can be embedded within tolerance. That \( \bar{p}(n) = O(\log n) \) is impressive, but the absolute number of dimensions is likely to be too large for most applications. For this reason, low-distortion embeddings are usually used to preprocess extremely high-dimensional data, achieving dimensions that can then be addressed by other methods. We will return to this observation in Section 3.3.2.

Example 2.1 further illustrates that imposing a condition on each interpoint distance of every \( n \)-point configuration is extremely ambitious. The JL Lemma leaves open the possibility that \( p \ll \bar{p} \) might suffice to embed most interpoint distances of most \( n \)-point configurations within tolerance. This possibility suggests an alternative strategy—try to embed as well as possible in a fixed number of dimensions—that we will explore in Chapters 4 and 5.

### 2.9 Exercises

1. Let \( \mathcal{E} \) denote the 2-dimensional ellipsoid with axes \([-y_1, y_1]\) and \([-y_2, y_2]\), where

\[
y_1 = \begin{bmatrix}
  \sqrt{2} \\
  \sqrt{2}
\end{bmatrix} \quad \text{and} \quad y_2 = \begin{bmatrix}
  -\sqrt{2}/2 \\
  \sqrt{2}/2
\end{bmatrix}.
\]

Find two different \( 2 \times 2 \) matrices, \( A_1 \) and \( A_2 \), such that \( \mathcal{E}(A_1) = \mathcal{E} = \mathcal{E}(A_2) \).
2. Let $X$ denote an $n \times q$ data matrix and let $P = I - ee^t/n$. Show that

(a) $P^t = P$
(b) $P^2 = P$
(c) The sum of each column of $PX$ is zero.

Hint: The row vector of column sums is given by $e^t(PX)$.

3. Verify equation (2.4).

4. Let

$$A = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t$$

denote the singular value decomposition of an arbitrary $m \times n$ matrix $A$. We seek an $n \times m$ matrix $B$ that satisfies the Moore-Penrose conditions for a pseudoinverse:

MP1: $ABA = A$
MP2: $BAB = B$
MP3: $(AB)^t = AB$
MP4: $(BA)^t = BA$

(a) Show that $A^\dagger = V \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^t$

satisfies the Moore-Penrose conditions.

(b) Show that $A^\dagger$ is the unique Moore-Penrose pseudoinverse of $A$.

Hint: Show that, if both $B$ and $C$ satisfy the Moore-Penrose conditions, then $B = C$.

5. Let $P = I - ee^t/n$ and let $P^\dagger$ denote the Moore-Penrose pseudoinverse of $P$. Show that $P^\dagger = P$.

6. Use Theorem 2.6 to work Exercise 1.5.4.

Hint: Given a square matrix $A$, the following R commands produce the spectral decomposition of the matrix $\tau(A)$.

```r
> n <- nrow(A)
> j <- 1/n
> P <- diag(n)-matrix(j,nrow=n,ncol=n)
> B <- -0.5 * P %*% A %*% P
> B.eig <- eigen(B,symmetric=TRUE)
```

7. Let $A$, $B$, and $C$ denote the vertices of a triangle, and let $a$, $b$, and $c$ denote the lengths of the opposing sides. Let $\theta$ denote the angle at vertex $C$. If $C = \pi/2$, then the Pythagorean Theorem states that $c^2 = a^2 + b^2$.

Use (2.2) to derive the Law of Cosines,

$$c^2 = a^2 + b^2 - 2ab \cos \theta,$$

which generalizes the Pythagorean Theorem.
8. A popular technique in text mining is *Latent Dirichlet Allocation*. From a corpus of $n$ documents, Latent Dirichlet Allocation constructs a set of $q$ topics and a probability vector $\pi_i \in \mathbb{R}^q$ that specifies the distribution of topics in document $i$. The $n \times q$ matrix

$$X = \begin{bmatrix}
\pi_1^t \\
\vdots \\
\pi_n^t
\end{bmatrix} = [x_{ik}]$$

is a data matrix, each row of which is a vector of nonnegative real numbers whose sum is one. It follows that the rows of the data matrix $Z = [\sqrt{x_{ik}}]$ are points on the unit sphere; hence, a natural measure of the dissimilarity between rows $i$ and $j$ of $X$ is

$$\delta_{ij} = \arccos(z_i, z_j),$$

where $z_i^t$ and $z_j^t$ denote rows $i$ and $j$ of $Z$. 

Chapter 3

Principal Component Analysis

We continue our study of Euclidean space by discussing principal component analysis (PCA). For our needs, PCA is best understood as a technique for identifying a coordinate system that is often useful for representing Euclidean data. In particular, this coordinate system facilitates the construction of low-dimensional representations of high-dimensional Euclidean feature vectors.

It is sometimes the case that we observe Euclidean data directly, i.e., we observe \( q \) real-valued measurements for which we believe that Euclidean distance is an appropriate measure of dissimilarity. PCA was conceived for the purpose of obtaining low-dimensional representations of such data. More generally, we will study various methods for constructing Euclidean representations of non-Euclidean data and use PCA to reduce the dimension of the constructed Euclidean representation.

3.1 Principal Components as Coordinates

Our initial approach to PCA is geometric. After specifying a dimension, \( d \), we will find the \( d \)-dimensional hyperplane that best approximates \( x_1, \ldots, x_n \in \mathbb{R}^q \).

3.1.1 Hyperplanes

We begin by collecting some elementary facts about hyperplanes.

**Lemma 3.1** If \( L = A - \bar{a} \) is a linear subspace, then \( \bar{0} \in L \) and therefore \( \bar{a} \in A \).

**Lemma 3.2** Let \( A \) be a hyperplane. If \( a, x, y \in A \) and \( t \in \mathbb{R} \), then \( a + t(x - y) \in A \).

**Proof** If \( L = A - \bar{a} \) is a linear subspace, then

\[
(a - \bar{a}) + t [(x - \bar{a}) - (y - \bar{a})] = a - \bar{a} + t(x - y) \in L,
\]

hence

\[
a - \bar{a} + t(x - y) + \bar{a} = a + t(x - y) \in A.
\]

\( \square \)
To interpret Lemma 3.2, think of \( a \) as a point in \( A \) and \( x - y \) as a direction in \( A \). Then

\[
\{ a + t(x - y) : t \in \mathbb{R} \}
\]

is a line through \( a \) that lies entirely in \( A \).

### 3.1.2 Projection into Hyperplanes

Given an affine linear subspace \( A \) and any \( x \in \mathbb{R}^q \), we try to find the point in \( A \) that is nearest \( x \).

**Definition 3.1** The projection of \( x \in \mathbb{R}^q \) into the affine linear subspace \( A \subseteq \mathbb{R}^q \) is the unique point \( \hat{x} \in \mathbb{R}^q \) for which

\[
\|x - \hat{x}\|_2^2 \leq \|x - a\|_2^2 \quad \text{for all } a \in A.
\]

The quantity \( \|x - \hat{x}\|_2^2 \) is the squared residual error of approximating \( x \) with \( \hat{x} \). If \( x \in A \), then \( \hat{x} = x \) and the squared residual error vanishes.

To obtain a useful characterization of \( \hat{x} \), choose any \( a \in A \) and consider the line

\[
\{ \hat{x} + t(a - \hat{x}) : t \in \mathbb{R} \} \subseteq A.
\]

Because this line intersects \( \hat{x} \) at \( t = 0 \), the function

\[
f(t) = \|x - \hat{x} - t(a - \hat{x})\|_2^2 = \|x - \hat{x}\|_2^2 - 2t \langle x - \hat{x}, a - \hat{x} \rangle + t^2 \|a - \hat{x}\|_2^2
\]

has a unique minimizer at \( t = 0 \). It follows that

\[0 = f'(0) = -2 \langle x - \hat{x}, a - \hat{x} \rangle,\]

hence that

\[
\langle x - \hat{x}, a - \hat{x} \rangle = 0 \quad \text{for all } a \in A. \tag{3.1}
\]

These *normal equations* characterize \( \hat{x} \), i.e., they provide a necessary and sufficient condition for a point to be the projection of \( x \) into \( A \). They state that projection into \( A \) is accomplished by “dropping a perpendicular” from \( x \) to \( A \).

In what follows, it will be helpful to denote the projection of \( x \) into \( A \) by \( \pi_A(x) \). We collect several useful facts about projection.

**Lemma 3.3** For any \( x, y \in \mathbb{R}^q \) and \( \alpha \in [0, 1] \), \( \pi_A(\alpha x + [1 - \alpha]y) = \alpha \pi_A(x) + [1 - \alpha] \pi_A(y) \).

**Proof** Choose any \( a \in A \). By Lemma 3.2,

\[
a + (1 - \alpha) [\pi_A(x) - \pi_A(y)], a + \alpha [\pi_A(y) - \pi_A(x)] \in A.
\]

Hence, applying the normal equations to \( x \) and to \( y \) separately,

\[
\langle \alpha x + (1 - \alpha)y - [\alpha \pi_A(x) + (1 - \alpha)\pi_A(y)], a - [\alpha \pi_A(x) + (1 - \alpha)\pi_A(y)] \rangle
\]

\[
= \langle \alpha x - \alpha \pi_A(x), a - \alpha \pi_A(x) - (1 - \alpha)\pi_A(y) \rangle +
\]

\[
\langle (1 - \alpha)y - (1 - \alpha)\pi_A(y), a - \alpha \pi_A(x) - (1 - \alpha)\pi_A(y) \rangle
\]

\[
= \alpha \langle x - \pi_A(x), a + (1 - \alpha)[\pi_A(x) - \pi_A(y)] \rangle - \pi_A(x)) +
\]

\[
(1 - \alpha) \langle y - \pi_A(y), a + \alpha [\pi_A(y) - \pi_A(x)] - \pi_A(y) \rangle
\]

\[
= 0.
\]
Because, $\alpha \pi_A(x) + (1 - \alpha)\pi_A(y)$ satisfies the normal equations, it must be the projection of $\alpha x + (1 - \alpha)y$ into $A$. □

Given $x_1, \ldots, x_n \in \mathbb{R}^q$, let
\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.2}
\]
denote the centroid of the $x_i$. The proof of the following corollary to Lemma 3.3 is left as an exercise.

**Corollary 3.1**
\[
\pi_A(\bar{x}) = \frac{1}{n} \sum_{i=1}^{n} \pi_A(x_i)
\]

The following result describes how translating an affine linear subspace affects projection into it.

**Lemma 3.4** Let $A + b = \{a + b \in \mathbb{R}^q : a \in A\}$. Then $\pi_{A+b}(x) = \pi_A(x-b) + b$.

**Proof** Because $\pi_{A+b}(x) \in A + b$, we can write $\pi_{A+b}(x) = a_s + b$ for some $a_s \in A$. The definition of projection means that
\[
\| (x-b) - a_s \|_2^2 = \| x - (a_s + b) \|_2^2 \leq \| x - (a + b) \|_2^2 = \| (x-b) - a \|_2^2
\]
for every $a \in A$, so $a_s = \pi_A(x-b)$. □

### 3.1.3 Best Hyperplane Contains Centroid

If we approximate $x_1, \ldots, x_n \in \mathbb{R}^q$ with points in the affine linear subspace $A$, then the total residual squared error is
\[
T(A) = \sum_{i=1}^{n} \| x_i - \pi_A(x_i) \|_2^2.
\]

We seek the affine linear subspace that minimizes this quantity. In this section we establish that the optimal $A$ must contain $\bar{x}$.

We begin by writing the total residual squared error as
\[
T(A) = \sum_{i=1}^{n} \| x_i - \bar{x} + \bar{x} - \pi_A(\bar{x}) + \pi_A(\bar{x}) - \pi_A(x_i) \|_2^2
\]
\[
= \sum_{i=1}^{n} \| x_i - \bar{x} + \pi_A(\bar{x}) - \pi_A(x_i) \|_2^2 + \sum_{i=1}^{n} \bar{x} - \pi_A(\bar{x}) \|_2^2 + 2 \sum_{i=1}^{n} \langle x_i - \bar{x} + \pi_A(\bar{x}) - \pi_A(x_i), \bar{x} - \pi_A(\bar{x}) \rangle
\]

Notice that the last term vanishes because
\[
\frac{1}{n} \sum_{i=1}^{n} \left[ x_i - \bar{x} + \pi_A(\bar{x}) - \pi_A(x_i) \right] = \bar{x} - \bar{x} + \pi_A(\bar{x}) - \frac{1}{n} \sum_{i=1}^{n} \pi_A(x_i) = 0
\]
by Corollary 3.1.

Given any hyperplane $A$, we claim that we can decrease the total residual squared error by translating $A$ so that the translated hyperplane contains $\bar{x}$. To do so, let $b = \bar{x} - \pi_A(\bar{x})$. Then $\bar{x} = \pi_A(\bar{x}) + b \in A + b$, and we claim that $T(A + b) \leq T(A)$.

First, because $\bar{x} \in A + b$,

$$\| \bar{x} - \pi_{A+b}(\bar{x}) \|^2 = \| \bar{x} - \bar{x} \|^2 = 0 \leq \| \bar{x} - \pi_A(\bar{x}) \|^2.$$  \hfill (3.3)

Second, it follows from Lemma 3.2 that $a + [\pi_A(\bar{x}) - \pi_A(x_i)] \in A$. As a result, it follows from the normal equations that

$$\langle [x_i - \bar{x} + \pi_A(\bar{x})] - \pi_A(x_i), a - \pi_A(x_i) \rangle = \langle x_i - \pi_A(x_i), a - \pi_A(x_i) \rangle - \langle \bar{x} - \pi_A(\bar{x}), a - \pi_A(x_i) \rangle = 0 - \langle \bar{x} - \pi_A(\bar{x}), [a - \pi_A(x_i) + \pi_A(\bar{x})] - \pi_A(\bar{x}) \rangle = 0,$$

hence (again by the normal equations) that

$$\pi_A(x_i - \bar{x} + \pi_A(\bar{x})) = \pi_A(x_i)$$

and finally that

$$\pi_{A+b}(\bar{x}) - \pi_{A+b}(x_i) = \pi_A(\bar{x} - b) + b - \pi_A(x_i - b) - b$$

$$= \pi_A(\bar{x} - \bar{x} + \pi_A(\bar{x})) - \pi_A(x_i - \bar{x} + \pi_A(\bar{x}))$$

$$= \pi_A(\bar{x}) - \pi_A(x_i).$$  \hfill (3.4)

Now we apply (3.3) and (3.4), obtaining

$$T(A + b) = \sum_{i=1}^{n} \| x_i - \bar{x} + \pi_{A+b}(\bar{x}) - \pi_{A+b}(x_i) \|^2 + n \| \bar{x} - \pi_{A+b}(\bar{x}) \|^2$$

$$\leq \sum_{i=1}^{n} \| x_i - \bar{x} + \pi_A(\bar{x}) - \pi_A(x_i) \|^2 + n \| \bar{x} - \pi_A(\bar{x}) \|^2$$

$$= T(A),$$

as claimed.

### 3.1.4 Centering the Data

Because the best affine linear approximation of $x_1, \ldots, x_n \in \mathbb{R}^q$ must contain $\bar{x}$, we might as well find the best linear approximation of the centered data, $\tilde{x}_i = x_i - \bar{x}$. The sample principal components of $x_1, \ldots, x_n$ will be orthogonal coordinate axes of the best linear approximation of the centered data.

The original data matrix is

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1q} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nq} \end{bmatrix} = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix}.$$
Let $e = (1, \ldots, 1)^t \in \mathbb{R}^n$, so that
\[ e^t X = \left( \sum_{i=1}^n x_{i1}, \ldots, \sum_{i=1}^n x_{iq} \right) \]
and
\[ \frac{e^t}{n} X = \bar{x}^t. \]

Then the centered data matrix is
\[
\tilde{X} = \begin{bmatrix}
    x_1^t - \bar{x}^t \\
    \vdots \\
    x_n^t - \bar{x}^t
\end{bmatrix} = X - \begin{bmatrix}
    \bar{x}^t \\
    \vdots \\
    \bar{x}^t
\end{bmatrix} = X - e\bar{x}^t = X - \frac{ee^t}{n} X = \left( I - \frac{ee^t}{n} \right) X = PX.
\]

As noted in Section 2.6, $P$ is a projection matrix and the effect of left-multiplying $X$ by $P$ is to center the columns of $X$.

Henceforth, we assume that the data have been centered and we proceed to find the linear subspace that best approximates the centered data.

### 3.1.5 Rotating the Coordinate Axes

Let $M_1, \ldots, M_q$ denote a set of coordinate axes, i.e., $x_i = (x_{i1}, \ldots, x_{iq})^t$ means that the measured value of variable $M_j$ on experimental unit $i$ is $x_{ij}$. We can form new axes by taking linear combinations of the original axes, e.g.,
\[
N_1 = a_{11}M_1 + \cdots + a_{q1}M_q, \\
N_2 = a_{12}M_1 + \cdots + a_{q2}M_q, \\
\vdots \\
N_q = a_{1q}M_1 + \cdots + a_{qq}M_q.
\]

If we store the coefficients of this linear transformation in a matrix, $A = [a_{jk}]$, then column $k$ of $A$ tells us how to weight $M_1, \ldots, M_q$ in order to obtain $N_k$.

To represent $x_i$ with respect to $M_1, \ldots, M_q$ as $y_i$ with respect to $N_1, \ldots, N_q$, we simply write
\[
y_{i1} = a_{11}x_{i1} + \cdots + a_{q1}x_{iq}, \\
\vdots \\
y_{iq} = a_{1q}x_{i1} + \cdots + a_{qq}x_{iq},
\]
or
\[ y = A^t x, \quad \text{or} \quad y^t = x^t A. \]

Typically we restrict attention to $A$ that rotate the original coordinate axes. If $A$ is orthogonal, then we know from Section 2.2 that the transformation $x \mapsto A^t x$ preserves inner products, lengths, and angles.
3.1.6 Best Linear Approximation

We seek to approximate centered \( x_1, \ldots, x_n \in \mathbb{R}^q \) with a \( d \)-dimensional linear subspace, \( L \). Let \( \hat{x}_i \) denote the projection of \( x_i \) into \( L \) and let

\[
\hat{X}(L) = \begin{bmatrix}
\hat{x}_1^t \\
\vdots \\
\hat{x}_n^t
\end{bmatrix}
\]

We seek the \( d \)-dimensional \( L \) for which the sum of the squared residual errors,

\[
\| X - \hat{X}(L) \|_F^2,
\]

is minimal.

The linear subspace \( L \subset \mathbb{R}^q \) is \( d \)-dimensional if and only if there exist orthogonal coordinate axes \( N_1, \ldots, N_q \) with respect to which we have the following representation:

\[
y_i = (\hat{y}_{i1}, \ldots, \hat{y}_{iq})^t \in L \quad \text{iff} \quad \hat{y}_{i,d+1} = \cdots = \hat{y}_{iq} = 0.
\]

Hence, the \( \hat{x}_i \) lie in a \( d \)-dimensional linear subspace iff there exists an orthogonal matrix \( A \) such that

\[
\hat{Y}_A(L) = \hat{X}(L)A = \begin{bmatrix}
\cdot & 0
\end{bmatrix},
\]

i.e., at least \( q - d \) columns of \( \hat{X}A \) vanish. This representation is possible iff the rank of \( \hat{X} \) is no greater than \( d \); hence, the problem of minimizing (3.5) is the problem of finding the best rank-\( d \) approximation of \( X \).

The problem of finding the best rank-\( d \) approximation of \( X \) is solved by applying Theorems 2.4 and 2.5 to the centered data matrix \( X \). Let

\[
X = U \begin{bmatrix}
\Sigma & 0 \\
0 & 0
\end{bmatrix} V^t
\]

denote the singular value decomposition (SVD) of \( X \) and let \( \Sigma_d = \text{diag}(\sigma_1, \ldots, \sigma_d) \). Then

\[
\hat{X}_d = U \begin{bmatrix}
\Sigma_d & 0 \\
0 & 0
\end{bmatrix} V^t
\]

is the best rank-\( d \) approximation of \( X \), and

\[
\hat{Y}_d = \hat{X}_d V = \begin{bmatrix}
U_d & \cdot
\end{bmatrix} \begin{bmatrix}
\Sigma_d & 0 \\
0 & 0
\end{bmatrix} V^t V = \begin{bmatrix}
U_d \Sigma_d & 0
\end{bmatrix}
\]

represents \( \hat{X}_d \) with respect to \( d \) orthogonal coordinate axes. These coordinate axes are the sample principal component (PC) axes.

In summary:

Let \( M_1, \ldots, M_q \) denote the coordinate axes of the centered data. The sample principal component (PC) axes are the rotated axes

\[
N_k = v_{1k} M_1 + \cdots + v_{kk} M_q.
\]

The coefficients that define the rotation are sometimes called loadings. The first \( d \) PC axes span the best linear subspace for approximating the centered data. Translating this subspace back to the centroid of the uncentered data, we obtain the best affine linear subspace for approximating the uncentered data. Figure 3.1 summarizes the calculations that construct a \( d \)-dimensional PC representation of Euclidean data.
Let $X$ be an $n \times q$ data matrix of rank $r$. Fix $d \leq r$.

1. Compute the centered data matrix,

$$\tilde{X} = \left( I - \frac{ee^t}{n} \right) X.$$

2. Compute the singular value decomposition of $\tilde{X}$.

$$\tilde{X} = \sum_{i=1}^{r} \sigma_i u_i v_i^t,$$

where $\sigma_1 \geq \cdots \geq \sigma_r > 0$ are the strictly positive singular values of $\tilde{X}$, $u_1, \ldots, u_r$ are corresponding left singular vectors, and $v_1, \ldots, v_r$ are corresponding right singular vectors.

3. The configuration matrix

$$\hat{Y}_d = \begin{bmatrix} \sigma_1 u_1 & \cdots & \sigma_d u_d \end{bmatrix} = \begin{bmatrix} y_1^t \\ \vdots \\ y_n^t \end{bmatrix}$$

represents the data matrix $X$ with respect to its first $d$ principal component axes. The entries of $\hat{Y}_d$ are sometimes called *scores*.

Figure 3.1: Constructing the $d$-dimensional principal component representation of $x_1, \ldots, x_n \in \mathbb{R}^q$.

### 3.1.7 Squared Distances

PCA has an interesting effect on squared distances, noted in [27, Section 14.4]. If

$$Q = \begin{bmatrix} q_1 & \cdots & q_d & q_{d+1} & \cdots & q_q \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$$

is an orthogonal matrix, then the linear transformation $x \mapsto x^tQ$ represents $x$ with respect to the orthonormal basis $q_1, \ldots, q_q$. Setting the last $q-d$ coordinates equal to zero, it follows that the linear transformation $x \mapsto x^tQ_1$ projects $x$ into $\text{col}(Q_1)$, the column space of $Q_1$.

Let $X$ denote a centered $n \times q$ data matrix with singular value decomposition

$$X = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix} = U \Sigma V^t$$

and write

$$V = \begin{bmatrix} v_1 & \cdots & v_d & v_{d+1} & \cdots & v_q \end{bmatrix} = \begin{bmatrix} V_1 & V_2 \end{bmatrix}.$$
Then the linear transformation \( X \mapsto XQ \) projects each \( x_i \) into \( \text{col}(Q_1) \) and choosing \( Q_1 = V_1 \) results in
\[
XV_1 = \left[ U_d \right] \cdot \left[ \begin{array}{c|c}
\Sigma_d & 0 \\
\hline 
0 & V_1^t V_2 \\
\end{array} \right] V_1 = \left[ U_d \right] \cdot \left[ \begin{array}{c|c}
\Sigma_d & 0 \\
\hline 
0 & I \\
\end{array} \right] = U_d \Sigma_d,
\]
the PC representation of \( x_1, \ldots, x_n \).

Next write
\[
XQ = \left[ Xq_1 \quad \cdots \quad Xq_d \quad Xq_{d+1} \quad \cdots \quad Xq_q \right] = \begin{bmatrix} y_1^t & z_1^t \\
\vdots & \vdots \\
y_n^t & z_n^t \end{bmatrix}.
\]

Recall that the trace of a square matrix is the sum of its diagonal entries and that \( \text{trace}(AB) = \text{trace}(BA) \). Then
\[
d^2(x_i, x_j) = d^2_{ij}(X) = d^2_{ij}(XQ) = d^2(y_i, y_j) + d^2(z_i, z_j) \geq d^2(y_i, y_j) = d^2_{ij}(XQ_1),
\]
i.e., projecting \( x_i \) and \( x_j \) into \( \text{col}(Q_1) \) can only decrease the (squared) distance between them, and twice the sum of the squared interpoint distances in \( X \) is
\[
\sum_{i,j=1}^n d^2_{ij}(X) = e^t D_2(X) e = e^t \kappa \left( XX^t \right) e
\]
\[
= e^t \text{diag} \left( XX^t \right) e - 2e^t XX^t e + e^e \text{diag} \left( XX^t \right)^t e
\]
\[
= 2n \text{trace} \left( XX^t \right).
\]

We inquire what choice of \( Q_1 \) preserves the greatest part of this total, i.e., what choice of \( Q_1 \) maximizes
\[
\sum_{i,j=1}^n d^2_{ij}(XQ_1) = 2n \text{trace} \left( XQ_1 Q_1^t X^t \right)
\]
\[
= 2n \text{trace} \left( U^t \Sigma V^t Q_1 Q_1^t V \Sigma' U \right)
\]
\[
= 2n \text{trace} \left( \Sigma V^t Q_1 Q_1^t V \Sigma' \right)
\]
\[
= 2n \text{trace} \left( \left[ Q_1^t V \right] \left[ \Sigma' \Sigma \right] \left[ Q_1^t V \right]^t \right).
\]

Notice that
\[
\left[ Q_1^t V \right] \left[ Q_1^t V \right]^t = Q_1^t V V^t Q_1 = Q_1^t Q_1 = I,
\]
i.e., the rows of \( Q_1^t V \) are orthonormal, and that
\[
\Sigma' \Sigma = \begin{bmatrix}
\sigma_1^2 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \sigma_r^2 \\
\end{bmatrix}.
\]

Let
\[
A = \begin{bmatrix}
a_{11} & \cdots & a_{1q} \\
\vdots & \ddots & \vdots \\
a_{d1} & \cdots & a_{dq} \\
\end{bmatrix}.
\]
be any $d \times q$ matrix with orthonormal rows. Then

$$f(A) = \text{trace} \left( A \Sigma' \Sigma A^t \right) = \sum_{j=1}^{r} \left( a_{1j}^2 + \cdots + a_{dj}^2 \right) \sigma_j^2 = \sum_{j=1}^{n} c_j \sigma_j^2,$$

where, by orthogonality, $c_1 + \cdots + c_r = \| A \|_F^2 = d$ and each $c_j \leq 1$. Because $\sigma_1^2 \geq \cdots \geq \sigma_r^2$, we maximize $f(A)$ by choosing $A$ so that $c_1 = \cdots = c_d = 1$. If $Q_1 = V_1$, then

$$A = V_1^t V = V_1^t \left[ V_1 \mid V_2 \right],$$

$$a_{11}^2 = \cdots = a_{dd}^2, \quad \text{and}$$

$$c_1 = \cdots = c_d = 1,$$

as desired. We thus obtain Theorem 14.4.1 in [27].

**Theorem 3.1** Among all projections of $x_1, \ldots, x_n \in \mathbb{R}^d$ into a $d$-dimensional hyperplane, the principal component projection maximizes the sum of the squared interpoint distances.

For future reference, we emphasize that Theorem 3.1 does not state that $D_2(XQ_1)$ best approximates $D_2(X)$ among all $d$-dimensional representations of $x_1, \ldots, x_n$.

### 3.2 Principal Components as Variables

We now describe PCA from an algebraic perspective. Instead of regarding $M_1, \ldots, M_q$ as coordinate axes, we regard $M_j$ as a variable and $x_{1j}, \ldots, x_{nj}$ as observed values of $M_j$.

#### 3.2.1 Covariance

From univariate statistics, recall that the sample mean of $M_j$ is

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

and that the sample variance of $M_j$ is

$$s_{jj} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2.$$

Note that the sample variance of $tM_j$ is $t^2 s_{jj}$.

Our notation for sample variance may seem curious, but it is well-suited to the needs of multivariate statistics.

**Definition 3.2** The sample covariance of $M_j$ and $M_k$ is

$$s_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j) (x_{ik} - \bar{x}_k)$$

and the sample correlation of $M_j$ and $M_k$ is

$$r_{jk} = \frac{s_{jk}}{\sqrt{s_{jj} s_{kk}}}.$$
CHAPTER 3. PRINCIPAL COMPONENT ANALYSIS

Let

$$\tilde{X} = PX = \left( I - \frac{ee^t}{n} \right) X,$$

the centered data matrix. Then it is easily verified that the sample covariance matrix of $M_1, \ldots, M_q$ is

$$S = [s_{jk}] = \frac{1}{n-1} \tilde{X}^t \tilde{X},$$

which summarizes information about the sample variation of all linear combinations of $M_1, \ldots, M_q$. Let

$$a = (a_1, \ldots, a_q)^t \quad \text{and} \quad b = (b_1, \ldots, b_q)^t.$$

Let

$$N_1 = a_1 M_1 + \cdots + a_q M_q \quad \text{and} \quad N_2 = b_1 M_1 + \cdots + b_q M_q.$$ 

Then straightforward calculations reveal that the sample mean of $N_1$ is $a^t \bar{x}$ and that the sample covariance of $N_1$ and $N_2$ is $a^t S b$. One consequence of the latter is that the sample variance of $N_1$ is $a^t S a$.

3.2.2 Linear Combinations

Again we begin by centering the data. Let $\tilde{M}_1, \ldots, \tilde{M}_q$ denote the centered variables and let $\tilde{X}$ denote the centered data matrix. Writing the singular value decomposition of $\tilde{X}$ as

$$\tilde{X} = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t,$$

we discover that

$$S = \frac{1}{n-1} \tilde{X}^t \tilde{X} = \frac{1}{n-1} V \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}^t U^t \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t = V \begin{bmatrix} \frac{1}{n-1} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} V^t.$$

This factorization is the spectral decomposition of $S$. The eigenvalues of $S$ are

$$\frac{\sigma_1^2}{n-1} \geq \cdots \geq \frac{\sigma_r^2}{n-1} > 0 = \cdots = 0,$$

and the corresponding eigenvectors are the columns of $V$. Indeed, it is easily verified that, if $v_k$ is a column of $V$, then

$$S v_k = \left( \frac{\sigma_k^2}{n-1} \right) v_k.$$

Recalling the role of $V$ in Section 3.1.6, we have deduced that the eigenvectors of $S$ specify the linear combinations of $\tilde{M}_1, \ldots, \tilde{M}_q$ that define the PC variables,

$$\tilde{N}_k = v_{1k} \tilde{M}_1 + \cdots + v_{qk} \tilde{M}_q.$$ 

The sample covariance matrix of $\tilde{N}_1, \ldots, \tilde{N}_q$ is

$$\frac{1}{n-1} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}^t \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} = \frac{1}{n-1} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}^t U^t \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{n-1} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix}.$$
thus, the PC variables are uncorrelated (in the sense that their sample correlations vanish) and their sample variances are the eigenvalues of the sample covariance matrix.

Now we apply Theorem 2.2 to the sample covariance matrix. Restricting attention to orthonormal vectors of coefficients, the first eigenvector of the sample covariance matrix defines the linear combination of maximal variance, the second eigenvector defines the linear combination of next greatest variance, etc. Thus, a popular summary of PCA is the following:

The first sample principal component is the linear combination of the original centered variables with largest sample variance. Among all linear combinations not correlated with the first sample principal component, the one with largest sample variance is the second sample principal component. Et cetera, et cetera, et cetera.

3.2.3 Concentration Ellipsoids

For simplicity, suppose that $S$ is invertible. This will typically be the case when $n \gg q$.

Let $\lambda_1 \geq \cdots \geq \lambda_q > 0$ denote the eigenvalues of $S$ and let $v_1, \ldots, v_q$ denote the corresponding eigenvectors. Then $S$ is positive definite and it follows from Theorem 2.3 that the set

$$E = \{ x \in \mathbb{R}^q : x^t S^{-1} x \leq 1 \}$$

has the following properties:

1. $E$ is an ellipsoid, centered at the origin.
2. The axes of $E$ are coincident with the lines $\{ \alpha v_j : \alpha \in \mathbb{R} \}$, which intersect the boundary of $E$ at the points $\pm \sqrt{\lambda_j} v_j$.

Recall that $v_j$ defines PC coordinate axis $j$ and that $\sqrt{\lambda_j}$ is the sample standard deviation of the data along this axis. Thus, $E$ conveys information about the dispersion of the data. The translated ellipsoid, $E + \bar{x}$, is centered at the centroid of the original data and provides an ellipsoidal summary of it. This translated ellipsoid is sometimes called the sample concentration ellipsoid. Thus, an elegant summary of PCA: the sample principal components of the original data are the axes of the sample concentration ellipsoid.

3.2.4 Total Variation

The total variation of $X$ is the sum of the sample variances of $M_1, \ldots, M_q$, i.e.,

$$\sum_{j=1}^q s_{jj} = \text{trace}(S) = \text{trace} \left( V \left[ \begin{array}{cc} \frac{1}{n-1} \Sigma^2 & 0 \\ 0 & 0 \end{array} \right] V^t \right) = \text{trace} \left( \left[ \begin{array}{cc} \frac{1}{n-1} \Sigma^2 & 0 \\ 0 & 0 \end{array} \right] \right)$$

$$= \sum_{j=1}^r \frac{\sigma_j^2}{n-1} = \sum_{j=1}^r \lambda_j,$$

where $\lambda_j$ is the sample variance of PC variable $j$. The conservation of total variance is not surprising, as the variation in $x_1, \ldots, x_n$ should not depend on the choice of axes used to coordinatize $\mathbb{R}^q$. Decisions about how many principal components will be used to approximate the original data are often informed by examining $\lambda_1, \ldots, \lambda_r$, and/or

$$R^2(d) = \frac{\sum_{j=1}^d \lambda_j}{\sum_{j=1}^r \lambda_j} = \frac{\sum_{j=1}^d \sigma_j^2}{\sum_{j=1}^r \sigma_j^2}.$$
the proportion of variation “explained” by the first \( d \) principal components. A scree plot plots the pairs \((j, \lambda_j)\).\(^1\)

### 3.2.5 Scale Dependence

The shape and orientation of the concentration ellipsoid is determined by the covariance structure of \( M_1, \ldots, M_q \). Because covariance depends on scale, the results of PCA depend on the units in which the variables were measured.

**Example 3.1** Suppose that \((M_1, M_2)\) have sample covariance matrix

\[
S = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}.
\]

Then the major axis of the sample concentration ellipsoid coincides with \( M_1 \), the minor axis coincides with \( M_2 \), and the \( M_1 \) axis is twice as long as the \( M_2 \) axis.

Now suppose that we multiply \( M_2 \) by 10, as when converting from centimeters to millimeters. Then \((M_1, 10M_2)\) have sample covariance matrix

\[
S = \begin{bmatrix} 4 & 0 \\ 0 & 100 \end{bmatrix},
\]

the major axis of the sample concentration ellipsoid coincides with \( M_2 \), the minor axis coincides with \( M_1 \), and the \( M_2 \) axis is five times as long as the \( M_1 \) axis. The change of scale has a profound effect on the principal components!

The lesson of Example 3.1 is that PCA is best-suited for situations in which \( M_1, \ldots, M_q \) are commensurate, i.e., there is a natural way to scale the variables with respect to each other. If this is not the case, then several different attitudes may be encountered:

1. Don’t use PCA unless the variables are commensurate. This attitude is ideologically pure, but greatly limits the practical utility of PCA.

2. Think carefully about how to transform the variables before performing PCA. We prefer this attitude, but recognize that adopting it may prove difficult in practice.

3. Convert the variables to standard units before performing PCA. This popular prescription, available in most statistical software packages, is equivalent to computing eigenvalues and eigenvectors of the sample correlation matrix instead of the sample covariance matrix.\(^2\) Be careful! Converting to standard units may reveal structure, but it may also destroy structure. In Example 3.1, converting to standard units results in a circular sample concentration ellipsoid for which any pair of orthogonal coordinates may be taken as the principal components.

\(^1\)The word *scree* is derived from an Old Norse word for landslide. It refers to broken rock fragments at the base of a solid rock cliff or shoulder. The idea is that one should retain the principal components that form the solid rock and discard the principal components that form the fragments. Unfortunately, scree plots frequently fail to resemble the geological formations that inspired their name.

\(^2\)In R, for example, the *princomp* function uses the sample correlation matrix by default. One must specify *cor=FALSE* to perform PCA using the sample covariance matrix.
CHAPTER 3. PRINCIPAL COMPONENT ANALYSIS

3.2.6 Biplots

Next we consider how to represent information about the original variables with respect to the principal components. Toward that end, we write the sample covariance matrix as

\[ S = V \begin{bmatrix} \frac{1}{n-1} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} V^t = V \Lambda V^t = \left( V \Lambda^{1/2} \right) \left( V \Lambda^{1/2} \right)^t = HH^t \]

and let \( h_1^t, \ldots, h_q^t \) denote the rows of \( H \). We claim that \( h_1, \ldots, h_q \in \mathbb{R}^q \) convey information about the original variables, \( M_1, \ldots, M_q \).

Because \( HH^t = S \),

1. \( \|h_j\|^2 = h_j^t h_j = s_{jj} \), the sample variance of \( M_j \); and
2. \( h_j^t h_k = s_{jk} \), the sample covariance between \( M_j \) and \( M_k \).

Thus, the length of the vector \( h_j \) is the sample standard deviation of \( M_j \) and the cosine of the angle between the vectors \( h_j \) and \( h_k \) is the sample product-moment correlation between \( M_j \) and \( M_k \). Furthermore,

\[ \|h_j - h_k\|^2 = h_j^t h_j + h_k^t h_k - 2h_j^t h_k = s_{jj} + s_{kk} - 2s_{jk}, \]

the sample variance of \( M_j - M_k \). We conclude that \( h_1, \ldots, h_q \) provide a geometric representation of \( M_1, \ldots, M_q \). The coordinate axes of this representation are the principal components.

The only difficulty with the preceding representation is that it requires \( q \) dimensions for \( q \) variables. To approximate this representation in two dimensions, we approximate \( S = HH^t \) with \( S_2 = H_2 H_2^t \), the best approximation of rank \( d = 2 \). Let \( v_1, \ldots, v_q \) denote the columns of \( V \), i.e., the eigenvectors of \( S \). Then

\[ H = \begin{bmatrix} \sqrt{\lambda_1} v_1 | \cdots | \sqrt{\lambda_q} v_q \end{bmatrix}, \]
\[ H_2 = \begin{bmatrix} \sqrt{\lambda_1} v_1 | \sqrt{\lambda_2} v_2 \end{bmatrix}, \]

and the 2-dimensional representation of \( M_1, \ldots, M_q \) by the \( q \) rows of \( H_2 \) approximates the \( q \)-dimensional representation of \( M_1, \ldots, M_q \) by the \( q \) rows of \( H \). This representation is called an \( h \)-plot. If we superimpose an \( h \)-plot on a scatter diagram of the data represented with respect to the first two principal components, then we obtain a biplot.

3.3 Application to Text Mining

The phrase \textit{text mining} describes the extraction of quantitative information from a corpus (collection) of documents. The concept of proximity is intrinsic to most text mining activities. For example, what is a \textit{cluster} of documents? What does it mean for subsets of a corpus to exhibit internal cohesion and external isolation? In theory, document proximities might be obtained by direct comparison of actual documents; more commonly, attributes of each document are quantified, then proximities are computed from a mediating vector space model (VSM).

Suppose that a corpus comprises \( n \) documents, and that \( q \) terms are of interest. To compare documents \( i \) and \( j \), we might construct a \( 2 \times 2 \) contingency table that crosstabs the terms:
An obvious measure of document similarity is the simple matching coefficient that we encountered in Example 1.2, \( \gamma_{ij} = (a + d)/(a + b + c + d) \). In text mining, however, it is often the case that \( d \) will be very large, i.e., many terms that appear somewhere in the entire corpus may not appear in either of the two documents under comparison. In consequence, most pairs of documents will appear to have very small similarity. One way to circumvent this difficulty is to divide the number of affirmative matches by the number of terms that appear in at least one of the two documents. Doing so results in Jaccard’s matching coefficient: \( \gamma_{ij} = a/(a + b + c) \). It can be shown that, for either of the above matching coefficients, the similarity matrix \( \Gamma = [\gamma_{ij}] \) is positive semidefinite. Hence, an exact Euclidean representation of document similarity can be constructed by the method described in Figure 2.1. This approach is rarely used, however, because it takes no account of how frequently each affirmative match occurs.

The preceding approach might be described as a binary VSM, in contrast to more popular quantitative approaches. To construct a quantitative VSM, first let \( o_{ik} \) denote the presence (\( o_{ik} = 1 \)) or absence (\( o_{ik} = 0 \)) of term \( k \) in document \( i \) and let \( m_{ik} \) denote the number of occurrences of term \( k \) in document \( i \). To construct an \( n \times q \) data matrix \( Y = [y_{ik}] \), one might then quantify the importance of term \( k \) in document \( i \) in various ways, e.g.,

\[
\begin{align*}
y_{ik} &= m_{ik}, \\
y_{ik} &= (1 + m_{ik}) \log_2 \left( \frac{n}{o_{ik}} \right), \\
y_{ik} &= \log \left( \frac{m_{ik}}{m_{i+}} \frac{m_{i+}}{m_{++}} \right).
\end{align*}
\]

Two natural measures of dissimilarity between documents \( i \) and \( j \) are weighted Minkowski (\( L^p \)) distance, defined by

\[
\delta^p_{ij} = \sum_{k=1}^{q} w_k |y_{ik} - y_{jk}|^p,
\]

and a measure proposed by Lance and Williams [24],

\[
\delta_{ij} = \sum_{k=1}^{q} \frac{|y_{ik} - y_{jk}|}{|y_{ik} + y_{jk}|}.
\]

Suppose that the VSM represents documents as \( y_1, \ldots, y_n \in \mathbb{R}^q \) and that we desire a representation \( x_1, \ldots, x_n \in \mathbb{R}^d \). If \( \delta_{ij} = \|y_i - y_j\| \) is a suitable measure of dissimilarity, then a natural approach is to project the \( y_i \) into a \( d \)-dimensional hyperplane. PCA finds the \( d \)-dimensional hyperplane that minimizes the total squared residual error.

### 3.3.1 Latent Semantic Indexing

The hugely successful text mining technique of *Latent Semantic Indexing* (LSI) was proposed by Deerwater and collaborators in 1988 and patented the following year.\(^3\) The essential idea

\(^3\)The practice of patenting mathematical algorithms is extremely controversial.
of LSI is nothing more than constructing a \( d \)-dimensional representation from the singular value decomposition of \( Y \). Thus, the essential distinction between LSI and PCA is that LSI factors the data matrix, whereas PCA factors the centered data matrix. Whereas PCA finds the best affine linear approximation of \( y_1, \ldots, y_n \in \mathbb{R}^q \), LSI finds the best linear approximation. A caricature of the distinction is displayed in Figure 3.2.

![Figure 3.2: Latent semantic indexing finds the best linear subspace; principal component analysis finds the best affine linear subspace.](image)

### 3.3.2 Random Indexing

Exploiting pioneering work by Johnson and Lindenstrauss [19] (see Section 2.8), Papadimitriou and collaborators [29] proposed the following two-step approach to the problem of constructing a \( d \)-dimensional Euclidean representation of \( y_1, \ldots, y_n \in \mathbb{R}^q \) when \( q \) is large.

1. Apply a random projection to \( y_1, \ldots, y_n \in \mathbb{R}^q \), resulting in \( \hat{y}_1, \ldots, \hat{y}_n \in \mathbb{R}^p \), where \( p \) is chosen so that \( d < p \ll q \).

2. Apply LSI (we prefer PCA) to \( \hat{y}_1, \ldots, \hat{y}_n \in \mathbb{R}^p \), resulting in \( z_1, \ldots, z_n \in \mathbb{R}^d \).

The trick is to generate a random projection is such a way that, with high probability, it produces a low-distortion embedding of \( y_1, \ldots, y_n \) in \( \mathbb{R}^p \).

**Example 3.2** Let \( X \) denote the \( n \times 2 \) data matrix that contains the coordinates of the \( n = 201 \) points displayed in Figure 1.1(a). Let \( q = 30000 \) and suppose that \( q_1 \) and \( q_2 \) are any two orthonormal vectors in \( \mathbb{R}^q \). Then

\[
Y = X \begin{bmatrix} q_1^t \\ q_2^t \end{bmatrix}
\]

is an \( n \times q \) data matrix that represents an isometric configuration, but which does so in such a way that the 2-dimensional, spiral nature of the configuration is obscured. One can recover
the underlying structure by performing PCA, noting that only two singular values are strictly positive, and representing the data with respect to the first two principal component axes. This approach necessitates performing a singular value decomposition of an \( n \times q \) matrix. Alternatively, one might randomly generate a \( q \times 10 \) matrix \( Q = [q_{ij}] \) by drawing each \( q_{ij} \) from \( \text{Normal}(0, 1/q) \), then perform PCA on the (approximately) projected data matrix \( YQ \). This approach requires an additional matrix multiplication, but the subsequent singular value decomposition is performed on an \( n \times 10 \) data matrix. See Exercises 3.4.5 and 3.4.6 for details.

### 3.4 Adding Points

For \( x_1, \ldots, x_n \in \mathbb{R}^q \), let \( \tilde{x}_i = x_i - \bar{x} \), let \( \tilde{X} \) denote the corresponding \( n \times q \) centered data matrix, and let

\[
\tilde{X} = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t
\]

denote the singular value decomposition of \( \tilde{X} \). The symmetric \( n \times n \) matrix of centered pairwise inner products is then

\[
B = \tilde{X} \tilde{X}^t = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} U^t = U \begin{bmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} U^t,
\]

and the \( d \)-dimensional PCA representation of \( x_i \) is row \( i \) of the \( n \times d \) data matrix \( U_d \Sigma_d \), where \( \Sigma_d \) is the \( d \times d \) diagonal matrix that contains the \( d \) largest singular values of \( \tilde{X} \) and \( U_d \) is the \( n \times d \) matrix that contains the corresponding left singular vectors.

Assuming that \( \sigma_d > 0 \), the \( n \times q \) data matrix

\[
\hat{Y}_d = \begin{bmatrix} U_d \Sigma_d & 0 \end{bmatrix} = \begin{bmatrix} U_d \mid \cdot \end{bmatrix} \begin{bmatrix} \Sigma_d & 0 \\ 0 & 0 \end{bmatrix} \\
= \begin{bmatrix} U_d \mid \cdot \end{bmatrix} \begin{bmatrix} \Sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix} I \begin{bmatrix} \Sigma_d^{-1} & 0 \\ 0 & 0 \end{bmatrix} \\
= U \begin{bmatrix} \Sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix} U^t \begin{bmatrix} \Sigma_d^{-1} & 0 \\ 0 & 0 \end{bmatrix} \\
= B \begin{bmatrix} U_d \mid \cdot \end{bmatrix} \begin{bmatrix} \Sigma_d^{-1} & 0 \\ 0 & 0 \end{bmatrix} = [BU_d \Sigma_d^{-1} \mid 0].
\]

Letting \( b_i \) denote column \( i \) of the symmetric matrix \( B \), the \( d \)-dimensional PCA representation of \( x_i \) can then be written as

\[
y_i = b_i^t U_d \Sigma_d^{-1} \quad \text{or} \quad y_i = \Sigma_d^{-1} U_d^t b_i.
\]

Now suppose that we wish to insert another \( x \in \mathbb{R}^q \) into the \( d \)-dimensional PCA representation of \( x_1, \ldots, x_n \), a procedure sometimes called *out-of-sample embedding*. Let \( \mathcal{L} \) denote the linear subspace of \( \mathbb{R}^q \) spanned by \( u_1, \ldots, u_d \). Each \( y_i \) is computed by (1) projecting \( \tilde{y}_i \) into \( \mathcal{L} \), then (2) rotating \( \mathcal{L} \) to a representation in which the first \( d \) coordinates correspond to \( u_1, \ldots, u_d \) and the remaining \( n - d \) coordinates vanish, and finally (3) discarding the superfluous coordinates. We seek to do likewise for \( x \), obtaining \( y \in \mathbb{R}^d \).
First write $\tilde{x} = x - \bar{x}$ as
\[ \tilde{x} = \sum_{i=1}^{n} \alpha_i \tilde{x}_i + w. \]
where $w$ is orthogonal to the span of $\tilde{x}_1, \ldots, \tilde{x}_n$ and therefore to $L$. It follows from Exercise 3.5.1(b) that
\[ \pi_L(\tilde{x}) = \sum_{i=1}^{n} \alpha_i \pi_L(\tilde{x}_i) + \pi_L(w) = \sum_{i=1}^{n} \alpha_i \pi_L(\tilde{x}_i). \]
Rotating $L$ to coordinate axes $u_1, \ldots, u_d$ will not affect the representation of $\pi_L(\tilde{x})$ as a linear combination of the $\pi_L(\tilde{x}_i)$, nor will discarding superfluous coordinates; hence,
\[ y = \sum_{i=1}^{n} \alpha_i y_i = \sum_{i=1}^{n} \alpha_i \Sigma_{d}^{-1} U_d^t b_i = \Sigma_{d}^{-1} U_d^t \sum_{i=1}^{n} \alpha_i b_i \]
\[ = \Sigma_{d}^{-1} U_d^t \sum_{i=1}^{n} \alpha_i \begin{bmatrix} \langle \tilde{x}_1, \tilde{x}_i \rangle \\ \vdots \\ \langle \tilde{x}_n, \tilde{x}_i \rangle \end{bmatrix} = \Sigma_{d}^{-1} U_d^t \begin{bmatrix} \langle \tilde{x}_1, \sum_{i=1}^{n} \alpha_i \tilde{x}_i \rangle \\ \vdots \\ \langle \tilde{x}_n, \sum_{i=1}^{n} \alpha_i \tilde{x}_i \rangle \end{bmatrix} \]
\[ = \Sigma_{d}^{-1} U_d^t \begin{bmatrix} \langle \tilde{x}_1, \sum_{i=1}^{n} \alpha_i \tilde{x}_i + w \rangle \\ \vdots \\ \langle \tilde{x}_n, \sum_{i=1}^{n} \alpha_i \tilde{x}_i + w \rangle \end{bmatrix} = \Sigma_{d}^{-1} U_d^t \begin{bmatrix} \langle \tilde{x}_1, \tilde{x} \rangle \\ \vdots \\ \langle \tilde{x}_n, \tilde{x} \rangle \end{bmatrix} = \Sigma_{d}^{-1} U_d^t b. \quad (3.6) \]

We have derived an out-of-sample embedding formula for PCA that relies on inner products. We will return to this formula in Section 4.1. See Gower [14] for an equivalent exposition in terms of squared distances.

### 3.5 Exercises

1. Let $u \in \mathbb{R}^q$ have unit length and consider the 1-dimensional linear subspace
\[ L = \{ tu : t \in \mathbb{R} \}. \]
Show that the projection of $x \in \mathbb{R}^q$ into $L$ is given by $\hat{x} = (u^t x) u$.

2. Establish the following facts about projection.
   (a) Prove Corollary 3.1.
   (b) Let $x = \sum_{i=1}^{n} \alpha_i x_i$ be any linear combination of $x_1, \ldots, x_n \in \mathbb{R}^q$. Let $L$ be any linear subspace of $\mathbb{R}^q$. Show that
\[ \pi_L(x) = \sum_{i=1}^{n} \alpha_i \pi_L(x_i). \]

3. Consider a collection of skulls. Let $M_1$ denote the length (in mm) of a skull and let $M_2$ denote the breadth. Define two new variables, a size variable $N_1 = M_1 + M_2$ and a shape variable $N_2 = M_1 - M_2$.
   (a) Determine $A$, the matrix of coefficients that transforms $(M_1, M_2)$ to $(N_1, N_2)$.
   (b) Is $A$ orthogonal?
(c) Let \( d_{ij} \) denote the Euclidean distance between skulls \( i \) and \( j \) using \( M_1 \) and \( M_2 \) as coordinate axes. Let \( e_{ij} \) denote the Euclidean distance between skulls \( i \) and \( j \) using \( N_1 \) and \( N_2 \) as coordinate axes. Compute \( t = e_{ij}/d_{ij} \).

(d) Let \( B = A/t \). Show that \( B \) is orthogonal.

4. Given variables \( M_1, \ldots, M_q \), let \( s_{jk} \) denote the sample covariance of \( M_j \) and \( M_k \) and set \( S = [s_{jk}] \). Verify that
\[
S = \frac{1}{n-1} \tilde{X}^t \tilde{X},
\]
where \( \tilde{X} \) is the centered data matrix.

5. Suppose that \( M_1, \ldots, M_q \) have sample mean vector \( \bar{x} \) and sample covariance matrix \( S \). Let \( a = (a_1, \ldots, a_q)^t \) and let \( N = a_1 M_1 + \cdots + a_q M_q \). Show that

(a) The sample mean of \( N \) is \( a^t \bar{x} \).
(b) The sample variance of \( N \) is \( a^t S a \).

6. Let \( q = 30000 \). The following R commands generate a random \( q \times 10 \) matrix:
\[
> q <- 30000
> Q <- matrix(rnorm(q*10),ncol=10)/sqrt(q)
\]
Each column of \( Q \) is a vector in \( \mathbb{R}^q \). Investigate how nearly orthonormal these vectors turn out to be.

7. Download the \( 201 \times 2 \) data matrix \( X.spiral \) and scan it into R. Use the R Package Installer to download and install the package \texttt{pracma}. Let \( q = 30000 \). The following R commands construct an isometrically equivalent configuration of \( n = 201 \) points in \( \mathbb{R}^q \):
\[
> library(pracma)
> q <- 30000
> A <- matrix(rnorm(q*2),ncol=2)
> A.QR <- gramSchmidt(A)
> Y <- X.spiral %*% t(A.QR$Q)
\]
(a) Vectors containing the \( n(n-1)/2 \) pairwise Euclidean distances of a configuration can be computed using the \texttt{dist} function. Investigate the distribution of pairwise distances in the \( X.spiral \) configuration, e.g., by computing the following:
\[
> dX <- dist(X.spiral)
> summary(dX)
\]
(b) Verify that the configurations \( Y \) and \( X.spiral \) are isometrically equivalent by showing that they have the same pairwise distances, e.g., by showing that their root mean squared error (RMSE) vanishes:
> dY <- dist(Y)
> m <- length(dX)
> rmse <- sqrt(sum((dY-dX)^2)/m)

(c) Investigate the extent of distortion introduced by the following random projection technique, which replaces the $n \times q$ data matrix $Y$ with the $n \times 10$ data matrix $Y_{10}$:

> Q10 <- matrix(rnorm(q*10,sd=sqrt(1/q)),ncol=10)
> Y10 <- Y %*% Q10

Because projection shortens distance, first compute a dilation factor $r$:

> dY10 <- dist(Y10)
> r <- sum(dY10*dY)/sum(dY10^2)

(Extra credit: in what sense is $r$ the optimal dilation factor?) Now compute the RMSE between $r*dY_{10}$ and $dY$.

(d) Construct the 2-dimensional principal components representation of $r*Y_{10}$, obtaining a configuration $Z$. Compute a dilation factor $s$, then compare the interpoint distances of $sZ$ to the interpoint distances of $X_{\text{spiral}}$. How does their RMSE compare to the magnitude of the interpoint distances of $X_{\text{spiral}}$? Comment on the loss of information entailed by preceding PCA with the random projection technique in part (c).
Part II

Constructing Euclidean Representations
Chapter 4

Embedding Inner Products

The methods described in this chapter extract principal component representations from proximity data. These methods can be viewed as extensions of the algorithm, summarized in Figure 2.1, that embeds a Euclidean inner product matrix of rank \( r \) in \( r \)-dimensional Euclidean space. In Section 4.1 we demonstrate that the principal component representation of \( x_1, \ldots, x_n \in \mathbb{R}^q \) can be extracted from their pairwise inner products, a technique known as kernel PCA. Section 4.2 extends kernel PCA to the case of similarities, interpreted as fallible inner products. The analogous extension to the case of dissimilarities, interpreted as fallible distances, is described in Section 4.3. The resulting technique of classical multidimensional scaling was proposed in 1952, long before Gower noted its connection to PCA in 1966. The essential ideas of kernel PCA have been with us for many decades.

4.1 Kernel PCA

Given \( x_1, \ldots, x_n \in \mathbb{R}^q \), let \( X \) denote the corresponding \( n \times q \) data matrix and let

\[
\tilde{X} = PX = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t,
\]

denote the singular value decomposition of the corresponding centered data matrix. Then

\[
B = \tilde{X} \tilde{X}^t = U \begin{bmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} U^t
\]

contains the pairwise inner products of the centered data and it follows from Theorem 2.5 that the best (in the sense of least squares) rank-\( d \) approximation of these inner products are the inner products

\[
B = \begin{bmatrix} U_d & \cdot \end{bmatrix} \begin{bmatrix} \Sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_d^t \\ \cdot \end{bmatrix} = \begin{bmatrix} U_d \Sigma_d & 0 \end{bmatrix} \begin{bmatrix} U_d \Sigma_d & 0 \end{bmatrix}^t.
\]

The approximating inner products are the pairwise inner products of the \( n \times d \) data matrix \( U_d \Sigma_d \), which happens to be the \( d \)-dimensional principal component representation of \( x_1, \ldots, x_n \in \mathbb{R}^q \). We thus obtain the following optimality property of PCA.
Suppose that the \( n \times n \) matrix \( C \) is symmetric and positive semidefinite.

1. Double center \( C \), i.e., compute the kernel matrix \( B = PCP \), where \( P = I - ee' / n \).

2. Compute \( \lambda_1 \geq \cdots \geq \lambda_d \geq 0 \), the \( d \) largest eigenvalues of \( B \), and corresponding orthonormal eigenvectors \( u_1, \ldots, u_d \).

3. Let \( \lambda_i = \sigma_i^2 \) and

\[
Y = \begin{bmatrix} \sigma_1 u_1 & \cdots & \sigma_d u_d \end{bmatrix} = \begin{bmatrix} y_1' \\ \vdots \\ y_n' \end{bmatrix}.
\]

Then \( Y \) is centered and its Cartesian coordinate axes are its principal component axes.

Figure 4.1: Constructing a \( d \)-dimensional principal component representation from a Euclidean inner product matrix \( C \).

**Theorem 4.1** Let

\[
\bar{X} = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^t,
\]

be a centered data matrix with pairwise inner products \( B = \bar{X} \bar{X}^t \). The \( d \)-dimensional principal component representation of \( X \) is the \( n \times d \) configuration matrix \( U_d \Sigma_d \), with pairwise inner products \( \bar{B} = U_d \Sigma_d^2 U_d^t \). Let \( Z \) denote any \( n \times d \) configuration matrix, with pairwise inner products \( ZZ^t \). Then

\[
\|\bar{B} - B\|_F^2 \leq \|ZZ^t - B\|_F^2.
\]

Because

\[
B = \bar{X} \bar{X}^t = (PX)(PX)^t = PX X^t P^t = PX X^t P,
\]

we see that principal component scores can be extracted directly from \( XX^t \), the \( n \times n \) matrix pairwise inner products between objects. The calculations are summarized in Figure 4.1.

If one begins with only pairwise inner products between objects, then it is not possible to represent the principal components as linear combinations of the (centered) original variables. In this context, the principal components are sometimes called principal coordinates. The fact that principal components can be derived from pairwise inner products was first noted by J. C. Gower [13] in 1966. Gower identified methods based on the \( q \times q \) matrix of pairwise inner products between variables, e.g., \( \bar{X} \bar{X} = (n-1)S \), as \( R \) methods and methods based on pairwise inner products between objects as \( Q \) methods. Today, \( Q \) methods are called kernel methods and the extraction of principal components/coordinates from inner products is called kernel PCA.
Kernel PCA allows us to extract a principal component representation from any centered Euclidean inner product matrix $B$. Furthermore, if we begin with pairwise Euclidean distances $D = [d_{ij}]$ and compute $D_2 = [d_{ij}^2]$, then we can apply kernel PCA to $B = \tau(D_2)$ to obtain a principal components representation. Finally, notice that the matrix $B = \tilde{X}\tilde{X}^t$ in Section 3.4 is the kernel matrix for PCA; hence, we only require the vector $b \in \mathbb{R}^n$ to extend the out-of-sample embedding formula (3.6) from PCA to kernel PCA.

To embed out-of-sample similarity data, suppose that the objects reside in $\Xi$ and that $\gamma : \Xi \times \Xi \to \mathbb{R}$ is a symmetric positive definite function, i.e., for any objects $x_1, \ldots, x_n \in \Xi$, the $n \times n$ matrix $\Gamma = \left[\gamma(x_i, x_j)\right]$ is symmetric and positive semidefinite. We center $\gamma$ with respect to $x_1, \ldots, x_n$ by computing

$$\tilde{\gamma}(u, v) = \gamma(u, v) - \frac{1}{n} \sum_{j=1}^{n} \gamma(u, x_j) - \frac{1}{n} \sum_{i=1}^{n} \gamma(x_i, v) + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma(x_i, x_j).$$

Given $x_1, \ldots, x_n$, kernel PCA replaces $B = \tilde{X}\tilde{X}^t$ with

$$B = \tilde{\Gamma} = P\Gamma P = \left[\tilde{\gamma}(x_i, x_j)\right].$$

The derivation of an out-of-sample embedding formula for the case of kernel PCA is then identical to the case of PCA. To embed another $x \in \Xi$ in the $d$-dimensional PC representation of $x_1, \ldots, x_n$, first compute

$$b = \begin{bmatrix} \tilde{\gamma}(x_1, x) \\ \vdots \\ \tilde{\gamma}(x_n, x) \end{bmatrix},$$

then

$$y = \Sigma_d^{-1}U_d^tb.$$ (4.1)

This version of (3.6) appears in [41].

### 4.2 Similarities as Fallible Inner Products

Let $\Gamma = [\gamma_{ij}]$ be a similarity matrix and suppose that we adopt the Euclidean inner product as our mathematical model of similarity. If $\Gamma \geq 0$, then we can proceed as in Figure 4.1. One reason for the popularity of certain similarity measures, e.g., the heat kernel of Example 1.3, is that they automatically generate similarity matrices that are positive semidefinite.

If $\Gamma$ is not positive semidefinite, then it is standard practice to approximate $B = P\Gamma P$ with a nearby positive semidefinite $\tilde{B}$. Notice that constructing a $d$-dimensional principal component representation from the spectral decomposition

$$U\Lambda U^t = \sum_{i=1}^{n} \lambda_i u_i u_i^t$$

is equivalent to constructing a representation from

$$\sum_{i=1}^{d} \lambda_i u_i u_i^t.$$  

Accordingly, we restrict attention to $\tilde{B}$ of rank no greater than $d$. 

---

**CHAPTER 4. EMBEDDING INNER PRODUCTS**  

64
The following least squares approximation is implicit in the seminal work of W. S. Torgerson [39]. A formal proof appears in [26]. Notice that this result extends Theorem 4.1 from the special case of $B = X\tilde{X}^t$ to any symmetric $B$.

**Theorem 4.2** Let 

$$B = U\Lambda U^t = \sum_{i=1}^{n} \lambda_i u_i u_i^t$$

be the spectral decomposition of the $n \times n$ symmetric matrix $B$. Define $\bar{\lambda}_i = \max(\lambda_i, 0)$ for $i = 1, \ldots, d$, $\bar{\lambda}_i = 0$ for $i = d+1, \ldots, n$, $\bar{\Lambda} = \text{diag}(\bar{\lambda}_1, \ldots, \bar{\lambda}_n)$, and 

$$\bar{B} = U\bar{\Lambda} U^t = \sum_{i=1}^{d} \bar{\lambda}_i u_i u_i^t.$$ 

If $\bar{C}$ is any $n \times n$ symmetric positive semidefinite matrix of rank $\leq d$, then 

$$\|\bar{B} - B\|_F^2 \leq \|\bar{C} - B\|_F^2.$$ 

Notice that the approximation in Theorem 4.2 preserves centering. If $B = PCP$, then $Be = PCP\bar{e} = \bar{0}$ and therefore 

$$Be = \sum_{i=1}^{n} \lambda_i u_i u_i^t e = \sum_{i=1}^{n} \lambda_i \left(u_i^t e\right) u_i = \bar{0}.$$ 

Because the $u_i$ are linearly independent, it must be that each coefficient $\lambda_i(u_i^t e) = 0$. Because each $\bar{\lambda}_i$ equals either $\lambda_i$ or 0, it follows that each $\bar{\lambda}_i(u_i^t e) = 0$, hence that $Be = \bar{0}$.

How might we embed out-of-sample similarity data if the similarity function $\gamma : \Xi \times \Xi \rightarrow \mathbb{R}$ is not positive positive? An obvious possibility is to replace $B$ with $\bar{B}$, compute 

$$\bar{B} = U\bar{\Lambda} U^t = \begin{bmatrix} U_d & \cdot \end{bmatrix} \begin{bmatrix} \Sigma_d & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_d^t \\ \cdot \end{bmatrix},$$

and use equations (4.1) and (4.2). This substitution is useful (and widely used), but it has subtle implications that we now explore.

Let us write the centered similarities of $x_1, \ldots, x_n, x \in \Xi$ as 

$$B^+ = \begin{bmatrix} B & b \\ b^t & \beta \end{bmatrix},$$

where $\beta = \bar{\gamma}(x, x)$. Embedding $B^+$ in $\mathbb{R}^d$ entails approximating $B^+$ with an inner product matrix of rank $d$. The out-of-sample problem fixes the positions of $y_1, \ldots, y_n \in \mathbb{R}^d$ obtained by embedding $\bar{B}$, resulting in the nonlinear optimization problem 

$$\min_{y \in \mathbb{R}^d} \left\| \begin{bmatrix} y_1^t \\ \vdots \\ y_n^t \end{bmatrix} \begin{bmatrix} y_1 & \cdots & y_n & y \end{bmatrix} - B^+ \right\|_F^2 = \min_{y \in \mathbb{R}^d} 2 \|Yy - b\|_2^2 + (y^t y - \beta)^2, \quad (4.3)$$
where $Y$ is the $n \times d$ configuration matrix corresponding to $y_1, \ldots, y_n$. Global solutions of (4.3) are exact solutions of the out-of-sample problem.

If the nonlinear term $(y^t y - \beta)^2$ is dropped from the objective function in (4.3), then what remains is a linear least squares problem with normal equations $Y^t Y \hat{y} = Y^t b$. Assuming that $Y$ has full rank (otherwise, a smaller $d$ will suffice), $Y^t Y$ is invertible and the unique solution of the normal equations is

$$
\hat{y} = (Y^t Y)^{-1} Y^t b = (\Sigma_d U_d^t U_d \Sigma_d)^{-1} \Sigma_d U_d^t b = \Sigma_d^{-1} U_d^t b,
$$

which is (4.2) again. Thus, (4.2) provides an approximate solution of the out-of-sample embedding problem (4.3). Example 4.2 demonstrates that these solutions can be quite different.

### 4.3 Dissimilarities as Fallible Distances

Let $\Delta = [\delta_{ij}]$ be a dissimilarity matrix and let $\Delta_2 = [\delta_{ij}^2]$. If $\Delta$ is EDM-1, then $B = \tau(\Delta_2)$ is the corresponding matrix of centered Euclidean inner products from which one can construct a principal component representation by kernel PCA. If we adopt Euclidean distance as our mathematical model of dissimilarity—if, in the phrasing of Torgerson [39], we conceive of the $\delta_{ij}$ as fallible distances—then it is natural to proceed as if $\Delta$ is EDM-1, approximating as necessary.

Following Torgerson [39], let $B = \tau(\Delta_2)$ and compute the positive semidefinite approximation $\bar{B}$ of Theorem 4.2. Then $B$ is centered, hence $\bar{B}$ is centered and we can construct a principal component representation from $\bar{B}$ by kernel PCA. The complete algorithm, known in psychometrics and statistics as classical multidimensional scaling (CMDS), is summarized in Figure 4.2.

**Example 4.1** Consider the western United States and the dissimilarity matrix $\Delta = [\delta_{ij}]$ in Table 4.1, for which the objects are $n = 22$ western states. As in Exercise 1.4.1, $\delta_{ij}$ is the minimal number of state boundaries that must be crossed to drive from state $i$ to state $j$. Because the states vary in size and shape, $\delta$ is a crude measure of distance between states. Can one possibly reconstruct the spatial relationships of the states from such crude information? Before attempting such a reconstruction, notice that $\Delta$ contains no information whatsoever about compass orientation.

From $\Delta$, CMDS constructs the 2-dimensional configuration displayed in Figure 4.3 on page 72. At first glance, this configuration disappoints. We must impose our own knowledge of compass orientation to appreciate the quality of this reconstruction. Reversing the sign of the horizontal axis (the first principal component), we obtain the more recognizable configuration displayed in Figure 4.4 on page 73. A slight rotation of this configuration results in the configuration in Figure 4.5 on page 74. This configuration is quite remarkable in its fidelity to an actual map of the western United States.

How is this possible? The individual $\delta_{ij}$ are extremely crude, but $\Delta$ contains a great many pairwise comparisons ($n(n-1)/2 = 231$). It is unlikely that a human being could extract the spatial relationships of the 22 states by direct inspection of $\Delta$. CMDS does precisely that. Comparing Figure 4.5 to Table 4.1, one is reminded of a familiar saying: “a picture is worth a thousand words” (or numbers).
CHAPTER 4. EMBEDDING INNER PRODUCTS

Suppose that $\Delta = [\delta_{ij}]$ is an $n \times n$ dissimilarity matrix.

1. Compute $\Delta_2 = [\delta_{ij}^2]$ and $B = \tau(\Delta_2)$.

2. Compute $\lambda_1 \geq \cdots \geq \lambda_d$, the $d$ largest eigenvalues of $B$, and corresponding orthonormal eigenvectors $u_1, \ldots, u_d$.

3. For $i = 1, \ldots, d$, let $\bar{\lambda}_i = \max(\lambda_i, 0)$.

4. Let $\bar{\lambda}_i = \sigma_i^2$ and

$$Y = \left[ \begin{array}{c|c} \sigma_1 u_1 & \cdots & \sigma_d u_d \end{array} \right] = \left[ \begin{array}{c} y_1^t \\ \vdots \\ y_n^t \end{array} \right].$$

Then $Y$ is centered and its Cartesian coordinate axes are its principal component axes.

To embed out-of-sample dissimilarity data, suppose that the objects reside in $\Xi$ and that $\delta : \Xi \times \Xi \to \mathbb{R}$ is a symmetric nonnegative function. Convert dissimilarities to centered similarities by

$$\tilde{\gamma}(u, v) = -\frac{1}{2} \left[ \delta^2(u, v) - \frac{1}{n} \sum_{j=1}^n \delta^2(u, x_j) - \frac{1}{n} \sum_{i=1}^n \delta^2(x_i, v) + \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \delta^2(x_i, x_j) \right],$$

then proceed as in Section 4.2. The calculations are summarized in Figure 4.6. Notice that the use of $\tau_w$ maintains the $y_1, \ldots, y_n \in \mathbb{R}^d$ constructed from $\Delta$ by CMDS.

Example 4.2 Suppose that objects $x_1, \ldots, x_4$ have pairwise squared dissimilarities

$$\Delta_2 = \left[ \begin{array}{cccc} 0 & 100 & 45 & 45 \\ 100 & 0 & 45 & 45 \\ 45 & 45 & 0 & 64 \\ 45 & 45 & 64 & 0 \end{array} \right].$$

We embed $\Delta_2$ in $\mathbb{R}^2$ by CMDS, obtaining the configuration matrix

$$Y = U_2 \Sigma_2 = \left[ \begin{array}{c} 5 \\ -5 \\ 0 \\ 0 \end{array} \right].$$

Now suppose that another object, $x$, has pairwise squared dissimilarities of

$$a_x^2 = \left[ \begin{array}{cccc} 386 & 386 & 457 & 457 \end{array} \right]$$
Table 4.1: Dissimilarities of \( n = 22 \) western states. Each \( \delta_{ij} \) is the minimal number of state boundaries that must be crossed to drive from state \( i \) to state \( j \).

<table>
<thead>
<tr>
<th>State</th>
<th>0 3 1 2 2 4 3 3 5 3 3 3 1 1 4 2 2 3 2 1 3 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arizona</td>
<td>3 0 4 2 4 2 2 1 3 1 4 2 4 2 4 1 5 3 1 3 5 3</td>
</tr>
<tr>
<td>Arkansas</td>
<td>1 4 0 3 2 5 4 4 5 4 3 4 1 2 4 3 1 4 3 2 2 3</td>
</tr>
<tr>
<td>California</td>
<td>2 2 3 0 2 2 1 3 3 2 1 2 1 3 1 3 2 2 1 3 1</td>
</tr>
<tr>
<td>Colorado</td>
<td>2 4 2 2 0 3 3 5 3 3 3 1 2 1 3 2 3 1 2 3 1 1</td>
</tr>
<tr>
<td>Idaho</td>
<td>4 2 5 2 3 0 2 3 1 1 2 1 4 3 2 2 4 1 3 3 4 2</td>
</tr>
<tr>
<td>Iowa</td>
<td>3 2 4 1 3 2 0 3 3 1 3 1 3 2 3 1 4 2 2 2 4 2</td>
</tr>
<tr>
<td>Kansas</td>
<td>3 1 4 3 5 3 3 0 4 2 5 3 4 2 5 2 5 4 1 4 6 4</td>
</tr>
<tr>
<td>Louisiana</td>
<td>5 3 5 3 3 1 3 4 0 2 2 2 4 4 1 3 4 1 4 3 4 2</td>
</tr>
<tr>
<td>Minnesota</td>
<td>3 1 4 2 3 1 1 2 2 0 3 1 4 2 3 1 4 2 2 3 4 2</td>
</tr>
<tr>
<td>Missouri</td>
<td>3 4 3 2 1 2 3 5 2 3 0 2 2 3 1 3 2 1 4 2 2 1</td>
</tr>
<tr>
<td>Montana</td>
<td>3 2 4 1 2 1 1 3 2 1 2 0 3 2 2 3 1 3 2 3 1</td>
</tr>
<tr>
<td>Nebraska</td>
<td>1 4 1 2 1 4 3 4 4 4 2 3 0 2 3 1 3 1 3 1 2 2</td>
</tr>
<tr>
<td>Nevada</td>
<td>1 2 2 1 3 3 2 2 4 2 3 2 2 0 4 1 3 3 1 2 4 2</td>
</tr>
<tr>
<td>New Mexico</td>
<td>4 4 4 3 2 2 3 5 1 3 1 2 3 4 0 4 3 1 5 3 3 2</td>
</tr>
<tr>
<td>North Dakota</td>
<td>2 1 3 1 3 2 1 2 3 1 3 2 3 1 4 0 4 3 1 2 4 2</td>
</tr>
<tr>
<td>Oklahoma</td>
<td>2 5 1 3 1 4 4 5 4 4 2 3 1 3 3 4 0 3 4 2 1 2</td>
</tr>
<tr>
<td>Oregon</td>
<td>3 3 4 2 2 1 2 4 1 2 1 1 3 1 3 1 3 0 4 2 3 1</td>
</tr>
<tr>
<td>South Dakota</td>
<td>2 1 3 2 4 3 2 1 4 2 4 3 3 1 5 1 4 0 3 5 3</td>
</tr>
<tr>
<td>Texas</td>
<td>1 3 2 1 1 3 2 4 3 3 2 2 1 2 3 2 2 3 0 2 1</td>
</tr>
<tr>
<td>Utah</td>
<td>3 5 2 3 1 4 4 6 4 4 2 3 2 4 3 4 1 3 5 2 0 2</td>
</tr>
<tr>
<td>Washington</td>
<td>2 3 3 1 1 2 2 4 2 2 1 1 2 2 2 2 1 3 1 2 0</td>
</tr>
</tbody>
</table>

from objects \( x_1, \ldots, x_4 \), from which we obtain \( b = \vec{0} \) and \( \beta = 400 \). The objective function

\[
 f(y) = 2 \| Yy - \vec{0} \|_2^2 + (y^t y - 400)^2
\]

has two global minimizers, at \( y_* = (0, \pm \sqrt{368} \approx \pm 19.2) \), whereas the approximate solution \( \hat{y} = \Sigma_d^{-1} U_d \vec{0} = (0, 0) \). The profound difference in solutions warrants careful examination.

The configuration \( y_1, y_2, y_3, y_4, y_* \) is the optimal representation of \( x_1, x_2, x_3, x_4, x \) in \( \mathbb{R}^2 \), subject to the restriction that \( x_1, x_2, x_3, x_4 \) must be represented by \( y_1, y_2, y_3, y_4 \). It is evident from \( a_2 \) that \( x \) is more dissimilar from any original object than any original object is from another. This feature is preserved by \( y_1, y_2, y_3, y_4, y_* \), whereas \( \hat{y} \) represents \( x \) as the centroid of the original objects.

Notice, however, that \( \Delta_2 \) can be embedded without error in \( \mathbb{R}^3 \), e.g., as

\[
\begin{bmatrix}
5 & 0 & 1 \\
-5 & 0 & 1 \\
0 & 4 & -1 \\
0 & -4 & -1
\end{bmatrix}
\]

This is the principal component representation of \( x_1, x_2, x_3, x_4 \), so the 2-dimensional PC representation is just \( Y \). One can then embed \( x \) in \( \mathbb{R}^3 \) without error as \((0, 0, 20)\) and observe that projecting \((0, 0, 20)\) into the span of the first two principal components of \( y_1, y_2, y_3, y_4 \) results in \( \hat{y} = \vec{0} \).

If one’s objective is to position an out-of-sample object in a space that contains it, then one should prefer \( y_* \) to \( \hat{y} \). However, if one’s objective is to project an out-of-sample object
into a space that does not contain it, then one may well prefer \( \hat{y} \) to \( y^* \). For example, Trosset and Priebe [?] studied inference on an unknown subfamily of probability distributions, e.g., a subfamily of multinomial distributions. First, a discrete representation of the subfamily is constructed in \( \mathbb{R}^d \) by a manifold learning procedure (see Chapter 7). Minimum distance estimation is then accomplished by an out-of-sample embedding of the empirical distribution. The nature of the procedure is such that small dissimilarities (of the empirical distribution from known distributions in the subfamily) can be estimated more accurately than large dissimilarities; hence, it is natural to attempt out-of-sample embedding using only the nearest neighbors of the out-of-sample object. In this context \( \hat{y} \) seems preferable to \( y^* \), as illustrated in Example 4.3.

Example 4.3 Figure 4.7 displays \( n = 20 \) objects as \( \bullet = x_1, x_2, x_3 \in \mathbb{R}^2 \) and \( \circ = x_4, \ldots, x_{20} \in \mathbb{R}^2 \). These objects represent known elements of an unknown 2-dimensional subfamily \( \mathcal{M} \) of probability distributions. An out-of-sample object \( x \) represents an empirical distribution that lies outside \( \mathcal{M} \).

Minimum distance estimation seeks to find the element of \( \mathcal{M} \) that is nearest \( x \) in a suitable metric. The pairwise distances between probability distributions can be approximated by pairwise dissimilarities, but the approximation is more accurate for small distances than for large distances. Hence, our out-of-sample embedding of \( x \) will rely on the \( x_i \) nearest \( x \).

The three nearest neighbors of \( x \) are

\[
X = \begin{bmatrix}
  x_1^t \\
  x_2^t \\
  x_3^t
\end{bmatrix} = \begin{bmatrix}
  0.42426191 & -0.0668329 \\
  -0.0209624 & 0.56856485 \\
  -0.73664728 & 0.03765010
\end{bmatrix},
\]

with pairwise squared dissimilarity matrix

\[
\Delta_2 = \begin{bmatrix}
  \|x_i - x_j\|^2
\end{bmatrix} = \begin{bmatrix}
  0.0000000 & 0.6019542 & 1.3586267 \\
  0.6019542 & 0.0000000 & 0.7940753 \\
  1.3586267 & 0.7940753 & 0.0000000
\end{bmatrix}.
\]

The squared dissimilarities of \( x \) from \( x_1, x_2, x_3 \) are

\[
a^t_2 = \begin{bmatrix}
  25.15667 & 25.39333 & 25.55864
\end{bmatrix}.
\]

Now we proceed as in Figure 4.6, obtaining

\[
b^t = (0.12713019, -0.08529552, -0.04183466) \quad \text{and} \quad \beta = 25.06347.
\]

The out-of-sample embeddings \( y_* \) and \( \hat{y} \) are computed with respect to the principal components of \( x_1, x_2, x_3 \). Transforming \( y_* \) and \( \hat{y} \) to the coordinate system in Figure 4.7, we obtain

\[
\bullet = (-0.01162794, -4.80347724) \quad \text{and} \quad \bullet = (0.01211158, -0.06817954).
\]

Recalling that \( x_1, x_2, x_3 \) are the nearest neighbors of \( x \), \( \bullet \) is evidently a more appealing minimum distance estimate than \( \bullet \).

The discrepancy between \( \bullet \) and \( \bullet \) arises because \( \bullet \) models \( \beta \) whereas \( \bullet \) does not. As in Example 4.2, \( \beta \) is large relative to \( \|x_i\|^2 \) \( (i = 1, 2, 3) \); hence, modeling \( \beta \) causes \( x \) to be embedded far from \( x_1, x_2, x_3 \), which in fact are the \( x_i \) known to be nearest \( x \). This defeats the purpose of minimum distance estimation, which tries to find the distribution in \( \mathcal{M} \) that is nearest \( x \). In this context, a large value of \( \beta \) only means that the empirical distribution does not lie in \( \mathcal{M} \).
4.4 More About Cosine Similarity

Suppose that $\Gamma = [\gamma_{ij}]$ is a similarity matrix and we adopt the Euclidean inner product as our mathematical model of similarity. If we proceed as if the $\gamma_{ij}$ are in fact Euclidean inner products, then the “standard transformation” $\Delta_2 = \kappa(\Gamma)$ converts the fallible inner products to fallible (squared) distances and one can proceed as in Section 4.3. Notice, however, that

$$B = \tau(\kappa(\Gamma)) = P\Gamma P,$$

so this approach is equivalent to the more straightforward approach of Section 4.2.

Nevertheless, we prefer to convert similarities to dissimilarities. While it is standard practice to obtain $B$ from $\Gamma$ by double centering $\Gamma$, we observed in Section 2.7 that $\kappa$ may not be the most appropriate transformation from similarity to (squared) dissimilarity. We find it easier to visualize and interpret distances and dissimilarities than to visualize and interpret inner products and similarities; accordingly, we favor first transforming $\Gamma$ to a suitable $\Delta$, then applying CMDS as described in Figure 4.2. To illustrate, we return to the case of cosine similarity.

Let $y_1, \ldots, y_n \in \mathbb{R}^q$ denote nonzero feature vectors with nonnegative entries. In Section 2.7 we considered applications for which the angle between two such feature vectors is a natural measure of dissimilarity. Angle dissimilarity depends on rescaling each $y_i$ to a vector of unit length, $z_i = y_i/\|y_i\|_2$. The cosine similarity of objects $i$ and $j$ is $\gamma_{ij} = \langle z_i, z_j \rangle$, so named because $\gamma_{ij}$ is the cosine of the angle between the feature vectors $y_i$ and $y_j$. The angle dissimilarity of objects $i$ and $j$ is $\alpha_{ij} = \arccos \gamma_{ij}$. Because the propriety of angle dissimilarity is the motivation for cosine similarity, $\alpha_{ij} = \arccos \gamma_{ij}$ is the natural transformation from cosine similarity to dissimilarity. Notice that the unit vectors $z_i$ and $z_j$ lie on the unit sphere and that $\alpha_{ij}$ is also the geodesic (great circle) distance between $z_i$ and $z_j$.

In practice, a popular transformation of cosine similarity to dissimilarity is $1 - \gamma_{ij}$. In Section 2.7 we argued that $1 - \gamma_{ij}$ should be interpreted as squared dissimilarity. The closely related dissimilarity

$$d_{ij} = \sqrt{2(1 - \gamma_{ij})}$$

is the chordal distance between $z_i$ and $z_j$. For small distances, chordal distance is a plausible approximation of geodesic distance.

We proceed to consider the implications of using CMDS to embed $\Delta = [\delta_{ij}]$ for $\delta_{ij} = \alpha_{ij}$ and $\delta_{ij} = d_{ij}$. Embedding $\Delta = [\alpha_{ij}]$ constructs a $d$-dimensional representation of (a portion of) the unit sphere in $\mathbb{R}^q$. In contrast, $d_{ij}$ is the $q$-dimensional Euclidean distance between $z_i$ and $z_j$, and therefore $\Delta = [d_{ij}]$ is EDM-1 with embedding dimension $p = q$. Hence, embedding $\Delta = [d_{ij}]$ by CMDS is equivalent to performing PCA on $z_1, \ldots, z_n \in \mathbb{R}^q$. As the following example illustrates, projecting $z_1, \ldots, z_n$ into a $d$-dimensional hyperplane may distort the data in ways that are inconsistent with the rationale that is usually invoked to justify the use of cosine similarity.

**Example 4.4** Let $y_i \in \mathbb{R}^2$ be such that

$$z_i = \frac{y_i}{\|y_i\|_2} = \left(\cos \frac{i}{20} \pi, \sin \frac{i}{20} \pi\right)$$

for $i = 1, \ldots, 19$. The $z_i$ are equally spaced on the unit circle with pairwise angles

$$\alpha_{ij} = \frac{|i-j| \pi}{20}.$$
The matrix of pairwise angles, \( \Delta = [\alpha_{ij}] \), is also the matrix of pairwise Euclidean distances of \( n = 19 \) equally spaced points on the real line; hence, \( \Delta \) is EDM-1 with embedding dimension \( p = 1 \) and the 1-dimensional principal component representation of \( \Delta \) comprises the equally spaced points

\[
x_i = \frac{i - 10 \pi}{20}.
\]

These points form the upper configuration in Figure 4.8 on page 76.

In contrast, \( \Delta = [\|z_i - z_j\|_2] \) is EDM-1 with embedding dimension \( p = 2 \). The first principal component axis passes through the centroid of the \( y_i \); by symmetry, its slope is \( -1 \). Projecting \( z_i \) into this 1-dimensional hyperplane, we obtain the 1-dimensional principal component representation of \( \Delta \),

\[
\hat{z}_i = \cos \left( \frac{i \pi}{20} + \frac{\pi}{4} \right) = \cos \left( \frac{i + 10 \pi}{20} \right).
\]

These points form the lower configuration in Figure 4.8. Notice that they are not equally spaced.

4.5 Distance Matrix Completion

4.6 Exercises

1. Verify that \( \tau(\kappa(\Gamma)) = P \Gamma P \).

2. Use CMDS to construct a 2-dimensional embedding of the dissimilarity matrix \( \Delta \) determined in Exercise 1.4.2. Comment on the fidelity (or lack thereof) of this representation to the actual spatial relationships of the 11 states in question.

3. Let \( \Delta \) denote the 15 \( \times \) 15 matrix of Congressional voting dissimilarities. Use the following R commands to construct a 3-dimensional embedding of \( \Delta \) by CMDS:

\[
> \text{Delta.cmds} <- \text{cmdscale}(\text{Delta}, k=3, \text{eig}=\text{TRUE})
> \text{X <- Delta.cmds}\$\text{points}
\]

(a) The eigenvalues of \( \tau(\Delta_2) \) are returned in \( \text{Delta.cmds}\$\text{eig} \). Let \( r \) denote the number of strictly positive eigenvalues. Compare the sums of the positive and negative eigenvalues. How well does an \( r \)-dimensional Euclidean representation of \( \Delta \) approximate the structure of \( \Delta \)?

(b) Divide the sum of the largest \( d = 2 \) positive eigenvalues by the sum of all \( r \) positive eigenvalues. Do the same for \( d = 3 \). Is \( d = 2 \) enough dimensions to approximate the Euclidean structure of \( \Delta \)? Is \( d = 3 \)? Why or why not?

(c) Construct pairwise scatterplots of \( X \). Do you discern any interesting patterns with \( d = 3 \) that are not apparent in \( d = 2 \)? What choice of \( d \) would you recommend?
Figure 4.3: The 2-dimensional principal component representation of 22 western states, constructed by CMDS from the dissimilarities in Table 4.1. The exact location of each state is the center of the rectangle that encloses the state’s postal abbreviation.
Figure 4.4: An alternative 2-dimensional principal component representation of the 22 western states, identical to the representation in Figure 4.3 except that the orientation of the horizontal axis has been reversed.
Figure 4.5: A slight rotation of the representation in Figure 4.4, resulting in a more familiar compass orientation.
1. Let $\Delta_2 = [\delta_{ij}^2]$ denote the squared dissimilarities of objects $x_1, \ldots, x_n$. Compute $\tau(\Delta_2) = B = U\Lambda U^t$,
\[
\bar{B} = U\bar{\Lambda} U^t = \begin{bmatrix} U_d & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_d^t & 0 \end{bmatrix}, \quad \text{and} \quad Y = U_d \Sigma_d.
\]

2. Let $a_2 \in \mathbb{R}^n$ denote the squared dissimilarities of object $x$ from objects $x_1, \ldots, x_n$ and set $w = (e^t/n, 0)$. Let
\[
A_2 = \begin{bmatrix} \Delta_2 & a_2 \\ a_2^t & 0 \end{bmatrix}
\]
and compute $\tau_w(A_2) = \begin{bmatrix} B & b \\ b^t & \beta \end{bmatrix}$,
where $\tau_w$ is defined by (2.3).

3. Solve the nonlinear optimization problem
\[
\min_{y \in \mathbb{R}^d} 2\|Yy - b\|^2_2 + (y^t \beta - \beta)^2,
\]
to obtain $y_*$, or approximate a solution by computing $\hat{y} = \Sigma_d^{-1}U_d^tb$.

Figure 4.6: Out-of-sample embedding via classical multidimensional scaling. The objects $x_1, \ldots, x_n$ are embedded as $y_1, \ldots, y_n \in \mathbb{R}^d$. The out-of-sample object $x$ is then embedded as $y \in \mathbb{R}^d$.

Figure 4.7: The out-of-sample embedding discussed in Example 4.3. Objects • and ○ are fixed, with an out-of-sample object nearest the former. Out-of-sample embedding based on the • can produce • or •.
Figure 4.8: Two 1-dimensional principal component representations constructed by CMDS from the cosine similarities in Example 4.4. The upper representation was constructed by embedding geodesic distances (angle dissimilarities) and preserves equal spacing; the lower representation was constructed by embedding chordal distances and distorts equal spacing.
Chapter 5

Embedding Distances

We have expressed a conceptual preference for techniques that embed dissimilarities as distances to techniques that embed similarities as inner products. Although CMDS is often described as a technique that embeds dissimilarities, in fact it is based on a least squares approximation involving inner products. We now turn to an embedding approach that directly approximates dissimilarities with distances. Although this approach is widely known in psychometrics and statistics, the machine learning community has tended to prefer embedding approaches that rely on spectral decompositions of kernel matrices. In fact, multidimensional scaling based on the raw stress criterion not only is conceptually appealing but also offers various pragmatic advantages.

5.1 Raw Stress Criterion

Let \( X = [x_{ij}] \) denote the \( n \times d \) configuration matrix associated with \( x_1, \ldots, x_n \in \mathbb{R}^d \). Let

\[
    d_{ij} = d_{ij}(X) = \|x_i - x_j\| = \left[ \sum_{\ell=1}^d (x_{i\ell} - x_{j\ell})^2 \right]^{1/2}
\]

denote the Euclidean distance between \( x_i \) and \( x_j \). Given an \( n \times n \) dissimilarity matrix \( \Delta = [\delta_{ij}] \), the weighted raw stress criterion is

\[
    \sigma(X) = \sum_{i \neq j} w_{ij} [d_{ij}(X) - \delta_{ij}]^2 = \frac{1}{2} \sum_{i,j=1}^n w_{ij} [d_{ij}(X) - \delta_{ij}]^2
\]

\[
    = \frac{1}{2} \sum_{i,j=1}^n w_{ij} d_{ij}^2(X) - \sum_{i,j=1}^n w_{ij} \delta_{ij}^2 d_{ij}(X) + \frac{1}{2} \sum_{i,j=1}^n w_{ij} \delta_{ij}^2,
\]

where \( w_{ij} = w_{ji} \geq 0 \) is the weight assigned to approximating \( \delta_{ij} = \delta_{ji} \) with \( d_{ij} = d_{ji} \). The configuration that minimizes the raw stress criterion is, in the sense of weighted squared error, the configuration whose interpoint distances best approximate the specified dissimilarities.

The incorporation of weights into the raw stress criterion provides enormous flexibility. While the selection of weights is often specific to the application, a variety of weighting schemes are of general interest. For example:

1. Equal weighting: \( w_{ij} = 1 \)
Apparently neutral, $w_{ij} = 1$ is widely used in practice, possibly because it seems to relieve the user of the burden of assigning weights. Furthermore, it simplifies computation. With equal weights, the raw stress criterion measures absolute squared error. Because errors of comparable magnitude will distort small dissimilarities more than large dissimilarities, equal weighting tends to represent large-scale structure better than small-scale structure. Often, this is precisely what the user desires: a good low-dimensional representation of general patterns at the expense of fine details.

2. Sammon weighting: $w_{ij} = 1 / \delta_{ij}$

Assuming that $\delta_{ij} = \|y_i - y_j\|_2$ for $y_1, \ldots, y_n \in \mathbb{R}^q$, Sammon [32] proposed the error criterion

$$E(X) = \frac{1}{c} \sum_{i \leftrightarrow j} \frac{[d_{ij} - \delta_{ij}]^2}{\delta_{ij}}, \quad \text{where} \quad c = \sum_{i \leftrightarrow j} \delta_{ij}.$$  

The constant $c$ does not affect the optimization of $E$. Sammon’s approach is widely used in certain communities, in which configurations that minimize $E$ are called Sammon maps.

Sammon did not comment on the implications of adopting a relative error criterion. Because pairs with small $\delta_{ij}$ receive greater weight than pairs with large $\delta_{ij}$, Sammon weighting pays more attention to fine detail than does equal weighting. Depending on the application, this may or may not be desirable.

3. JL weighting: $w_{ij} = 1 / \delta_{ij}^2$

The inequalities that appear in Definition 2.1 can be written as

$$(1 - \epsilon)\delta_{ij} \leq d_{ij}, \quad \text{i.e.,} \quad \frac{\delta_{ij} - d_{ij}}{\delta_{ij}} \leq \epsilon; \quad \text{and}$$

$$d_{ij} \leq (1 + \epsilon)\delta_{ij}, \quad \text{i.e.,} \quad \frac{d_{ij} - \delta_{ij}}{\delta_{ij}} \leq \epsilon; \quad \text{hence,}$$

$$\epsilon_{ij}^2 = \frac{[d_{ij} - \delta_{ij}]^2}{\delta_{ij}^2} \leq \epsilon^2.$$  

Johnson and Lindenstrauss [19] sought a dimension for which $\epsilon_{ij}^2 \leq \epsilon^2$ for all $ij$ pairs. In contrast, minimizing the raw stress criterion with JL weights minimizes the sum of the $\epsilon_{ij}^2$ in a fixed dimension, trading higher fidelity for lower dimension. The corresponding weighting scheme, however, places extraordinarily high weight on $ij$ pairs with small $\delta_{ij}$. If weighting small dissimilarities too heavily is a problem with Sammon weighting, then JL weighting will greatly exacerbate that problem.

4. Missing data.

If $\delta_{rs}$ is missing, then one can omit $r \leftrightarrow s$ in the sum over all $i \leftrightarrow j$ or, equivalently, set $w_{rs} = 0$. In practice, the ease with which the raw stress criterion can accommodate missing data is often a compelling reason for adopting it.

Let $G$ denote an undirected graph with vertices $x_1, \ldots, x_n$. If $w_{ij} = w_{ji} > 0$, then vertices $i$ and $j$ are connected with edge weight $w_{ij}$. We assume that $G$ is connected. Let $\epsilon_1, \ldots, \epsilon_n \in \mathbb{R}^n$ denote the unit coordinate vectors, let

$$E_{ij} = (\epsilon_i - \epsilon_j)(\epsilon_i - \epsilon_j)^t,$$
and notice that the combinatorial Laplacian matrix of \( G \) can be written as

\[
L = \sum_{i \leftrightarrow j} w_{ij} E_{ij} = \frac{1}{2} \sum_{i,j=1}^n w_{ij} E_{ij}.
\]  

(5.1)

Then

\[
X^t E_{ij} X = \begin{bmatrix} x_{11} & \cdots & x_{n1} \\ \vdots & \ddots & \vdots \\ x_{1d} & \cdots & x_{nd} \end{bmatrix} (e_i - e_j)(e_i - e_j)^t \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots \\ x_{n1} & \cdots & x_{nd} \end{bmatrix}
\]

\[
= \begin{bmatrix} x_{i1} - x_{j1} \\ \vdots \\ x_{id} - x_{jd} \end{bmatrix} \begin{bmatrix} x_{i1} - x_{j1} & \cdots & x_{id} - x_{jd} \end{bmatrix},
\]

hence

\[
\text{trace} \left( X^t E_{ij} X \right) = \sum_{\ell=1}^d (x_{i\ell} - x_{j\ell})^2 = d_{ij}^2(X)
\]

and

\[
\eta^2(X) = \text{trace} \left( X^t L X \right) = \frac{1}{2} \sum_{i,j=1}^n w_{ij} \text{trace} \left( X^t E_{ij} X \right) = \frac{1}{2} \sum_{i,j=1}^n w_{ij} d_{ij}^2(X).
\]

Next we modify \( G \), removing any edges that connect identical vertices, i.e., vertices \( x_i \) and \( x_j \) for which \( d_{ij} = \|x_i - x_j\| = 0 \), and assigning edge weights of \( w_{ij} \delta_{ij}/d_{ij} \). Let \( M(X) \) denote the combinatorial Laplacian matrix of the modified graph. Then the same reasoning leads us to conclude that

\[
\rho(X) = \text{trace} \left( X^t M(X) X \right) = \frac{1}{2} \sum_{i,j=1}^n \frac{w_{ij} \delta_{ij}}{d_{ij}(X)} d_{ij}(X) = \frac{1}{2} \sum_{i,j=1}^n w_{ij} \delta_{ij} d_{ij}(X),
\]

and it follows that minimizing \( \sigma(X) \) is equivalent to minimizing

\[
\eta^2(X) - 2\rho(X) = \text{trace} \left( X^t L X \right) - 2 \text{trace} \left( X^t M(X) X \right).
\]

### 5.2 Optimization Strategy

This section sketches a strategy for embedding dissimilarities using the raw stress criterion.

1. Initialization

   The methods that we will describe for stress reduction require an initial configuration of points in \( \mathbb{R}^d \). One possibility is to use CMDS to construct an initial configuration, but doing so requires the calculation of the \( d \) largest eigenvalues and corresponding eigenvectors of a symmetric \( n \times n \) matrix. For large \( n \), such calculations may be prohibitively expensive. Instead, Section 5.3 describes a heuristic method that uses \( O(n) \) flops to construct a plausible initial configuration.

2. Stress Reduction

   Reducing the value of the raw stress criterion requires numerical optimization. We will proceed iteratively, exploiting the structure of the objective function to successively...
improve the current configuration. Section 5.4 describes the technique of Guttman majorization (GMA), whereas Section 5.5 describes the technique of diagonal majorization (DMA). Both techniques reduce the value of the raw stress criterion when applied to any nonstationary configuration matrix, i.e., any \( X_k \) for which \( \nabla \sigma \neq \vec{0} \). Both techniques are globally convergent, in the sense that repeated application is guaranteed to converge to a connected set of stationary configurations.

Because GMA and DMA create monotone decreasing sequences of raw stress values, it is extremely unlikely that the corresponding sequences of configurations will converge to stationary configurations that are not local minimizers of the raw stress criterion. They may, however, converge to local minimizers that are not global minimizers. Which local minimizer one finds will depend on the initial configuration from which one starts, so there is evident value in trying to construct an initial configuration with a reasonably small raw stress value.

3. Termination

Both GMA and DMA can be viewed as weighted gradient methods for numerical optimization. Convergence tends to be slow. If convergence to a solution is desired, then it makes sense to follow several iterations of GMA or DMA with a globalized Newton method and use a traditional stopping criterion, as in [20]. If the dissimilarities are only approximate, or if \( n \) is very large, then an approximate solution may suffice. We will adopt the latter perspective, terminate stress reduction after a finite number of iterations, and settle for a good but suboptimal embedding.

5.3 Initialization by Landmark MDS

Let \( X_0 \) denote the initial \( n \times d \) configuration matrix from which stress reduction will commence. A popular practice is to set \( X_0 \) equal to the inner product embedding provided by CMDS. This typically works well, at least in the case of equal weights. Unfortunately, CMDS requires calculation of the \( d \) largest eigenvalues and corresponding eigenvectors of \( B = \tau(\Delta^2) \). If \( n \) is large, then these calculations may be expensive.

It is easy to construct inexpensive \( X_0 \), e.g., by using a random number generator to fill each entry of \( X_0 \). Unless one is extraordinarily lucky, however, \( X_0 \) will be nowhere near a global minimizer of \( \sigma \). The present section describes a compromise.

If the computational expense of embedding a large number of objects is prohibitive, then one may elect to construct an initial configuration by embedding a subset of reference objects, then individually position the remaining objects with respect to the reference objects by out-of-sample embedding. This technique, once called the method of standards, was pioneered by Kruskal and Hart [22], who successfully embedded \( n = 10^4 \) objects (in 1966!!) by first embedding \( r = 200 \) reference objects. Our implementation of the method of standards will rely on the embedding techniques described in Section 4, resulting in landmark multidimensional scaling (LMDS) [8, 9]. Notice, however, that we use LMDS not to produce a final configuration, but as a prelude to stress reduction. Thus, we set \( X_0 \) equal to the configuration matrix \( Y \) constructed in Figure 5.1, which provides a template for LMDS.

Step 1 in Figure 5.1 contains two crucial ambiguities, viz., the choices of \( r \) and \( \ell_1, \ldots, \ell_r \). To use LMDS effectively, one needs useful heuristics for choosing the reference (landmark) objects. De Silva and Tenenbaum [9] observed that
Let $\Delta$ denote a dissimilarity matrix for $n$ objects, to be embedded as $y_1, \ldots, y_n \in \mathbb{R}^d$.

1. Choose reference objects $\ell_1, \ldots, \ell_r$. Let $\Delta_2[\ell, \ell]$ denote the $r \times r$ matrix of squared pairwise dissimilarities for these objects.

2. Compute $\tau (\Delta_2[\ell, \ell]) = B = U \Lambda U^t$, 

$$
\tilde{B} = U \tilde{\Lambda} U^t = \begin{bmatrix} \Sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_d \\ 0 \end{bmatrix}, \quad \text{and} \quad Y_r = U_d \Sigma_d.
$$

Set $y_{\ell_i}$ equal to row $i$ of $Y_r$.

3. Compute $C = \Sigma_d^{-1} U_d^t$, 

$$
\mu_j = \frac{1}{r} \sum_{k=1}^{r} \delta^2(\ell_j, \ell_k), \quad \text{and} \quad \bar{\mu} = \frac{1}{r} \sum_{h=1}^{r} \mu_h.
$$

4. For each object $i \not\in \{\ell_1, \ldots, \ell_r\}$, compute

$$
\nu_i = \frac{1}{r} \sum_{h=1}^{r} \delta^2(\ell_h, i),
$$

$$
b = -\frac{1}{2} \begin{bmatrix} \delta^2(\ell_1, i) - \mu_1 - \nu_i + \bar{\mu} \\ \vdots \\ \delta^2(\ell_r, i) - \mu_r - \nu_i + \bar{\mu} \end{bmatrix}, \quad \text{and}
$$

$$
y_i = C b.
$$

Figure 5.1: Landmark MDS.

“For a $d$-dimensional embedding, we require at least $d + 1$ landmarks; specifically the affine span of the [embedded] landmarks...must be $d$-dimensional. For stability reasons it is better to avoid configurations which lie close to some $(d-1)$—or lower-dimensional affine subspace. It is generally advisable to to choose rather more landmark points than the strict minimum.”

They further

“suggest two ways of selecting the landmark set:

- Random choice. [Specifically, each set is equally likely to be selected.]

- MaxMin (greedy optimisation): landmark points are chosen one at a time, and each new landmark maximises, over all unused data points, the minimum distance to any of the existing landmarks. The first point is chosen arbitrarily.”

However, they report no numerical experiments with MaxMin.
5.4 Stress Reduction by Guttman Majorization

To motivate an algorithm for minimizing the raw stress criterion, we first derive a standard form of the stationary equation $\nabla \sigma(X) = \vec{0}$. Assuming that each pairwise distance is strictly positive, the partial derivatives of $\sigma$ are

$$
\frac{\partial}{\partial x_{rs}} \sigma(X) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \sum_{k=1}^{d} (x_{ik} - x_{jk})^2 - \frac{\partial}{\partial x_{rs}} \sum_{i,j=1}^{n} w_{ij} \delta_{ij} \left[ \sum_{k=1}^{d} (x_{ik} - x_{jk})^2 \right]^{1/2}
$$

$$
= \sum_{j=1}^{n} w_{rj} (x_{rs} - x_{js}) + \sum_{i=1}^{d} w_{ir} (x_{is} - x_{rs}) (1) + \sum_{j=1}^{n} w_{rj} \frac{\delta_{rj}}{d_{rj}} (x_{rs} - x_{js}) + \sum_{i=1}^{d} w_{ir} \frac{\delta_{ir}}{d_{ir}} (x_{is} - x_{rs}) (1)
$$

$$
= 2 \sum_{k=1}^{n} w_{rk} (x_{rs} - x_{ks}) + 2 \sum_{k=1}^{n} w_{rk} \frac{\delta_{rk}}{d_{rk}} (x_{rs} - x_{ks}),
$$

which vanish iff

$$
\sum_{k=1}^{n} w_{rk} (x_{rs} - x_{ks}) = \sum_{k=1}^{n} w_{rk} \frac{\delta_{rk}}{d_{rk}} (x_{rs} - x_{ks}).
$$

The latter equations can be written in matrix form as $LX = M(X)X$. For these equations to hold, it must be that $d_{rk} > 0$ if $w_{rk} \delta_{rk} > 0$. If that is the case, then $\sigma$ is smooth at $X$ and $LX = M(X)X$ is a necessary condition for $X$ to be a local minimizer of $\sigma$. Accordingly, solutions of $LX = M(X)X$ are plausible candidates for minimizers of $\sigma$.

5.4.1 Guttman Transform

The representation of $\nabla \sigma(X) = \vec{0}$ as $LX = M(X)X$ suggests the possibility of an iterative method for finding a stationary configuration: given a configuration matrix $X_k$, choose $X_{k+1}$ to solve $LX = M(X_k)X_k = M_k$. This system of equations does not have a unique solution, for recall that $Le = \vec{0}$ and therefore $L$ is singular. Which solution shall we choose?

Because the weighted graph $G$ is connected, its combinatorial Laplacian matrix $L$ has exactly one zero eigenvalue with corresponding eigenvector $e/\sqrt{n}$. The spectral decomposition of $L$,

$$
L = \begin{bmatrix}
    u_1 & \cdots & u_{n-1} & e/\sqrt{n}
\end{bmatrix}
\begin{bmatrix}
    \lambda_1 & & & \\
    & \ddots & & \\
    & & \lambda_{n-1} & \\
    & & & 0
\end{bmatrix}
\begin{bmatrix}
    u_1^t \\
    \vdots \\
    u_{n-1}^t \\
    e^t/\sqrt{n}
\end{bmatrix},
$$

is also its singular value decomposition; hence, the Moore-Penrose pseudoinverse of $L$ is

$$
L^+ = \begin{bmatrix}
    u_1 & \cdots & u_{n-1} & e/\sqrt{n}
\end{bmatrix}
\begin{bmatrix}
    1/\lambda_1 & & & \\
    & \ddots & & \\
    & & 1/\lambda_{n-1} & \\
    & & & 0
\end{bmatrix}
\begin{bmatrix}
    u_1^t \\
    \vdots \\
    u_{n-1}^t \\
    e^t/\sqrt{n}
\end{bmatrix}.
$$

---

1 This equation can also be derived by taking the matrix derivative of $\eta^2(X) - 2\rho(X)$. 
Consider the solution of \( LX = M_k \) defined by \( \bar{X} = L^\dagger M_k \) and notice that \( e^t \bar{X} = e^t L^\dagger M_k = 0 \), i.e., \( \bar{X} \) is centered. Let \( \bar{x} \) denote a column of \( \bar{X} \) and let \( m \) denote the corresponding column of \( M_k \), so that \( \bar{x} \) solves \( Lx = m \). Then \( Lx = m \) iff \( x = \bar{x} + \alpha e \) and it follows that \( \bar{x} \) is the only centered solution of \( Lx = m \). Applying this reasoning to each column of \( \bar{X} \), we conclude that \( \bar{X} \) is the only centered solution of \( LX = M_k \).

Although the choice of solutions is arbitrary, it is convenient to work with centered configurations; accordingly, we choose

\[
X_{k+1} = L^\dagger M_k X_k = \Gamma(X_k).
\]

Following de Leeuw and Heiser [7], we refer to \( \Gamma \) as the *Guttman transform*.\(^2\) Because \( e^t X_{k+1} = 0 \), we also have

\[
\left( L + ee^t \right) X_{k+1} = LX_{k+1} + ee^t X_{k+1} = LX_{k+1} = M_k X_k.
\]

It follows that we can write the Guttman transform as

\[
\Gamma(X_k) = \left( L + ee^t \right)^{-1} M_k X_k,
\]

avoiding the use of a matrix pseudoinverse.

The immediate virtue of the Guttman transform is that its application to any nonstationary configuration decreases the value of the raw stress criterion. More precisely...

**Theorem 5.1** Fix \( \Delta \) and \( W \). Let \( X_k \) be any configuration matrix and set \( X_{k+1} = \Gamma(X_k) \). Then \( \sigma(X_{k+1}) \leq \sigma(X_k) \), with strict inequality if \( LX_k \neq M_k X_k \).

Theorem 5.1 is a consequence of two lemmas, the application of which illustrates a technique known as *majorization*.

**Lemma 5.1** Fix \( \Delta \) and \( W \). For any two \( n \times d \) configuration matrices \( X \) and \( Z \),

\[
-\rho(X) \leq -\text{trace} \left( X^t M(Z) Z \right).
\]

**Proof** Applying the Cauchy-Schwartz Inequality,

\[
\left| \text{trace} \left( X^t E_{ij} Z \right) \right| = \left| \sum_{\ell=1}^d (x_{i\ell} - \bar{x}_{i\ell}) (z_{i\ell} - \bar{z}_{i\ell}) \right| \\
\leq \left[ \sum_{\ell=1}^d (x_{i\ell} - \bar{x}_{i\ell})^2 \right]^{1/2} \left[ \sum_{\ell=1}^d (z_{i\ell} - \bar{z}_{i\ell})^2 \right]^{1/2} \\
= d_{ij}(X)d_{ij}(Z).
\]

\(^2\)The use of \( \Gamma \) to denote the Guttman transform is traditional in the literature on multidimensional scaling. We rely on context to distinguish the Guttman transform from a matrix of pairwise similarities.
Noting that $\text{trace}(X^tE_{ij}Z) = 0$ if $d_{ij}(Z) = 0$, we then have

$$\text{trace}(X^tM(Z)Z) \leq \left|\text{trace}(X^tM(Z)Z)\right|$$

$$\leq \frac{1}{2} \sum_{d_{ij}(Z) > 0} \frac{w_{ij}\delta_{ij}}{d_{ij}(Z)} \left|\text{trace}(X^tE_{ij}Z)\right|$$

$$\leq \frac{1}{2} \sum_{d_{ij}(Z) > 0} \frac{w_{ij}\delta_{ij}}{d_{ij}(Z)} d_{ij}(X)d_{ij}(Z)$$

$$\leq \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}\delta_{ij}d_{ij}(X) = \rho(X).$$

$$\square$$

**Lemma 5.2** Fix $M$. The unique minimizer of the function

$$\tau_M(X) = \text{trace}\left[X^t\left(L + ee^t\right)X\right] - 2 \text{trace}\left(X^tM\right)$$

is $X_* = (L + ee^t)^{-1}M$.

**Proof** Let $\tilde{L} = L + ee^t$ and recall that $\tilde{L}$ is symmetric and positive definite. Define a matrix norm $\| \cdot \|_L$ by $\|A\|_L^2 = \text{trace}\left(A^t\tilde{L}^{-1}A\right)$. Then $X_*$ is evidently the unique minimizer of

$$\begin{align*}
\|\tilde{L}X - M\|^2_L &= \text{trace}\left[(\tilde{L}X - M)^t\tilde{L}^{-1}(\tilde{L}X - M)\right] \\
&= \text{trace}\left[X^t\tilde{L}^{-1}\tilde{L}X - X^t\tilde{L}^{-1}M - M^t\tilde{L}^{-1}\tilde{L}X + M^t\tilde{L}^{-1}M\right] \\
&= \text{trace}\left(X^t\tilde{L}X\right) - 2 \text{trace}\left(X^tM\right) + \text{trace}\left(M^t\tilde{L}^{-1}M\right) \\
&= \tau_M(X) + \text{trace}\left(M^t\tilde{L}^{-1}M\right)
\end{align*}$$

$$\square$$

**Proof of Theorem 5.1** We need to prove that $\sigma(X_{k+1}) < \sigma(X_k)$ if $LX_k \neq M(X_k)X_k$. Without loss of generality assume that $X_k$ is centered, in which case $(L + ee^t)X_k = LX_k$. As noted in Section 5.1, it suffices to demonstrate that

$$\eta^2(X_{k+1}) - 2 \rho(X_{k+1}) < \eta^2(X_k) - 2 \rho(X_k).$$

Because $X_{k+1}$ is centered, $LX_{k+1} = (L + ee^t)X_{k+1}$; hence, applying Lemma 5.1 with $X = X_{k+1}$ and $Z = X_k$, we obtain

$$\eta^2(X_{k+1}) - 2 \rho(X_{k+1}) \leq \text{trace}\left(X_{k+1}^tLX_{k+1}\right) - 2 \text{trace}\left(X_{k+1}^tM(X_k)X_k\right) = \tau_M(X_{k+1})$$

for $M = M(X_k)X_k$. Because $X_{k+1} = (L + ee^t)^{-1}M$, it then follows from Lemma 5.2 that

$$\begin{align*}
\tau_M(X_{k+1}) &< \tau_M(X_k) \\
&= \text{trace}\left[X_k^t\left(L + ee^t\right)X_k\right] - 2 \text{trace}\left(X_k^tM(X_k)X_k\right) \\
&= \eta^2(X_k) - 2 \rho(X_k).
\end{align*}$$
CHAPTER 5. EMBEDDING DISTANCES

graph.laplacian <- function(W) {
    deg <- apply(W,1,sum)
    return(diag(deg)-W)
}

mds.gma <- function(X,Delta,W,k=20) {
    L <- graph.laplacian(W)
    R <- chol(L+1)
    Linv <- chol2inv(R)
    for (i in 1:k) {
        m <- as.dist(W)*as.dist(Delta)/dist(X)
        X <- Linv %*% graph.laplacian(as.matrix(m)) %*% X
    }
    return(X)
}

Figure 5.2: R functions for Guttman majorization. The user calls mds.gma() with an initial configuration (X), a dissimilarity matrix (Delta), a weight matrix (W), and (optionally) a fixed number of iterations (k).

Starting from an initial configuration $X_0$, the Guttman Majorization Algorithm (GMA) constructs the sequence $X_1, X_2, \ldots$ defined by $X_k = \Gamma(X_{k-1})$. By Theorem 5.1, the raw stress values of a GMA sequence decrease unless a fixed point of $LX = M(X)X$ is encountered, and it was demonstrated by de Leeuw [6] that the GMA sequence converges to a connected set of fixed points. If each $w_{ij}\delta_{ij} > 0$, then fixed points are stationary points and, because the sequence of raw stress values is nondecreasing, it is typically the case in practice that the GMA sequence converges to a connected set of local minimizers. However, a local minimizer of $\sigma$ may not be a global minimizer of $\sigma$. To improve the prospect of finding a global minimizer, one should attempt to find a good initial configuration and/or start GMA from multiple initial configurations.

5.4.2 Computation

A general implementation of GMA in the statistical programming language R appears in Figure 5.2. The dominant expense of GMA lies in the Cholesky factorization used to invert $L + ee^t$. In some cases, $L^\dagger$ can be computed explicitly and Guttman majorization is considerably less expensive.

Equal Weights

In the important special case of equal weights ($w_{ij} = c$), we obtain

$$L = c \sum_{i<j} (e_i - e_j) (e_i - e_j)^t = cn \left( I - \frac{ee^t}{n} \right) = cnP,$$

$$L^\dagger = \frac{1}{cn} P^\dagger = \frac{1}{cn} P,$$

and

$$M(X)X = c \sum_{i<j} \frac{\delta_{ij}}{d_{ij}(X)} (e_i - e_j) (e_i - e_j)^t X.$$
Because
\[ P(e_i - e_j) = (e_i - e_j) - \frac{e}{n} e^t (e_i - e_j) = e_i - e_j, \]
\[ \Gamma(X) = L^\dagger M(X)X = M(X)X \] and therefore
\[ \Gamma(X) = \frac{1}{n} \sum_{i<j} \frac{\delta_{ij}}{d_{ij}(X)} \left( \begin{array}{c} y_{ij1} \\
\vdots \\
y_{ijn} \end{array} \right), \] where \( y_{ij} = \begin{cases} x_i - x_j & s = i \\
x_j - x_i & s = j \\
0 & s \neq i, j \end{cases} \).

5.5 Stress Reduction by Diagonal Majorization

5.6 Numerical Experiments

5.7 Exercises

1. Verify equation (5.1), then use it to prove that \( L \) is symmetric and positive semidefinite.

2. Verify that equations (5.2) are equivalent to the matrix equation \( LX = M(X)X \).

3. Let \( L \) denote the combinatorial Laplacian matrix of a connected graph. Show that \( L + ee^t \) is positive definite, hence invertible.

4. Verify that
\[ \sum_{i<j} (e_i - e_j) (e_i - e_j)^t = n \left( I - \frac{ee^t}{n} \right). \]
Chapter 6

Embedding Graphs

The embedding methods developed in Chapters 4–5 can be used to construct Euclidean representations of graphs. The objects to be embedded are the vertices of the graph, the edges and edge weights of which convey information about the proximity of vertex pairs. How one embeds a graph depends on how one interprets its edge structure.

Let $G$ be a connected weighted directed graph. If $G$ is unweighted and $i \rightarrow j$ is an edge, then we assign edge weight $w_{ij} = 1$. If $G$ is undirected, then we replace it with a directed graph by replacing each undirected edge $i \leftrightarrow j$ with the directed edges $i \rightarrow j$ and $j \rightarrow i$, each of which is assigned the weight of $i \leftrightarrow j$. Thus, we can interpret a weighted undirected graph as a weighted directed graph with symmetric edge weights $w_{ij} = w_{ji}$. We need not exclude loops, but the graphs that interest us typically are simple. In the sections that follow, it is critical to distinguish graphs with edge weights that measure vertex dissimilarity from graphs with edge weights that measure vertex similarity.

6.1 Embedding Shortest Path Distances

Assume that the edge weights of $G$ measure dissimilarity and let $\delta_{ij}$ denote the shortest path distance between vertices $i$ and $j$. It is natural to approximate shortest path distances with Euclidean distances, embedding $\Delta = [\delta_{ij}]$ in $\mathbb{R}^d$ by minimizing a weighted raw stress criterion:

$$\sigma(X) = \frac{1}{2} \sum_{i,j=1}^{n} u_{ij} [d_{ij}(X) - \delta_{ij}]^2.$$ 

We emphasize that summation in $\sigma$ is over all pairs of vertices, not just the pairs connected by edges. The rationale for computing and embedding shortest path distances is that doing so uses the structure of $G$ to impute the dissimilarities between pairs of vertices that are not connected.

In contrast, we might treat dissimilarities between unconnected pairs of vertices as missing data. Suppose that $G$ is undirected, so that $w_{ij} = w_{ji}$. We might then embed the edge weights

---

1We use the phrase *graph embedding* to emphasize that the Euclidean structure of the representation is intended to approximate certain features of the graph. In contrast, *graph layout* (or *graph drawing*) constructs 2-dimensional pictorial representations of graph connectivity, sometimes called *node-link diagrams*, according to various usability and aesthetic criteria. Figure 1.2 displays two elementary node-link diagrams.
by setting \( u_{ij} = u_{ji} = 0 \) if vertices \( i \) and \( j \) are not connected, then minimizing

\[
\sigma_0(X) = \sum_{i 
eq j} u_{ij} [d_{ij}(X) - w_{ij}]^2 = \frac{1}{2} \sum_{i,j=1}^n u_{ij} [d_{ij}(X) - w_{ij}]^2.
\]

The summation in \( \sigma_0 \) is only over the edges of \( G \). Even if \( \delta_{ij} = w_{ij} \) for every edge in \( G \), minimizing \( \sigma \) is not equivalent to minimizing \( \sigma_0 \). Imputing missing dissimilarities by shortest path distance exploits the structure of \( G \) to derive information that may (or may not) be desired. Example 6.1 illustrates the distinction.

**Example 6.1** Let \( i = 1, \ldots, 6 \), \( \theta_i = (i - 1)\pi/3 \), and \( y_i = (\cos \theta_i, \sin \theta_i) \in \mathbb{R}^2 \), plotted in Figure 6.1 (left). Some of the pairwise Euclidean distances between \( y_i \) and \( y_j \) are measured, others are not, resulting in the following partial dissimilarity matrix:

\[
\begin{bmatrix}
0 & 1 & b & 2 & ? & ? \\
1 & 0 & 1 & b & 2 & ? \\
b & 1 & 0 & 1 & b & 2 \\
2 & b & 1 & 0 & 1 & ? \\
? & 2 & b & 1 & 0 & ? \\
? & ? & 2 & b & 1 & 0 \\
\end{bmatrix},
\]

where \( b^2 = 3 \). Despite the missing measurements, we can recover a configuration that is isometrically equivalent to \( Y \) by minimizing

\[
\sigma_0(X) = (d_{12}(X) - 1)^2 + (d_{13}(X) - b)^2 + (d_{14}(X) - 2)^2 + (d_{23}(X) - 1)^2 + (d_{24}(X) - b)^2 + (d_{25}(X) - 2)^2 + (d_{34}(X) - 1)^2 + (d_{35}(X) - b)^2 + (d_{36}(X) - 2)^2 + (d_{45}(X) - 1)^2 + (d_{46}(X) - b)^2 + (d_{56}(X) - 1)^2.
\]

In contrast, consider an undirected graph with \( n = 6 \) vertices and the following edge weights:

\[
w_{12} = w_{23} = w_{34} = w_{45} = w_{56} = 1, \\
w_{13} = w_{24} = w_{35} = w_{46} = b, \\
w_{14} = w_{25} = w_{36} = 2
\]
Compute the shortest path distance between each pair of vertices, obtaining the dissimilarity matrix
\[
\begin{bmatrix}
0 & 1 & b & 2 & 3 & b+2 \\
1 & 0 & 1 & b & 2 & 3 \\
b & 1 & 0 & 1 & b & 2 \\
2 & b & 1 & 0 & 1 & b \\
3 & 2 & b & 1 & 0 & 1 \\
b+2 & 3 & 2 & b & 1 & 0
\end{bmatrix}
\]
then minimize
\[
\sigma(X) = (d_{12}(X) - 1)^2 + (d_{13}(X) - b)^2 + (d_{14}(X) - 2)^2 + (d_{15}(X) - 3)^2 + (d_{16}(X) - b - 2)^2 + (d_{23}(X) - 1)^2 + (d_{24}(X) - b)^2 + (d_{25}(X) - 2)^2 + (d_{26}(X) - 3)^2 + (d_{34}(X) - 1)^2 + (d_{35}(X) - b)^2 + (d_{36}(X) - 2)^2 + (d_{45}(X) - 1)^2 + (d_{46}(X) - b)^2 + (d_{56}(X) - 1)^2.
\]

The configuration displayed in Figure 6.1 (right) was obtained by first constructing an initial configuration by CMDS, then performing 200 successive Guttman iterations.

Constructing a Euclidean representation of the shortest path distances of a neighborhood graph is an especially important example of graph embedding.

Example 6.2 Consider the \( n = 201 \) points in \( \mathbb{R}^2 \) displayed in Figure 1.1(a) and let \( e_{ij} \) denote the Euclidean distance between points \( i \) and \( j \). Let \( \mathcal{G} \) denote the corresponding \( \epsilon \)-neighborhood graph displayed in Figure 1.1(b), with edge weights \( e_{ij} \). Let \( \Delta = [\delta_{ij}] \) denote the matrix of shortest path distances between the vertices of \( \mathcal{G} \). Embed \( \Delta \) in \( \mathbb{R}^2 \) by using CMDS to construct \( X_0 \), then performing 20 successive iterations of the Guttman transform with equal weights (\( u_{ij} = 1 \)). The resulting configuration is displayed in Figure 6.2. This construction exemplifies the Isomap procedure for manifold learning, further discussed in Chapter 7.

6.2 Adjacency Spectral Embedding

6.3 Random Walks on Graphs

Assume that the edge weights of \( \mathcal{G} \) measure similarity and set \( \gamma_{ij} = w_{ij} \).

A random walk on \( \mathcal{G} \) is a walk with randomly generated steps. A random walk produces a sequence of vertices indexed by a time parameter, \( s = 0, 1, 2, \ldots \). If the vertex visited at time \( s \) is \( i \), then the transition probability of stepping from vertex \( i \) to vertex \( j \) is \( p_{ij}(s) \). The possible steps are constrained by the edge structure of \( \mathcal{G} \), i.e., if \( i \to j \) is not an edge, then \( p_{ij}(s) = 0 \). If the transition probabilities do not vary with \( s \), i.e., \( p_{ij}(s) = p_{ij} \), then the random walk is stationary and \( P = [p_{ij}] \) is the matrix of transition probabilities.

A random walk on \( \mathcal{G} \) is a finite Markov chain whose states are the vertices of \( \mathcal{G} \). We will apply well-known facts about Markov chains to establish the results that interest us. A standard reference is [21], from which our terminology and notation is borrowed.
Figure 6.2: An embedding of the shortest path distances for the $\epsilon$-neighborhood graph displayed in Figure 1.1(b). This construction exemplifies the Isomap procedure for manifold learning.

The basic idea that underlies our construction of a random walk on $G$ is that the transition probabilities should reflect the similarities of vertex pairs, so that the random walk has a tendency to step to a vertex that is similar to the current vertex. Thus, we desire $p_{ij} > p_{ik}$ if $\gamma_{ij} > \gamma_{ik}$. If the edge weights are the similarities, then we can proceed by proportionalizing the relevant edge weights.

Assume that $\mathcal{G}$ is undirected, with symmetric weight matrix $W = [w_{ij}]$. Let $T = [t] = \text{diag}(We)$, so that $t_{ii} = \sum_{j=1}^{n} w_{ij}$ is the total weight of all edges originating from vertex $i$. We construct a random walk on $\mathcal{G}$ by assigning transition probabilities $p_{ij} = w_{ij}/t_{ii}$. In matrix form, the transition probabilities are

$$P = T^{-1}W = T^{-1}(T - L) = I - T^{-1}L,$$

where $L = T - W$ is the combinatorial Laplacian matrix of $\mathcal{G}$. Note that $P$ may not be symmetric.

Following [21], the stationary distribution of the random walk assigns probability $\pi_i = t_{ii}/\bar{w}$ to vertex $i$. Let $D = \bar{w}T^{-1}$, where $\bar{w} = e^tWe = e^tTe$ is the volume of $\mathcal{G}$. The fundamental matrix of Kemeny and Snell is

$$Z = (I - P + e^{-t})^{-1} = (I - I + T^{-1}L + e^{-t})^{-1} = (T^{-1}L + e^{-t})^{-1}.$$

It is easily checked that $Ze = e$; hence, $ZD\pi = Ze = e$. Furthermore,

$$ZD = (T^{-1}L + e^{-t})^{-1} \cdot \bar{w} = \bar{w} \left[T \left(T^{-1}L + e^{-t}\right)\right]^{-1} = \bar{w} \left[L + \bar{w}e^{-t}\right]^{-1} = \left[L/\bar{w} + \pi e^{-t}\right]^{-1}.$$
6.4 Combinatorial Laplacian Eigenmaps

6.4.1 Expected Commute Time

Suppose that we start a random walk at vertex $i$ and count the number of steps that are taken to visit vertex $j$ for the first time. This quantity is a random variable called the first passage time from $i$ to $j$. Let $m_{ij}$ denote the expected value of this random variable. In general, $m_{ij}$ need not equal $m_{ji}$, but the sum $c_{ij} = m_{ij} + m_{ji}$, the expected commute time (ECT) between vertices $i$ and $j$, is an intuitively appealing measure of their dissimilarity. Thus, by constructing a random walk on $G$, we obtain a complete matrix of dissimilarities from a partial set of similarities.

The matrix of expected first passage times can be expressed as

$$M = \left( I - Z + ee^t Z_{dg} \right) D$$

and the matrix of expected commute times as

$$C = (M - M_{dg}) + (M - M_{dg})^t.$$

Notice that

$$M_{dg} = \left( I - Z + ee^t Z_{dg} \right)_{dg} D = \left( I - Z_{dg} + (ee^t Z_{dg})_{dg} \right) D$$

$$= \left( I - Z_{dg} + (ee^t)_{dg} Z_{dg} \right) D = (I - Z_{dg} + IZ_{dg}) D = D,$$

so

$$M - M_{dg} = \left( I - Z + ee^t Z_{dg} \right) D - D = -ZD + ee^t Z_{dg} D = -ZD + ee^t (ZD)_{dg}$$

and, because $ZD$ is symmetric,

$$C = ee^t (ZD)_{dg} - 2ZD + (ZD)_{dg} ee^t = \kappa(ZD).$$

While ECT is an intuitively appealing measure of dissimilarity, the equation $C = \kappa(ZD)$ suggests that it might be more appropriate to regard ECT as a measure of squared dissimilarity. In fact, we will demonstrate that $C = [c_{ij}] = [\delta_{ij}^2]$ is EDM-2 and that the embedding of $\Delta = [\delta_{ij}]$ by CMDS can be obtained directly from the combinatorial Laplacian matrix of $G$. This embedding is the standard combinatorial Laplacian eigenmap, although we will subsequently observe that certain properties of ECT may favor interpreting the $c_{ij}$ as distances rather than squared distances.

6.4.2 Expected Commute Times are EDM-2

Let $L$ denote the combinatorial Laplacian matrix of $G$. Because $G$ is connected, $L$ has $n - 1$ strictly positive eigenvalues and one zero eigenvalue. Let $0 < \sigma_1^2 \leq \cdots \leq \sigma_{n-1}^2$ denote the strictly positive eigenvalues of $L$ and let $v_1, \ldots, v_{n-1}$ denote corresponding orthonormal eigenvectors. Then

$$L = \sum_{i=1}^{n-1} \sigma_i^2 v_i v_i^t.$$
and the pseudoinverse of $L$ is

$$L^\dagger = \sum_{i=1}^{n-1} \frac{1}{\sigma_i^2} v_i v_i^t.$$  

(6.1)

Notice that $L$ and $L^\dagger$ have the same eigenvectors, but that their strictly positive eigenvalues are reciprocal. Hence, the eigenvectors that correspond to the smallest positive eigenvalues of $L$ correspond to the largest positive eigenvalues of $L^\dagger$. Both $L$ and $L^\dagger$ are symmetric and positive semidefinite; hence, $\kappa(L^\dagger)$ is EDM-2.

### 6.4.3 Embedding Expected Commute Times

If we interpret ECT as a squared distance on $G$, then kernel PCA constructs a Euclidean representation of $G$ by factoring

$$B = \tau(C) = -\frac{1}{2} P \kappa(ZD) P = PZDP = P \left( L/\tilde{w} + \pi\pi^t \right)^{-1} P.$$

We claim that $B = \tilde{w} L^\dagger = (L/\tilde{w})^\dagger$.

Let $\tilde{L} = L/\tilde{w}$. Because $\tilde{L} e = (L e)/\tilde{w} = 0$, $\tilde{L} P = \tilde{L}$. Also let $A = ZD$ and recall that $A e = e$. It then follows that

$$\tilde{L} B = \tilde{L} P A P = \tilde{L} A P = \left( A^{-1} - \pi\pi^t \right) A P = \left( I - \pi\pi^t A \right) P = \left( I - \pi e^t \right) \left( I - \frac{ee^t}{n} \right) = I - \frac{ee^t}{n}.$$

Because $\tilde{L} B$ is symmetric,

$$\tilde{L} B = (\tilde{L} B)^t = B^t \tilde{L}^t = B \tilde{L}.$$

Furthermore,

$$\tilde{L} B \tilde{L} = \left( I - \frac{ee^t}{n} \right) \tilde{L} = \tilde{L} \quad \text{and} \quad B \tilde{L} B = B \left( I - \frac{ee^t}{n} \right) = B.$$

Thus, the matrices $\tilde{L}$ and $B$ are pseudoinverse and $B = \tilde{L}^\dagger = \tilde{w} L^\dagger$ as claimed. These calculations allow us to construct combinatorial Laplacian eigenmaps directly from $L$. Typically one dispenses with the factor $\tilde{w}$, as in Figure 6.3.

### 6.4.4 Comparison to Shortest Path Distance

In general, combinatorial Laplacian eigenmaps are constructed when edge weights are similarities, whereas shortest path distances are embedded when edge weights are dissimilarities. In the special case of unweighted graphs, $w_{ij} = 1$ can be interpreted as a measure of either similarity or dissimilarity, in which case comparison is possible.

**Example 6.3** Again consider the $n = 201$ points in $\mathbb{R}^2$ displayed in Figure 1.1(a) and let $G$ denote the corresponding $\epsilon$-neighborhood graph displayed in Figure 1.1(b). In contrast to Example 6.2, assign unit edge weights $w_{ij} = 1$.

First, interpret the $w_{ij}$ as dissimilarities and compute $\Delta = [\delta_{ij}]$, the matrix of shortest path distances between the vertices of $G$. Use CMDS to embed $\Delta$ in $\mathbb{R}^2$ and examine the
Let $G$ denote a connected graph with $n$ vertices and symmetric edge weights $w_{ij} = w_{ji}$. Fix $d$, the dimension of the desired Euclidean representation of $G$.

1. Let $L$ denote the combinatorial Laplacian matrix of $G$. Let $\lambda_0 = 0 < \sigma_1^2 \leq \cdots \leq \sigma_{n-1}^2$ denote the eigenvalues of $L$ and let $v_0, v_1, \ldots, v_{n-1}$ denote corresponding orthonormal eigenvectors. Compute the $d$ smallest strictly positive eigenvalues and their corresponding eigenvectors.

2. Construct the configuration matrix

$$X = \begin{bmatrix} \frac{v_1}{\sigma_1} & \cdots & \frac{v_d}{\sigma_d} \end{bmatrix} = \begin{bmatrix} x_1^t \\ \vdots \\ x_n^t \end{bmatrix}.$$  

Figure 6.3: Constructing a $d$-dimensional principal component representation of a graph by a combinatorial Laplacian eigenmap.

The sum of the positive eigenvalues (36396.36) dwarfs the sum of the negative eigenvalues (−372.1549), indicating that the shortest path distances are nearly Euclidean. The first principal component accounts for 97.6% of the total Euclidean variation; the first two account for 98.4%. The resulting configuration is displayed in Figure 6.4(a).

Second, interpret the $w_{ij}$ as similarities and construct a combinatorial Laplacian eigenmap in $\mathbb{R}^2$. This procedure is equivalent to computing $\Delta_2 = C/\tilde{w}$ and embedding $\Delta$ by CMDS. Examine the eigenvalues of $B = \tau(\Delta_2)$. Because $\Delta_2$ is EDM-2, there are no negative eigenvalues; however, the first two principal components account for just 72.8% of the total variation. The resulting configuration is displayed in Figure 6.4(b).

### 6.4.5 Resistance Distance

Constructing a combinatorial Laplacian eigenmap from $L$ instead of $L/\tilde{w}$ is equivalent to dividing expected commute time by graph volume. This quantity is sometimes called resistance distance.$^2$ Combinatorial Laplacian eigenmaps treat resistance distances as squared Euclidean distances; however, it is also possible to treat the $r_{ij} = c_{ij}/\tilde{w}$ as distances.

**Theorem 6.1** Let $G$ be a connected directed graph with resistance distances $r_{ij}$. Then

1. $r_{ij} \geq 0$, with equality iff $i = j$;

---

$^2$The phrase comes from the theory of electrical circuits, in which context vertices are terminals, edges are resistors, edge weights are conductances, and $r_{ij} = c_{ij}/\tilde{w}$ is the potential difference between vertices $i$ and $j$ when a unit current source is applied. The discovery that potential differences are EDM-2 appears to predate the discovery that expected commute times are EDM-2.
Figure 6.4: Two embeddings of the $\epsilon$-neighborhood graph displayed in Figure 1.1(b): (a) CMDS of shortest path distances derived from unit edge weights; (b) Combinatorial Laplacian eigenmap derived from unit edge weights.

2. $r_{ij} = r_{ji}$; and

3. $r_{ik} \leq r_{ij} + r_{jk}$.

We know that $R = [r_{ij}]$ is EDM-2. The following example demonstrates that $R$ may not be EDM-1.

### 6.5 Diffusion Maps

### 6.6 Normalized Laplacian Eigenmaps

### 6.7 Tang’s Equivalence Theorem

### 6.8 Exercises

1. Embedding the partial dissimilarity matrix in Example 6.1 requires care, as there are nonglobal minimizers of $\sigma_0$ that must be avoided.

   (a) What weight matrix $U_0$ corresponds to the objective function $\sigma_0$?
   (b) Generate an initial configuration matrix $X_0$ by drawing each entry from $\text{Normal}(0, 1)$:

       ```r
       > X0 <- matrix(rnorm(12), ncol=2)
       ```

   Use the algorithm in Figure 5.2 with $k=200$ iterations to construct a configuration matrix $X_{200}$. Repeat 100 times, each time computing $\sigma_0(X_{200})$. Count the number of times that $\sigma_0(X_{200}) > 0.0001$. 

(c) Embed vertices 2, 3, 4, 5 by CMDS, then assign coordinates to vertices 1, 6 by drawing from Normal(0, 1), thereby generating $X_0$. Again use the algorithm in Figure 5.2 with k=200 iterations to construct a configuration matrix $X_{200}$. Repeat 100 times, each time computing $\sigma_0(X_{200})$. Count the number of times that $\sigma_0(X_{200}) > 0.0001$. Does this initialization strategy improve the chance of finding a global minimizer of $\sigma_0(X_{200})$? If not, then propose a better strategy.

2. Another way to measure the pairwise similarities of the $n = 201$ points in $\Re^2$ displayed in Figure 1.1(a) is to compute a heat kernel. One can then construct a 2-dimensional combinatorial Laplacian eigenmap from the completely connected graph with edge weights equal to the similarities.

For some $h > 0$, let

$$w_{ij} = \gamma_{ij} = \exp \left( -hd_{ij}^2 \right),$$

where $d_{ij}$ is the Euclidean distance between points $i$ and $j$. The choice of $h$ is a model selection problem that is often left to the user. Experiment with different choices of $h$, constructing combinatorial Laplacian eigenmaps corresponding to each choice. What choice of $h$ would you recommend? What happens if $h$ is too small? What happens if $h$ is too large?
Chapter 7

Manifold Learning

As described in Chapter 1, multivariate data are often represented as points in an ambient feature space, e.g., \( x_1, \ldots, x_n \in \mathbb{R}^q \). By \textit{dimension reduction}, we mean the representation of \( x_1, \ldots, x_n \) as \( y_1, \ldots, y_n \in \mathbb{R}^d \) for \( d < q \). In Chapter 3 we studied one popular technique (PCA) for dimension reduction.

PCA constructs \( y_1, \ldots, y_n \) by projecting \( x_1, \ldots, x_n \) into a \( d \)-dimensional hyperplane, a classic example of a \textit{linear dimension reduction} technique. Sometimes, however, one can obtain a more parsimonious representation of the data by performing \textit{nonlinear dimension reduction}. For example, suppose that we sample points in \( \mathbb{R}^2 \) along the sine wave \((\alpha, \sin \alpha)\). By straightening the sine wave we obtain a perfect 1-dimensional representation of the data. Notice, however, that this procedure is nonlinear: any projection of the data into a straight line will distort the arc length distances between the points, i.e., the distances measured along the trajectory of the sine wave.

A sine wave is an example of a 1-dimensional manifold, i.e., its local structure resembles \( \mathbb{R} \). The present chapter describes several techniques for nonlinear dimension reduction, each motivated by the conceit that \( x_1, \ldots, x_n \) lie on (or near) a low-dimensional manifold in \( \mathbb{R}^q \).

Do actual multivariate data lie (approximately) on low-dimensional manifolds? Describing “the neglected case of nonlinear data structures,” Shepard and Carroll [35] argued that

“there may well be strong nonlinear relations among the variables. If so, the objects will not scatter in all directions according, say, to some ellipsoidal distribution in the multivariate space. Instead, they will tend to fall on some manifold, of lower intrinsic dimensionality, that may nevertheless curve and twist through the space in such a way as to give the superficial \textit{appearance} of filling an ellipsoidal volume.”

More recently, Roweis and Saul [31] argued that “Coherent structure in the world leads to strong correlations between inputs..., generating observations that lie on or close to a smooth low-dimensional manifold.” Manifold learning is concerned with such situations.

7.1 Manifolds

7.1.1 Differentiable Manifolds

The following definition appears in [28].
**Definition 7.1** A set \( M \subset \mathbb{R}^q \) is called a smooth manifold of dimension \( p \) iff each \( m \in M \) has a neighborhood that is diffeomorphic to an open subset of \( \mathbb{R}^p \).

Several elements of this definition require elaboration:

- In this context, a *neighborhood* of \( m \) is the intersection of \( M \) and an open set \( W \subset \mathbb{R}^q \).
- The sets \( W \cap M \) and \( U \subset \mathbb{R}^p \) are diffeomorphic if there is a one-to-one function \( g : U \to W \cap M \) such that both \( g \) and \( g^{-1} \) are smooth. The function \( g \) is a *parametrization* of \( W \cap M \), whereas the function \( g^{-1} \) induces a system of *coordinates* on \( W \cap M \).
- A function is \( C^r \) if its derivatives of order \( r \) are continuous. We may understand *smooth* to specify a specific order of differentiability (e.g., \( r = 1 \) in [28]), or in the somewhat more vague sense of “as many derivatives as the situation requires.” In Definition 7.1, the smoothness of \( M \) is determined by the smoothness of \( g \).
- The dimension \( p \) is fixed, i.e., it may not vary with \( m \).

Here are some elementary examples of low-dimensional manifolds.

**Example 7.1** The spiral displayed in Figure 1.1(a) is a 1-dimensional manifold embedded in a 2-dimensional ambient space. The \( n = 201 \) points in \( \mathbb{R}^2 \) lie near, but not on the manifold. As demonstrated in Example 6.2, these data can be embedded in \( \mathbb{R} \) with small sacrifice, a nonlinear reduction in dimensionality.

**Example 7.2** According to Wikipedia, a Swiss roll is a rolled cake spread with jelly (or jam, whipped cream, icing, etc.). Its spiral shape suggests an obvious extension of Example 7.1. Suppose that \( \sigma : [a,b] \to \mathbb{R}^2 \) parametrizes a spiral. Then \( \tau : [a,b] \times [c,d] \to \mathbb{R}^3 \) defined by \( \tau(s,t) = (\sigma(s), t) \) parametrizes the mathematical abstraction of a Swiss roll, a 2-dimensional manifold embedded in a 3-dimensional ambient space.

Swiss rolls have played an outsized role in the brief history of manifold learning. Figure 3 in [38] used 1000 points on a Swiss roll to demonstrate Isomap, Figure 1 in [31] used points on a Swiss roll to demonstrate LLE, and numerous other researchers have followed suit. In Section 7.4 we will suggest a reason why Swiss rolls pervade the manifold learning literature.

**Example 7.3** A great circle divides the sphere in which it resides into two opposing halves, each of which is a hemisphere. For example, the unit sphere in \( \mathbb{R}^3 \) is

\[
S^2 = \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 : x_1^2 + x_2^2 + x_3^2 = 1 \right\}
\]

and the northern hemisphere of \( S^2 \) is the set of points

\[
\left\{ (x_1, x_2, x_3) \in S^2 : x_3 \geq 0 \right\},
\]

another 2-dimensional manifold embedded in a 3-dimensional ambient space.
Given: feature vectors \( x_1, \ldots, x_n \in \mathbb{R}^q \) and a target dimension \( d \).

1. Construct an \( \epsilon \)-neighborhood or \( K \)-nearest-neighbor graph of the observed feature vectors. Weight edge \( j \leftrightarrow k \) of the graph by \( \|x_j - x_k\| \).

2. Compute the dissimilarity matrix \( \Delta = [\delta_{jk}] \), where \( \delta_{jk} \) is the shortest path distance between vertices \( j \) and \( k \). The key idea that underlies Isomap is that shortest path distances on a locally connected graph approximate geodesic distances on an underlying manifold.

3. Embed \( \Delta \) in \( \mathbb{R}^d \). Traditionally, Isomap embeds by classical multi-dimensional scaling (CMDS); however, if one’s goal is to approximate shortest path distance with Euclidean distance, then one might prefer to minimize the raw stress criterion.

Figure 7.1: Isomap, the manifold learning procedure proposed in [38].

### 7.1.2 Riemannian Manifolds

### 7.1.3 Manifold Learning

Techniques for manifold learning attempt to localize structure in the ambient feature space, then exploit properties of Euclidean space to construct a low-dimensional representation of the manifold. Consider, for example, the motivations for two seminal manifold learning techniques.

- Isomap “seeks to preserve the intrinsic geometry of the data, as captured in the geodesic manifold distances between all pairs of data points.” [38]
- The premise of Locally Linear Embedding (LLE) is that the same weights that locally reconstruct \( x_i \) from its neighbors in the feature space should also reconstruct \( y_i \) from its neighbors in the representation space. [31]

### 7.2 Proximity Analysis of Data Profiles

### 7.3 Isomap

The Isomap procedure for manifold learning is the concatenation of several techniques discussed in previous chapters. Indeed, the construction of the configuration displayed in Figure 6.2 from the data displayed in Figure 1.1(a) is a prototypical example of Isomap.

A generic recipe for Isomap is summarized in Figure 7.1. The key idea that underlies Isomap is that shortest path distances on a locally connected graph approximate geodesic
distances on an underlying manifold. For that reason, it seems natural to embed by minimizing an error criterion such as the raw stress criterion that measures how well the embedded Euclidean interpoint distances approximate the shortest path distances. In fact, the authors’ elected to embed the shortest path distances by CMDS. Because the originality of Isomap lies in its approximation of geodesic distances with shortest path distances, our view is that any embedding of shortest path distances is appropriately described as Isomap.\footnote{Some have described Isomap as an \textit{extension} of CMDS, but the embedding step in Isomap is routine. We regard Isomap as an ingenious \textit{application} of CMDS—or of some other embedding technique.}

Isomap poses two model selection problems for the user: the choice of the localization parameter ($\epsilon$ or $K$), and the choice of the target dimension ($d$).

Dimensionality can be reduced in various ways, and competing goals may collide. Here is an intriguing paragraph from [38], in which the authors struggle to identify and reconcile two conceptually distinct interpretations of Isomap:

“Just as PCA and MDS are guaranteed, given sufficient data, to recover the true structure of linear manifolds, Isomap is guaranteed asymptotically to recover the true dimensionality and geometric structure of a strictly larger class of nonlinear manifolds. Like the Swiss roll, these are manifolds whose intrinsic geometry is that of a convex region of Euclidean space, but whose ambient geometry in the high-dimensional input space may be highly folded, twisted, or curved. For non-Euclidean manifolds, such as a hemisphere or the surface of a doughnut, Isomap still produces a globally optimal low-dimensional Euclidean representation, as measured by Eq. 1.”

The technical details of this paragraph will be explicated in Sections 7.4 and 7.5.

### 7.4 Parametrization Recovery

The concept of parametrization recovery appears in [38], in the claim that “Just as PCA and MDS are guaranteed, given sufficient data, to recover the true structure of linear manifolds, Isomap is guaranteed asymptotically to recover the true dimensionality and geometric structure of a strictly larger class of nonlinear manifolds.” The Parametrization Recovery Problem was formally stated by Donoho and Grimes [11], who proposed the technique of Hessian eigenmaps for its solution. Isomap can (in theory) recover manifolds that are convex and globally isometric to Euclidean space. Donoho and Grimes relaxed both conditions: Hessian eigenmaps can (in theory) recover manifolds that are connected (not necessarily convex) and locally (not necessarily globally) isometric to Euclidean space. They introduced a quadratic form, $\mathcal{H}$, defined on $C^2$ (continuously twice differentiable) functionals on a manifold $M = \psi(\Theta)$, where $\Theta$ is an open connected subset of $\mathbb{R}^p$ and $\psi$ is a locally isometric embedding of $\Theta$ into $\mathbb{R}^q$. Their key result states that “the original isometric coordinates $\theta$ can be recovered, up to a rigid motion, by identifying a suitable basis for the null space of $\mathcal{H}$.” Hessian eigenmaps are constructed from discrete approximations of $\mathcal{H}$.

An isometry is a mapping between two Riemannian manifolds that preserves lengths of curves. The function $\psi : \Theta \to M$ is a (global) isometry between $\Theta$ and $M$ iff the length of every curve $\alpha : [0, 1] \to \Theta$ equals the length of the corresponding curve $\beta : [0, 1] \to M$ defined by $\beta(t) = \psi(\alpha(t))$. It is a local isometry iff, for every $\theta \in \Theta$, there is a neighborhood $N_\theta$ for which $\psi$ is an isometry between $N_\theta$ and $\psi(N_\theta)$. It is difficult to imagine that $\Theta$ could...
be recovered from $\mathcal{M}$ if $\psi$ is not a local isometry, so assuming that $\psi$ is a local isometry would seem to be as weak an assumption as possible. And yet, even this assumption turns out to be severely limiting.

Following [25], a Riemannian manifold is flat iff it is locally isometric to an open subset of Euclidean space. All is well if $d = 1$, for every 1-dimensional Riemannian manifold is flat. Thus, spirals can be straightened and their parametrizations recovered.

The case of $d = 2$ and $q = 3$ is the subject of classical differential geometry. A 2-dimensional Riemannian manifold embedded in $\mathbb{R}^3$ is called a surface. The Gaussian curvature of a surface $\mathcal{M}$ at $m$ is the product of the principal curvatures at $m$: $K(m) = \kappa_1(m)\kappa_2(m)$. If $K(m) = 0$, then there is an arc in $\mathcal{M}$ through $m$ that is a straight line in $\mathbb{R}^3$. A Swiss roll has constant zero curvature (it curves in one principal direction and not in the other), whereas a hemisphere has constant positive curvature (it curves in both principal directions).

Gauss’s celebrated Theorema Egregium (1827) states that Gaussian curvature is invariant under local isometry. Hence, if $\mathcal{M}$ is locally isometric to some $\Theta \subset \mathbb{R}^2$, then $\mathcal{M}$ must have constant zero Gaussian curvature. Thus, parametrization recovery is possible for Swiss rolls, but not for hemispheres. More dramatically: the only surfaces for which parametrization recovery is possible are Swiss rolls, surfaces that only curve in one principal direction.

In the general case, “A Riemannian manifold is flat if and only if its curvature tensor vanishes identically.” [25, Theorem 7.3] The curvature tensor at $m \in \mathcal{M}$ is completely determined by the sectional curvatures at $m$, i.e., the Gaussian curvatures at $m$ of the 2-dimensional submanifolds at $m$ that are swept out by geodesics whose initial tangent vectors lie in a 2-dimensional subspace of the tangent space of $\mathcal{M}$ at $m$. See [25, Chapter 8] for details. It follows that $\mathcal{M}$ is flat iff each sectional curvature at every $m \in \mathcal{M}$ is zero. But this means that, at any point in a flat manifold, there can be at most one principal direction in which the manifold curves. Thus, there are no manifolds with curvature more complicated than a Swiss roll for which parametrization recovery is possible. Small wonder that Swiss rolls appear so frequently in the manifold learning literature!

### 7.5 Geodesic Representation

The concept of geodesic representation also appears in [38], in the claim that “For non-Euclidean manifolds, such as a hemisphere or the surface of a doughnut, Isomap still produces a globally optimal low-dimensional Euclidean representation...” In the context of this claim, “optimal” simply means that an error criterion is minimized in the embedding step of Isomap (see Figure 7.1). Whereas we find the raw stress criterion a more natural way to approximate shortest path distances with Euclidean distances, Tenenbaum et al. preferred CMDS, which is “globally optimal” in the sense of Theorem 4.2.

Whatever the error criterion, the clear intent of Isomap is to construct a Euclidean representation of the geodesic structure of a Riemannian manifold $\mathcal{M}$. In certain special cases, when the geodesic structure of $\mathcal{M}$ is Euclidean, that representation may recover a parametrization of $\mathcal{M}$. If $\mathcal{M}$ is non-Euclidean, then one should expect distortion of its features when representing it as a Euclidean entity. Such distortion is not a failing; rather, it is precisely what is needed in order to see $\mathcal{M}$ in a way that respects its non-Euclidean structure.
7.5.1 Motivating Example

Consider (1) the rectangle $R \subset \mathbb{R}^2$ with vertices at $(\pm 0.05, 0.05)$ and $(\pm 0.05, 0.95)$, and (2) the circle $S \subset \mathbb{R}^2$, centered at $(0, 0)$ and of radius $1/\pi$. The perimeter of $R$ has a total length of $2$; similarly, the circumference of $S$ is $2$.

For $x, y \in R$, let $\delta_R(x, y)$ denote the length of the shortest arc in $R$ that connects $x$ and $y$. Likewise, for $x, y \in S$, let $\delta_S(x, y)$ denote the length of the shortest arc in $S$ that connects $x$ and $y$.

Let $n = 200$. Let $x_1, \ldots, x_n \in R$ be equally spaced with respect to $\delta_R$ and let $y_1, \ldots, y_n \in S$ be equally spaced with respect to $\delta_S$. (For example, place $x_1$ at $(0.05, 0.05)$ and place $x_2, \ldots, x_n$ counterclockwise at increments of $0.01$. Place $y_1$ at $(1/\pi, 0)$ and place $y_2, \ldots, y_n$ counterclockwise at increments of $0.01$.) Let $X$ and $Y$ denote the corresponding configuration matrices and define the following dissimilarity matrices:

\[
D(X) = [||x_i - x_j||] \quad \text{and} \quad D(Y) = [||y_i - y_j||];
\]

\[
\Delta(X) = [\delta_R(x_i, x_j)] \quad \text{and} \quad \Delta(Y) = [\delta_S(y_i, y_j)].
\]

By definition, both $D(X)$ and $D(Y)$ are EDM-1. Hence, a configuration that is isometric to $X$ can be recovered from $D(X)$ and a configuration that is isometric to $y$ can be recovered from $D(y)$. For example, CMDS recovers a configuration whose centroid lies at the origin and whose coordinate axes are the principal components of the configuration. Because $X$ and $Y$ are not isometric, $D(X) \neq D(Y)$ and the recovered configurations are not isometric.

By construction, $\Delta(X) = \Delta(Y) = \Delta$. The metric structures of $R$ and $S$ defined by $\delta_R$ and $\delta_S$ are identical: it is not possible to recover from $\Delta$ the distinction between the rectangular shape of $R$ and the circular shape of $S$. Furthermore, $\Delta$ is not EDM-1. The $n \times n$ matrix $\tau(\Delta_2)$ has 100 positive eigenvalues, one zero eigenvalue, and 99 negative eigenvalues. The negative eigenvalues correspond to the non-Euclidean portion of $\Delta$ and have a total variation of 16.665. The positive eigenvalues correspond to the Euclidean portion of $\Delta$ and have a total variation of 50. Furthermore, the first two principal components of this 100-dimensional Euclidean configuration explain $40.53181/50 \approx 81\%$ of its total variation. This 2-dimensional configuration of points is displayed in Figure 7.2.

One’s initial impression of Figure 7.2 is that CMDS has recovered $Y$ but not $X$. This impression is misleading, because Figure 7.2 was constructed solely from $\Delta$. The pairwise arc distances are the same for $Y$ and $X$; hence, the representation in Figure 7.2 is equally valid for $Y$ and for $X$. The key to understanding Figure 7.2 is appreciating that it is a Euclidean approximation of a non-Euclidean structure.

First, the circle on which the points in Figure 7.2 lie is not $S$. Its radius is approximately $0.45$, not $1/\pi \approx 0.32$. Second, the 2-dimensional configuration in Figure 7.2 is only an approximation, the projection of a 100-dimensional configuration onto its first two principal components. Third, even the 100-dimensional configuration is only an approximation, specifically the best least squares approximation of $\tau(\Delta_2)$ by centered Euclidean inner products.

Properly interpreted, it makes perfect sense that equally spaced points on any closed curve would lead CMDS to construct a circular configuration of points from the pairwise arc distances. If the matrix of equally spaced arc distances is $\Delta$, then the corresponding matrix of (fallible) centered inner products $B = \tau(\Delta_2)$ has constant diagonal entries of $b_i^2$, suggesting that all points should be placed on a sphere of radius $b$. Moreover, the matrix of (fallible) angles is $A = [\arccos(b_{ij}/b^2)]$, and the angles between each pair of consecutive
points have a constant value $a$. Viewed in this way, a circular configuration of points is the obvious 2-dimensional embedding of $\Delta$.

Although embedding $\Delta$ in $\mathbb{R}^2$ (or in any $\mathbb{R}^d$) does not recover $X$ or $Y$, the representation of $\Delta$ in Figure 7.2 is of evident value. However, it must be emphasized that Figure 7.2 represents the metric structure of $R$ and $S$ by 2-dimensional Euclidean structure. The arc distances measured by $\delta_R$ and $\delta_S$ are approximated by Euclidean distances in Figure 7.2. Although the points in Figure 7.2 lie on a circle, it is the chordal distances between these points that approximate the arc distances in $\Delta$.

7.5.2 Representing Swiss Rolls

7.6 Exercises

1. Referring to Section 7.5.1, determine $b^2$ and $a$ for $\Delta = \Delta(X) = \Delta(Y)$. 
Figure 7.2: CMDS embedding of 200 equally spaced points on a closed curve of length 2. The matrix of pairwise arc distances, hence the embedding, does not depend on the shape of the curve.
Part III

Clustering
Figure 8.0: Unsupervised versus supervised learning. These $n = 18$ feature vectors in $\mathbb{R}^2$ have been labelled blue or red. Unsupervised: absent the labels, the horizontal axis explains more total variation and better separates the data into two clusters with good internal cohesion and external separation. Supervised: the vertical axis better discriminates red objects from blue objects.

Chapter 1 introduced three fundamental types of data: measurements on objects, proximities between pairs of objects, and graphs that connect (some) pairs of objects. The chapters in Part IV describe several methods that use such data to construct rules for assigning labels to objects. These methods require data that have been suitably labelled, e.g., as “dogs” or “cats”. Such methods are variously described as supervised learning, classification, and statistical pattern recognition. In contrast, the phrase unsupervised learning refers to methods that attempt to discern structure in unlabelled objects. For example, each of the various dimension reduction techniques described thus far has been unsupervised. Figure 8.0 illustrates the distinction between unsupervised and supervised learning.

A prominent example of unsupervised learning is clustering (also cluster analysis or numerical taxonomy). The fundamental goal of hard clustering is to partition a set of $N$ objects into clusters. Hard clustering assigns each object to exactly one of $k$ possible clusters, but other types of clustering are possible, e.g., allowing an object to belong to multiple clusters, or identifying a single cluster that contains only certain objects.

The chapters in Part III describe several methods for identifying clusters. But just what is a cluster? Here are three possible answers:

1. Clusters are subsets of objects that display “internal cohesion” and “external separation”. These widely used phrases convey an excellent sense of what one seeks, but only to the extent that “one knows it when one sees it.” Alone, they provide little guidance on how to develop useful algorithms for identifying clusters.

2. Variation within clusters is small relative to variation between clusters. This answer is somewhat more useful, assuming that one is prepared to specify the type of variation. For example, measuring variation by squared Euclidean distance leads to the $k$-means algorithms described in Chapter 9.

3. Clusters are what clustering algorithms produce. Some clustering algorithms attempt to optimize well-defined objectives that clearly operationalize notions of internal cohesion and/or external separation. Many clustering algorithms are heuristic, i.e., they do something plausible, implicitly defining a conception of what is desired.
Because each clustering algorithm defines its own conception of clustering, it is difficult—perhaps impossible—to compare clustering algorithms. Each algorithm does what it was designed to do. Most clustering algorithms are purely exploratory, and different clustering algorithms explore data differently. If one is prepared to specify a probability model that describes how the data were generated, then one can formulate clustering as a problem in statistical inference, in which setting the direct comparison of different inferential procedures is possible.

Mathematically, a partition of a set is a collection of disjoint subsets whose union is the entire set. In the present context, the set to be partitioned is a finite set of \( N \) objects, the partition is a clustering, and the subsets are the clusters. Clustering methods are obliged to produce clusterings (“it’s what they do”), whether or not the data contain clusters with good internal cohesion and external separation. Automatic clustering is potentially dangerous because most users, after seeing a clustering, tend to reify it. Clusterings should be viewed as suggestive, and users should seek external validation (ideally from domain experts) that confirms that the clusters are truly meaningful. In (good) practice, cluster analysis is an interactive activity in which the user deploys one (or several) clustering algorithms to explore possible structure in the data.

The concepts of internal cohesion and external separation necessitate some measure of object proximity. Clustering methods that operate directly on proximities are appealing, both in their simplicity and in their generality. Hierarchical agglomerative clustering methods such as single linkage, complete linkage, and average linkage are among the earliest, most studied, and most widely used of all clustering methods. Many clustering methods operate on feature vectors in a metric space, typically Euclidean space, and use the metric to measure object proximity. This might involve clustering high-dimensional data observed to lie in \( \mathbb{R}^q \) or clustering a low-dimensional representation of the observed data. Frequently, clustering is preceded by dimension reduction, which functions as a form of regularization that (one hopes) filters noise and makes the clusters easier to discern. Graph-theoretic clustering methods operate on graphs, using edges and edge weights to construct plausible notions of proximity. Chapters 8—11 provide various examples of clustering and the interplay between proximities, features, and graphs.

We conclude our preface to clustering by illustrating that it is not at all obvious how to construct workable definitions of internal cohesion and external separation. Let \( \Delta = [\delta_{rs}] \) denote an \( N \times N \) matrix of pairwise dissimilarities. This matrix contains a total of \( N^2 \) pairwise comparisons of objects and is indexed by the set \( \mathcal{I} = 1:N \times 1:N \).² By definition, a clustering \( \mathcal{C} = \{C_1, \ldots, C_k\} \) partitions the \( N \) objects. It also partitions the \( N^2 \) pairwise comparisons, into within-cluster comparisons for which objects \( r \) and \( s \) lie in the same cluster and between-cluster comparisons for which \( r \) and \( s \) lie in different clusters.

Let \( \mathcal{W} \) index the within-cluster comparisons and let \( \mathcal{B} \) index the between-cluster comparisons. Naively, small values of

\[
W_1 = \sum_{(r,s) \in \mathcal{W}} \delta_{rs} = \sum_{i=1}^{k} \sum_{r,s \in C_i} \delta_{rs}
\]

suggest good internal cohesion and large values of

\[
B_1 = \sum_{(r,s) \in \mathcal{B}} \delta_{rs}
\]

²We might enumerate the possible comparisons in other ways, e.g., by restricting attention to \( \delta_{rs} \) for which \( r < s \), but we prefer the notational simplicity that results from considering all \( \delta_{rs} \).
suggest good external separation. It seems natural to seek partitions with small $W_1$ and large $B_1$; furthermore, because

$$W_1 + B_1 = \sum_{(r,s) \in \mathcal{I}} \delta_{rs} = e^t \Delta e$$

is constant, maximizing $B_1$ is equivalent to minimizing $W_1$ and it suffices to do the latter.

It is impossible to proceed without imposing a crucial restriction. If $k$ is free to vary, then the trivial clustering $C_N$ that places each object in its own cluster will minimize $W_1$. To avoid this pathology, we assume that $k$ has been fixed and that the immediate problem is to search for a clustering $C_k = \{C_1, \ldots, C_k\}$.

The preceding approach seems plausible, but how well does it work? Define objects $x_1, \ldots, x_{12} \in \mathbb{R}$ by $x_r = (r - 1)/10$ for $r = 1, \ldots, 11$ and $x_{12} = 2$. Let $\delta_{rs} = |x_r - x_s|$ and set $k = 2$. An obvious clustering is $C_1 = \{x_1, \ldots, x_{11}\}$ and $C_2 = \{x_{12}\}$, in which case the maximal within-cluster dissimilarity ($\delta_{1,11} = 1$) equals the minimal between-cluster dissimilarity ($\delta_{11,12} = 1$). An absurd clustering is $D_1 = \{x_2, \ldots, x_{11}\}$ and $D_2 = \{x_1, x_{12}\}$, which places the two least similar objects in the same cluster. But $W_1(D) = 37 < 44 = W_1(C)$. The fact that $W_1$ prefers $D$ to $C$ disqualifies this criterion from serious consideration.

What went wrong? Let $n_i = \#(C_i)$ denote the number of objects assigned to cluster $i$. Notice that $\#(\mathcal{W})$, the number of $\delta_{rs}$ that are summed to form $W_1$, equals $\sum_{i=1}^k n_i^2$, which varies according to how many objects are placed in each cluster. This number will be minimized if the clusters are of equal size, suggesting that using $W_1$ as an optimality criterion will bias clustering toward clusters of equal size. For $C$ there are $11^2 + 1^2 = 122$ within-cluster comparisons, whereas for $D$ there are only $10^2 + 2^2 = 104$ within-cluster comparisons. The bias toward clusters of equal size is sufficiently strong that $W_1$ prefers an absurd clustering to an obvious clustering.
Chapter 8

Agglomerative Hierarchical Clustering

Let $C_1, \ldots, C_N$ denote clusterings of $N$ objects with the following properties:

- $C_k$ contains $k$ nonempty clusters; and
- each cluster in $C_k$ is either a cluster in $C_{k+1}$ or the union of two clusters in $C_{k+1}$.

A variety of clustering methods construct such hierarchies of clusterings, one (or more) of which may subsequently be deemed useful. Notice that $C_1$ necessarily places all objects in a single cluster, whereas $C_N$ necessarily places each object in its own cluster. Some hierarchical methods are divisive, i.e., they begin with $C_1$ and construct $C_{k+1}$ from $C_k$ by splitting one of the clusters in $C_k$. Other hierarchical methods are agglomerative, i.e., they begin with $C_N$ and construct $C_k$ from $C_{k+1}$ by merging two of the clusters in $C_{k+1}$. Chapter ?? describes a divisive method. The present chapter surveys several agglomerative methods. These methods are among the oldest, most studied, and most widely used of all clustering methods.

8.1 External Separation

Agglomerative methods proceed by merging clusters. Different methods are defined by different merging criteria, but the heuristic is always the same: merge the pair of clusters that have the least external separation.

Let $\delta_{rs}$ denote the dissimilarity of objects $r$ and $s$. Here are three plausible (and widely used) definitions of external separation:

1. Single Linkage. The external separation of clusters $C_i$ and $C_j$ is the minimum dissimilarity among all pairwise comparisons of one object in $C_i$ and one object in $C_j$, i.e.,

$$ES(C_i, C_j) = \min \{ \delta_{rs} : x_r \in C_i, x_s \in C_j \}.$$  

Single linkage is also known as the nearest-neighbor or minimum method.

2. Complete Linkage. The external separation of clusters $C_i$ and $C_j$ is the maximum dissimilarity among all pairwise comparisons of one object in $C_i$ and one object in $C_j$, i.e.,

$$ES(C_i, C_j) = \max \{ \delta_{rs} : x_r \in C_i, x_s \in C_j \}.$$  

Complete linkage is also known as the farthest-neighbor or maximum method.
3. Average Linkage. The external separation of clusters $C_i$ and $C_j$ is the average dissimilarity among all pairwise comparisons of one object in $C_i$ and one object in $C_j$, i.e.,

$$ES(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x_r \in C_i} \sum_{x_s \in C_j} \delta_{rs},$$

where $n_i = \#(C_i)$ denotes the number of objects in $C_i$.

In fact, each of the above measures can be realized as a special case of a family of recursive formulas for updating cluster dissimilarity. First, for the initial clustering $C_N$, define the external separation of clusters $C_i = \{x_i\}$ and $C_j = \{x_i\}$ to be

$$ES(C_i, C_j) = \delta_{ij}.$$ 

Subsequently, if clusters $C_i$ and $C_j$ are merged, then update the external separations of the clusters by setting

$$ES(C_i \cup C_j, C_\ell) = \alpha_i ES(C_i, C_\ell) + \alpha_j ES(C_j, C_\ell) + \beta ES(C_i, C_j) + \gamma |ES(C_i, C_\ell) - ES(C_j, C_\ell)|.$$

Formula 8.1 was discovered by Lance and Williams [23, 24], who pointed out that a variety of well-known agglomerative methods could be obtained by appropriate specification of the parameters $\alpha_i$, $\alpha_j$, $\beta$, and $\gamma$. For example, setting $\alpha_i = \alpha_j = 1/2$, $\beta = 0$, and $\gamma = -1/2$ specifies single linkage, whereas $\alpha_i = \alpha_j = 1/2$, $\beta = 0$, and $\gamma = 1/2$ specifies complete linkage. See Exercise 8.3.3.

### 8.2 Dendrograms

Let $C_{k+2} = \{C_1, \ldots, C_k, C_{k+1}, C_{k+2}\}$ denote a clustering. For convenience, assume that the clusters have been labelled so that $C_{k+1}$ and $C_{k+2}$ are the pair of clusters with the smallest external separation, so that agglomeration will result in $C_{k+1} = \{C_1, \ldots, C_k, C_{k+1} \cup C_{k+2}\}$.

What can we say about the external separations of the clusters in $C_{k+1}$?

Our interest is in the external separations between $C_{k+1} \cup C_{k+2}$ and $C_\ell$, $\ell = 1, \ldots, k$, as external separations amongst $C_1, \ldots, C_k$ are not changed by merging $C_{k+1}$ and $C_{k+2}$. For complete linkage,

$$ES(C_{k+1} \cup C_{k+2}, C_\ell) = \max \{\delta_{rs} : x_r \in C_{k+1} \cup C_{k+2}, x_s \in C_\ell\}$$

$$= \max \{\max \{\delta_{rs} : x_r \in C_{k+1}, x_s \in C_\ell\} \cup \max \{\delta_{rs} : x_r \in C_{k+2}, x_s \in C_\ell\}\}$$

$$= \max \{ES(C_{k+1}, C_\ell), ES(C_{k+2}, C_\ell)\}$$

$$\geq ES(C_{k+1}, C_{k+2}).$$

For single linkage,

$$ES(C_{k+1} \cup C_{k+2}, C_\ell) = \min \{\delta_{rs} : x_r \in C_{k+1} \cup C_{k+2}, x_s \in C_\ell\}$$

$$= \min \{\min \{\delta_{rs} : x_r \in C_{k+1}, x_s \in C_\ell\} \cup \min \{\delta_{rs} : x_r \in C_{k+2}, x_s \in C_\ell\}\}$$

$$= \min \{ES(C_{k+1}, C_\ell), ES(C_{k+2}, C_\ell)\}$$

$$\geq ES(C_{k+1}, C_{k+2}).$$
In both cases, we conclude that the external separation of the clusters that will be merged to form $C_k$ from $C_{k+1}$ is no less than the external separation of the clusters that were merged to form $C_{k+1}$ from $C_{k+2}$.

Agglomerative methods construct a sequence of cluster agglomerations. If the corresponding sequence of external separations is nondecreasing, then the method is monotonic. We have just demonstrated that complete and single linkage are monotonic methods. In proposing (8.1), Lance and Williams [23] noted that monotonicity can be ensured by setting $\gamma = 0$ and requiring $\alpha_i + \alpha_j + \beta \geq 1$. See Exercise 8.3.5.

### 8.3 Exercises

1. Refer to the example constructed in the preface to Part III. Verify that $W_1(C) = 44$ and $W_1(D) = 37$. For $k = 2$, what clustering of $x_1, \ldots, x_{12}$ minimizes the value of $W_1$?

2. Suppose that $x_1, \ldots, x_N \in \mathbb{R}^p$ and $\delta_{rs} = \|x_r - x_s\|$. In this setting, single linkage measures the external separation of clusters $C_i$ and $C_j$ by

$$
\text{ES}(C_i, C_j) = \min \left\{ \|x_r - x_s\| : x_r \in C_i, x_s \in C_j \right\}.
$$

Does this measure satisfy the triangle inequality? Either (a) prove that $\text{ES}(C_i, C_\ell) \leq \text{ES}(C_i, C_j) + \text{ES}(C_j, C_\ell)$, or (b) construct a counterexample that demonstrates the reverse.

3. Verify the following substitutions into (8.1), the Lance-Williams update formula.

(a) Setting $\alpha_i = \alpha_j = 1/2$, $\beta = 0$, and $\gamma = -1/2$ specifies single linkage.

(b) Setting $\alpha_i = \alpha_j = 1/2$, $\beta = 0$, and $\gamma = 1/2$ specifies complete linkage.

4. Determine the values of $\alpha_i$, $\alpha_j$, $\beta$, and $\gamma$ in (8.1) that specify average linkage.

5. Show that setting $\gamma = 0$ and requiring $\alpha_i + \alpha_j + \beta \geq 1$ in (8.1) ensures that “the string of measures associated with successive hierarchical fusions will be monotonic.” [24] Is average linkage monotonic? Why or why not?

6. Suppose that we modify average linkage, replacing average pairwise dissimilarity between $C_i$ and $C_j$ with median pairwise dissimilarity between $C_i$ and $C_j$. Can the resulting method be represented using (8.1)? Is it monotonic?

7. Use single linkage, complete linkage, and average linkage to explore the structure of the $15 \times 15$ Congressional voting dissimilarity matrix. For each method, identify the clustering that you find most enlightening.

**Hint:** Use the `hclust` function in R.
Chapter 9

K-Means Clustering

A popular method of clustering is based on squared Euclidean distance.

9.1 Euclidean Formulation

Suppose that \(C_1, \ldots, C_k\) partitions \(x_1, \ldots, x_N \in \mathbb{R}^p\) with centroid \(\bar{x}\). Let

\[
T(X) = \sum_{j=1}^{N} \|x_j - \bar{x}\|^2,
\]

let \(n_i = \#(C_i)\) denote the cardinality of \(C_i\), and let \(\bar{x}_i = \bar{x}(C_i)\) denote the centroid of \(C_i\). Then

\[
\sum_{x_j \in C_i} (x_j - \bar{x}_i) = 0,
\]

hence

\[
\sum_{x_j \in C_i} \|x_j - \bar{x}_i + \bar{x}_i - \bar{x}\|^2 = \sum_{x_j \in C_i} \|x_j - \bar{x}_i\|^2 + n_i \|\bar{x}_i - \bar{x}\|^2
\]

and

\[
T(X) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \bar{x}_i + \bar{x}_i - \bar{x}\|^2
\]

\[
= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \bar{x}_i\|^2 + \sum_{i=1}^{k} n_i \|\bar{x}_i - \bar{x}\|^2
\]

\[
= \bar{W}(C_1, \ldots, C_k) + \bar{B}(C_1, \ldots, C_k).
\]

The quantity \(\bar{W}\) will be small if each \(C_i\) is tightly clustered around its centroid and the quantity \(\bar{B}\) will be large if (some of) the cluster centroids are far from the overall centroid. Hence, the general goals of clustering, internal cohesion and external separation, are operationally defined by choosing a partition that minimizes \(\bar{W}\) and maximizes \(\bar{B}\). Because \(\bar{W} + \bar{B}\) is fixed, it suffices to minimize \(\bar{W}\). Notice that the arguments of \(\bar{W}\) are discrete.

The problem of \(k\)-means clustering is usually stated as the problem of finding a partition that minimizes

\[
\bar{W}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \bar{x}_i\|^2.
\]
We prefer an equivalent formulation. Let \( m_i \in \mathbb{R}^p \) denote the representative of \( C_i \), not necessarily its centroid, and define
\[
W(C_1, \ldots, C_k; m_1, \ldots, m_k) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - m_i\|_2^2.
\]

We shall regard \( k \)-means clustering as the problem of minimizing \( W \). Because the optimal representative of \( C_i \) is \( \bar{x}_i \),
\[
\bar{W}(C_1, \ldots, C_k) = W(C_1, \ldots, C_k; \bar{x}_1, \ldots, \bar{x}_k) \leq W(C_1, \ldots, C_k; m_1, \ldots, m_k)
\]
and it suffices to minimize the variable projection functional \( \bar{W} \). Nevertheless, we prefer to regard minimizing \( \bar{W} \) as but one possible way of minimizing \( W \).

When \( N \) is large, the problem of \( k \)-means clustering poses both computational and graphical challenges. There exist a number of algorithms that monotonically decrease \( \bar{W} \) and converge to a locally optimal partition, but algorithms that search for global solutions are overwhelmed by several hundred \( x_i \). When \( N \) is very large (and especially when \( p \) is high), it also may be difficult to display information about the partition, in which case a natural compromise is to display information about the representatives.

### 9.2 Algorithms

The following notation will facilitate subsequent exposition. Given clusters \( C_1, \ldots, C_k \), representatives \( m_1, \ldots, m_k \), and \( x_j \in \{x_1, \ldots, x_N\} \), let \( C(x_j) \) denote the cluster to which \( x_j \) belongs and let \( m(x_j) \) denote the representative of \( C(x_j) \). Let \( m_*(x_j) \) denote any representative that is nearest \( x_j \), i.e.,
\[
\|x_j - m_*(x_j)\|_2 \leq \|x_j - m_i\|_2
\]
for \( i = 1, \ldots, k \), and let \( C_*(x_j) \) denote the cluster whose representative is \( m_*(x_j) \).

#### 9.2.1 Alternating Least Squares

It follows from (9.2) that, for \( C_1, \ldots, C_k \) fixed, one can potentially decrease \( W \) by replacing an arbitrary set of representatives \( m_1, \ldots, m_k \) with the cluster means, \( \bar{x}(C_1), \ldots, \bar{x}(C_k) \).

Analogously, fix representatives \( m_1, \ldots, m_k \) and define clusters
\[
\tilde{C}_i = \tilde{C}(m_i) = \{x_j \in \{x_1, \ldots, x_N\} : m_*(x_j) = m_i\}.
\]

Because the clustering \( \tilde{C}_1, \ldots, \tilde{C}_k \) is optimal for \( m_1, \ldots, m_k \),
\[
\tilde{W}(m_1, \ldots, m_k) = W(\tilde{C}_1, \ldots, \tilde{C}_k; m_1, \ldots, m_k) \leq W(C_1, \ldots, C_k; m_1, \ldots, m_k)
\]
and it follows that one can potentially decrease \( W \) by replacing an arbitrary set of clusters with \( \tilde{C}_1, \ldots, \tilde{C}_k \). Lloyd’s algorithm combines these observations. Starting from any set of \( k \) distinct representatives, it first computes the optimal clusters associated with the representatives, then computes the centroids of those clusters, then computes the optimal clusters associated with those centroids, and so on. Lloyd’s algorithm is summarized in Figure 9.1.
### 9.2.2 Exchange

If \( m_i = \bar{x}_i \) and the clustering is suboptimal, then there is an obvious way to decrease \( \bar{W} \). Denoting the elements of \( C_1 \) by \( \{y_1, \ldots, y_{n_1}\} \) and the elements of \( C_2 \) by \( \{z_1, \ldots, z_{n_2}\} \), suppose that

\[
\|y_{n_1} - m_1\|_2 > \|y_{n_1} - m_2\|_2,
\]

in which case \( y_{n_1} \) is identified with the wrong representative. Let \( C'_1 = \{y_1, \ldots, y_{n_1-1}\} \), let \( C'_2 = \{y_{n_1}, z_1, \ldots, z_{n_2}\} \), and denote the means of \( C'_1 \) and \( C'_2 \) by

\[
m'_1 = \frac{1}{n_1 - 1} \sum_{j=1}^{n_1-1} y_j = \frac{n_1}{n_1 - 1} m_1 - \frac{1}{n_1 - 1} y_{n_1},
\]

\[
m'_2 = \frac{1}{n_2 + 1} \left( y_{n_1} + \sum_{j=1}^{n_2} z_j \right) = \frac{n_2}{n_2 + 1} m_2 + \frac{1}{n_2 + 1} y_{n_1}.
\]

Then

\[
\bar{W}(C_1, C_2) = \sum_{j=1}^{n_1} \|y_j - m_1\|_2^2 + \sum_{j=1}^{n_2} \|z_j - m_2\|_2^2
\]

\[> \sum_{j=1}^{n_1-1} \|y_j - m_1\|_2^2 + \|y_{n_1} - m_2\|_2^2 + \sum_{j=1}^{n_2} \|z_j - m_2\|_2^2 \]

\[\geq \sum_{j=1}^{n_1-1} \|y_j - m'_1\|_2^2 + \|y_{n_1} - m'_2\|_2^2 + \sum_{j=1}^{n_2} \|z_j - m'_2\|_2^2 \]

\[> \sum_{j=1}^{n_1-1} \|y_j - m'_1\|_2^2 + \|y_{n_1} - m'_2\|_2^2 + \sum_{j=1}^{n_2} \|z_j - m'_2\|_2^2 \]

\[= \bar{W}(C'_1, C'_2).\]

These considerations suggest another popular approach to \( k \)-means clustering, the exchange algorithms summarized in Figure 9.2.

**Example 9.1** Fix \( \epsilon \in (0, 0.25) \) and consider the following \( N = 6 \) points in \( \mathbb{R}^2 \): \((-1, \pm 1), (\pm \epsilon, 0), (1, \pm 1)\). We attempt to find (local) minimizers of \( \bar{W} \) using Lloyd’s algorithm and an exchange algorithm, in each case starting from the following initial clustering:

\[
C_1^{(0)} = \{(-1, \pm 1), (\epsilon, 0)\} \quad \text{has centroid } \quad m_1^{(0)} = \left(\frac{-2 + \epsilon}{3}, 0\right);
\]

\[
C_2^{(0)} = \{(\pm \epsilon, 0), (1, \pm 1)\} \quad \text{has centroid } \quad m_2^{(0)} = \left(\frac{2 - \epsilon}{3}, 0\right).
\]
The first iteration of Lloyd’s algorithm replaces the initial clustering with the following clustering:

\[ C^{(1)}_1 = \tilde{C}(m^{(0)}_1) = \{(-1, \pm 1), (-\epsilon, 0)\} \text{ has centroid } m^{(1)}_1 = \left(-\frac{2-\epsilon}{3}, 0\right); \]

\[ C^{(1)}_2 = \tilde{C}(m^{(0)}_2) = \{(\epsilon, 0), (1, \pm 1)\} \text{ has centroid } m^{(1)}_2 = \left(\frac{2+\epsilon}{3}, 0\right). \]

In this clustering, each \( x_j \) belongs to the cluster whose centroid is nearest \( x_j \); hence, each

\[ C^{(2)}_i = \tilde{C}(m^{(1)}_i) = C^{(1)}_i \]

and Lloyd’s algorithm terminates at a clustering with internal cohesion

\[ W_\ell = \sum_{x_j \in C^{(1)}_1} \|x_j - m^{(1)}_1\|^2 + \sum_{x_j \in C^{(1)}_2} \|x_j - m^{(1)}_2\|^2 = 5 + \frac{1}{3} - \frac{8}{3}\epsilon + \frac{4}{3}\epsilon^2. \]

In contrast, consider an exchange algorithm that first chooses \( x = (\epsilon, 0) \). Because \( x \) is nearer \( m^{(0)}_2 \) than \( m^{(0)}_1 \), we move \( x \) from the first to the second cluster, obtaining the following clustering:

\[ C^{(1)}_1 = \{(-1, \pm 1)\} \text{ has centroid } m^{(1)}_1 = (-1, 0); \]

\[ C^{(1)}_2 = \{(\pm \epsilon, 0), (1, \pm 1)\} \text{ has centroid } m^{(1)}_2 = (1/2, 0). \]

In this clustering, each \( x_j \) belongs to the cluster whose centroid is nearest \( x_j \); hence, no further exchanges will occur and we terminate at a clustering with internal cohesion

\[ W_x = \sum_{x_j \in C^{(1)}_1} \|x_j - m^{(1)}_1\|^2 + \sum_{x_j \in C^{(1)}_2} \|x_j - m^{(1)}_2\|^2 = 5 + 2\epsilon^2. \]

Which clustering is better? If \( \epsilon = 1/8 \), then

\[ W_\ell = 5 + \frac{4}{3} \frac{1}{64} < 5 + 2 \frac{1}{64} = W_x; \]

however, if \( \epsilon = 1/9 \), then

\[ W_\ell = 5 + \frac{13}{243} > 5 + \frac{6}{243} = W_x. \]

We conclude that either algorithm can terminate at a nonglobal solution, and that neither algorithm is guaranteed to outperform the other.

Figure 9.2: Exchange algorithms for k-means clustering. Different ways of choosing \( x \) define different exchange algorithms.
9.2.3 Stochastic Gradient Descent

Even if we are only interested in the representatives, the previous algorithms require us to know and update the clustering partition. Especially when $N$ is large, it is natural to inquire if one can dispense with this bit of bookkeeping. Toward that end, consider the variable projection functional $\tilde{W}(m_1, \ldots, m_k)$ defined in (9.3). Just as minimizing $\bar{W}$ is one approach to minimizing $W$, so is minimizing $\tilde{W}$. Notice that, whereas the arguments of $\bar{W}$ are discrete, the arguments of $\tilde{W}$ are continuous. Furthermore, the problem of minimizing $\tilde{W}$ is not a continuous relaxation of the problem of minimizing $\bar{W}$; rather, it is a continuous formulation of the problem of minimizing $W$.

It is not obvious how to vary $m_1, \ldots, m_k$ so as to decrease $\tilde{W}$. Recall, however, that the inequality in (9.1) follows from the well-known fact that, for a fixed partition $C_1, \ldots, C_k$, the objective function $f(m_1, \ldots, m_k) = W(C_1, \ldots, C_k; m_1, \ldots, m_k)$ is minimized by setting each $m_i$ equal to the mean of the $x_j \in C_i$. This fact is easily deduced from the separability of $f$ into $k$ convex summands,

$$f_i(m_i) = \sum_{x_j \in C_i} \|x_j - m_i\|^2,$$

and the stationary equations

$$\nabla f_i(m_i) = \sum_{x_j \in C_i} 2(m_i - x_j) = 0$$

for $i = 1, \ldots, k$. Now, suppose that we eschew knowledge of the solution and try to minimize each $f_i$ by the method of steepest descent. This is an iterative method that replaces the current iterate, $m_i^{(n)}$, with

$$m_i^{(n+1)} = m_i^{(n)} - t_n \nabla f_i(m_i^{(n)})$$

$$= m_i^{(n)} - t_n \sum_{x_j \in C_i} 2(m_i^{(n)} - x_j)$$

$$= [1 - 2t_n \#(C_i)] m_i^{(n)} + [2t_n \#(C_i)] \bar{x}_i$$

$$= (1 - \alpha_n) m_i^{(n)} + \alpha_n \bar{x}_i,$$

where $\#(C_i)$ is the cardinality of $C_i$. In numerical optimization, the step length control parameter $t_n$ (equivalently $\alpha_n$) is usually determined by performing a line search; however, one can also specify certain predetermined sequences, as in stochastic approximation. This may well impress the reader as an absurdly inefficient way to compute a mean (especially as the iteration formula requires knowing the mean!), but let us see where it leads.

To try to decrease $\tilde{W}$, we first set $C_i = \bar{C}_i$. Then, for each $i = 1, \ldots, k$, $m_i - \bar{x}_i$ is a descent direction for $\tilde{W}$. Unfortunately, identifying descent direction $i$ requires knowing $\bar{x}_i$. We attempt to circumvent this difficulty by randomly drawing $x \in \{x_1, \ldots, x_N\}$ and determining $m_*(x)$. The choice of $x$ thus determines which $m_i$ is to be varied. Furthermore, we use $x$ to estimate $\bar{x}_*$, resulting in an iterative algorithm that replaces $m_i^{(n)} = m_i^{(n)}(x)$ with

$$\hat{m}_i^{(n+1)} = (1 - \alpha_n) m_i^{(n)} + \alpha_n x.$$
initialize \( m_1, \ldots, m_k \) and \( \alpha \)

\[
\text{do until termination:} \\
\quad \text{draw } x \sim \text{Uniform}\{x_1, \ldots, x_N\} \\
\quad \text{determine } m_\star(x) \\
\quad \text{replace } m_\star(x) \text{ with } (1 - \alpha) m_\star(x) + \alpha x \\
\quad \text{update } \alpha
\]

Figure 9.3: SOM algorithms for \( k \)-means clustering. Different ways of updating \( \alpha \) define different SOM algorithms.

Under uniform random sampling of \( \{x_1, \ldots, x_N\} \),

\[
E\left(\hat{m}_i^{(n+1)}\right) = (1 - \alpha_n) m_i^{(n)} + \alpha_n E(x) = m_i^{(n+1)};
\]

however,

\[
E\left\| x - \hat{m}_i^{(n+1)} \right\|^2 \geq E\left\| x - m_i^{(n+1)} \right\|^2
\]

and it may be that replacing \( m_i^{(n)} \) with \( \hat{m}_i^{(n+1)} \) actually increases \( \tilde{W} \). Nevertheless, we have derived a crude but plausible algorithm for attempting to decrease \( \tilde{W} \) without explicit knowledge of the partition \( \tilde{C}_1, \ldots, \tilde{C}_k \). Anticipating material in Section 9.3, we remark that this algorithm is a special case of the on-line algorithm for self-organizing maps (SOM); it is summarized in Figure 9.3.

Let us compare the exchange and SOM algorithms. Exchange algorithms decrease \( \bar{W} \), producing clusters \( (C_1^*, \ldots, C_k^*) \). If this partition minimizes \( \bar{W} \), then \( (C_1^*, \ldots, C_k^*; \bar{x}_1, \ldots, \bar{x}_k) \) minimizes \( W \). In contrast, SOM algorithms attempt to decrease \( \bar{W} \), producing representatives \( (m_1^*, \ldots, m_k^*) \). If these representatives minimize \( \bar{W} \), then \( (\tilde{C}_1, \ldots, \tilde{C}_k; m_1^*, \ldots, m_k^*) \) minimizes \( W \). Thus, both algorithms attempt to minimize the same objective function, \( W \), and we can inquire (without any prejudices about self-organizing maps) how successfully they do so.

**Example 9.2** Consider the \( N = 1000 \) points in \( \mathbb{R}^2 \) displayed in Figure 9.4. For \( k = 20 \), we attempt to minimize \( W \) using \( n = 2000 \) iterations of, respectively, an SOM algorithm and two exchange algorithms. Compared to traditional applications of \( k \)-means clustering, these are large values of \( N \) and \( k \); however, it is for large values of \( N \) and \( k \) that self-organizing maps are often recommended.

For \( (N = 1000, k = 20, n = 2000) \), we do not expect to find global—or even local—minimizers of \( W \). Instead, we investigate the effectiveness of each algorithm in finding small values of \( W \). To do so, we generate 1000 sets of initial values of \( m_1, \ldots, m_{20} \). Each set was generated by simple random sampling (without replacement) from \( \{x_1, \ldots, x_{1000}\} \). Each algorithm was started one time from each set of initial values. For the exchange algorithms, which manipulate clusters rather than representatives, the clusters were initialized by taking \( C_i \) to be the \( x_j \) to whom the nearest representative is \( m_i \). For the SOM algorithm, the values of \( \alpha \) followed Ripley’s [30, p. 323] observation that “a typical specification is that \( \alpha \) might decline linearly from 1.0 to 0.04 over 1000 examples, then linearly to zero over the second thousand...”
Each algorithm modifies its current iterate in light of a single \( x_j \). Our derivation of the SOM algorithm necessitates drawing \( x_j \) from a discrete uniform distribution on \( \{x_1, \ldots, x_{1000}\} \). Our randomized exchange algorithm does likewise; in contrast, our cyclical exchange algorithm begins each run by generating a random permutation of \( \{x_1, \ldots, x_{1000}\} \), then cycling through the \( x_j \) in the permuted order.

Each run of the SOM algorithm returns the value of \( \bar{W} \) after 2000 iterations. Each run of each exchange algorithm returns the value of \( \bar{W} \) after 2000 iterations. The five-number summaries of these samples presented in Table 9.1 clearly suggest that the SOM algorithm is competitive with—if not superior to—the exchange algorithms. These results are provocative and invite further study of the \( \bar{W} \) formulation of \( k \)-means clustering.

<table>
<thead>
<tr>
<th>Clustering Algorithm</th>
<th>min</th>
<th>( q_1 )</th>
<th>( q_2 )</th>
<th>( q_3 )</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>1531</td>
<td>1610</td>
<td>1644</td>
<td>1705</td>
<td>2218</td>
</tr>
<tr>
<td>Randomized Exchange</td>
<td>1610</td>
<td>1956</td>
<td>2101</td>
<td>2309</td>
<td>4303</td>
</tr>
<tr>
<td>Cyclical Exchange</td>
<td>1529</td>
<td>1701</td>
<td>1799</td>
<td>1917</td>
<td>2887</td>
</tr>
</tbody>
</table>

Table 9.1: Five-number summaries of 1000 values of \( W \) returned by each of three algorithms for 20-means clustering the 1000 points in Figure 9.4. The \( q_i \) are the sample quartiles.
We now introduce a 0-1 matrix that identifies cluster membership. Let \( H = [h_{rs}] \) denote the \( n \times k \) data matrix defined by setting \( h_{rs} = 1 \) if \( x_r \in C_s \) and \( h_{rs} = 0 \) otherwise. Notice that row \( r \) of \( H \) is the unit vector \( e^t_{C(x_r)} \in \mathbb{R}^k \). Clearly the problem of optimally partitioning \( x_1, \ldots, x_n \) is the problem of optimally choosing \( H \).

Observe that \( A = HH^t \) is the \( n \times n \) adjacency matrix of the undirected graph \( \mathcal{H} \) that connects two vertices if and only if they belong to the same cluster. Notice that \( H \) contains a loop at each vertex. Removing the loops results in the simple graph \( H_1 \) with adjacency matrix \( A_1 = HH^t - I \), but we find it more intuitive to work with \( H \).

Next let \( T \) denote the \( n \times n \) degree matrix of \( H \), i.e., the diagonal matrix whose diagonal entries count the numbers of edges connected to each vertex. The diagonal entry \( t_{rr} = \#(C(x_r)) \), the size of the cluster to which \( x_r \) belongs. The degree matrix of \( H_1 \) is \( T_1 = T - I \), and the two graphs have the same combinatorial Laplacian matrix:

\[
T - A = (T - I) - (A - I) = T_1 - A_1
\]

To compute the cluster sizes, we might compute the \( 1 \times k \) row vector \( e^t H \), but it will be more convenient to place these quantities on the diagonal of a \( k \times k \) matrix. This can be accomplished by computing \( N = H^t H \).

To reformulate the problem of \( k \)-means clustering, we first note that \( N^{-1}H^t X \) is the \( k \times p \) matrix whose rows contain the cluster mean vectors, i.e.,

\[
N^{-1}H^t X = \left( H^t H \right)^{-1} H^t X = \begin{bmatrix} \bar{x}^t_1 \\ \vdots \\ \bar{x}^t_k \end{bmatrix}
\]

It follows that \( H(N^{-1}H^t X) = H(H^t H)^{-1} H^t X \) is the \( n \times p \) matrix with row \( r \) equal to the mean of the cluster to which \( x_r \) belongs, i.e., \( \bar{x}_{C(x_r)} \). As a result, we can write the \( k \)-means clustering criterion \( \bar{W}(C_1, \ldots, C_k) \) as

\[
\bar{W}(H) = \left\| X - H \left( H^t H \right)^{-1} H^t X \right\|_F^2
= \left\| \left( I - H \left( H^t H \right)^{-1} H^t \right) X \right\|_F^2
= \text{trace} \left(PXX^tP^t \right),
\]

where \( P = I - HN^{-1}H^t = I - H(H^t H)^{-1} H^t \). Thus, \( k \)-means clustering can be formulated as a trace minimization problem, viz., the problem of finding \( H \) of the specified form that minimizes \( \bar{W}(H) \). Furthermore, \( \bar{W}(H) \) depends on the feature vectors only through their pairwise inner products. Because \( k \)-means clustering is unaffected by translation, we can center the feature vectors and replace \( XX^t \) with the kernel matrix \( \tilde{X} \tilde{X}^t \).

It is easily checked that \( P^t = P \) and \( P^2 = P \), hence \( P \) is a projection matrix. In fact, let \( W = HN^{-1}H^t \) denote the \( n \times n \) matrix of edge weights for the weighted undirected graph \( \mathcal{G} \) that connects two vertices if and only if they lie in the same cluster and assigns to each
edge the reciprocal of the corresponding cluster size. Then each vertex in $G$ has unit degree and the combinatorial Laplacian matrix of $G$ is

$$I - W = I - HN^{-1}H^t = P.$$ 

Now let $B = \tilde{X}\tilde{X}^t$. Exploiting the properties of $P$, we write

$$\tilde{W}(H) = \text{trace} \left( PXX^tP^t \right) = \text{trace} \left( BP^tP \right) = \text{trace} \left( BP \right) = \text{trace} \left( BP \right)$$

and conclude that $k$-means clustering is the problem of maximizing $\text{trace} \left( BP(H^tH)^{-1}H^t \right)$ subject to certain 0-1 constraints on the entries of $H$.

The kernel formulation invites the use of $k$-means clustering whenever a kernel matrix is available. Absent an algorithm that exploits the kernel structure directly, the resulting application is unimaginative: embed the kernel matrix in Euclidean space by kernel PCA, then apply one of the algorithms in Section 9.2 to the resulting Euclidean representation. If the data are represented as an undirected graph with similarity edge weights, then a natural Euclidean representation of the data is a combinatorial Laplacian eigenmap, obtained by choosing $B$ to equal either $L^\dagger$, the pseudoinverse of the combinatorial Laplacian matrix of the graph, or a low-rank approximation of it. It will be instructive to compare the resulting $k$-means problem with the ratio cut problem described in Section 10.3.1.

9.5 Exercises

Set A The following exercises use several R functions that implement the algorithms in Section 9.2. These functions are available on the web page that accompanies this book. *Note that you will have to modify the functions provided to complete the exercises.*

1. Please provide *verbal* explanations in response to the following questions:
   
   (a) How do these functions generate an initial partition of $N$ objects?
   (b) How do `kmeans.exchange` and `kmeans.exchange2` differ?

2. R contains a data set `quakes` that contains measurements on 1000 earthquakes near Fiji. We will cluster the quakes by location (latitude and longitude). Let $X$ denote the following data matrix:

$$> X \leftarrow \text{as.matrix}(\text{quakes[,1:2]})$$

Create a scatterplot of $X$ as follows:

$$> \text{plot}(X)$$

It should be obvious that there are at least two different faults responsible for these earthquakes.

3. Set $k = 8$. Apply `kmeans.exchange` and `kmeans.exchange2` to $X$. Do this several times. Which algorithm tends to produce smaller values of $W$?
4. The functions `kmeans.exchange` and `kmeans.exchange2` manipulate a modified $X$ for which the first column identifies cluster membership. Choose either of these functions and modify it so that it returns the modified $X$. Apply this function to the original $X$, save the output, and create a scatterplot in which the clusters are different colors. For example, if the modified function is `kmeans.exchange.X`, then the following commands should suffice:

```r
> X <- as.matrix(quakes[,1:2])
> j <- kmeans.start(1000,8)
> X.kmeans <- kmeans.exchange.X(X,j)
> plot(X.kmeans[,2:3],col=X.kmeans[,1])
```

5. The functions `kmeans.exchange` and `kmeans.exchange2` have an optional third argument, `niter`, that controls the number of iterations. The default is 2000. For a fixed $j$, try varying `niter` from 1000 to 10,000. Describe the effect of increasing `niter` on the final partition.

6. Repeat #4 with several different starts and `niter=10000`. Comment on the partitions that you produce. How well do your partitions represent the original data? Do any of your partitions misrepresent the original data? How?

7. How many clusters to you think that the original data contain? Why?

Set B  Let $C$ denote the following $N = 5$ points in $\mathbb{R}^2$: $(0, 0), (1, 0), (0, -1), (-1, 0), (0, 1)$. Note that the centroid of $C$ is $\bar{x} = (0, 0)$.

Let $m$ denote the current representative of $C$. To update $m$, we do the following:

(i) Draw $x \sim \text{Uniform}(C)$, i.e., each point in $C$ is equally likely to be drawn.

(ii) Replace $m$ with $(1 - \alpha)m + \alpha x$, where $\alpha \in (0, 1)$.

The new representative is closer to $\bar{x}$ iff $\| (1 - \alpha)m + \alpha x \|^2 < \|m\|^2$.

Given $m$ and $\alpha$, let

$$p(m, \alpha) = \text{Prob}\left(\| (1 - \alpha)m + \alpha x \|^2 < \|m\|^2\right).$$

1. Show that $p(m, \alpha) = 1$ if $\|m\| > 1$.

2. Let $m = (0.50, 0.25)$. Compute $p(m, \alpha)$ for (a) $\alpha = 0.96$ and (b) $\alpha = 0.04$.

Set C

1. Let $H$ be the 0-1 matrix defined in Section 9.4.

   (a) Verify that $HH^t$ is the $n \times n$ adjacency matrix of the undirected graph $\mathcal{H}$ that connects two vertices if and only if they belong to the same cluster.

   (b) Verify that $H^tH$ is the $k \times k$ diagonal matrix whose diagonal entries equal the cluster sizes.

2. Let $H$ be the 0-1 matrix defined in Section 9.4.
(a) Verify that $N^{-1}H^tX$ is the $k \times p$ matrix whose rows contain the cluster mean vectors.

(b) Verify that $H(N^{-1}H^tX)$ is the $n \times p$ matrix with row $r$ equal to the mean of the cluster to which $x_r$ belongs.

3. Let $H$ be the 0-1 matrix defined in Section 9.4 and let $P = I - HN^{-1}H^t$. Verify that $P$ is symmetric and idempotent, hence a projection matrix.

4. Suppose that the squared Euclidean distances between $x_1, x_2, x_3, x_4 \in \mathbb{R}^3$ are stored in

$$D_2 = \begin{bmatrix}
0 & 1 & 2 & 3 \\
1 & 0 & 1 & 2 \\
2 & 1 & 0 & 1 \\
3 & 2 & 1 & 0 \\
\end{bmatrix}.$$ 

What is the optimal 2-means clustering of $x_1, x_2, x_3, x_4$?
Chapter 10

Graph Partitioning

Let $G = (V,E)$ be a weighted undirected graph with $n$ vertices and edge weights stored in the symmetric $n \times n$ weight matrix $W$. Assume that $G$ is connected and that its edge weights measure similarity, so that large edge weights indicate highly similar vertices. Given $k$, we seek to partition $V$ into $k$ clusters. We might do so by combining the methods of Chapters 6 and 9–11, first constructing a Laplacian eigenmap, then applying $k$-means or model-based clustering to the resulting configuration of points. This approach to graph partitioning exemplifies spectral clustering.

10.1 The Fiedler Vector

To illustrate the potential value of spectral clustering, let $L$ denote the combinatorial Laplacian matrix of $G$, and recall that $v_0 = e/\sqrt{n}$ is an eigenvector of $L$ with eigenvalue $\lambda_0 = 0$. Let $\lambda_1 > 0$ denote the smallest strictly positive eigenvalue of $L$ and let $v_1$ denote an eigenvector that corresponds to $\lambda_1$. Fiedler [12] appears to have been the first researcher to discern the utility of $v_1$ in clustering the vertices of $G$, so that $v_1$ is often identified as the Fiedler vector.

Denote entry $i$ in $q \in \mathbb{R}^n$ by $q(i)$. We interpret $q$ as an embedding of $G$ in $\mathbb{R}$ that places vertex $i$ at location $q(i)$ and inquire what choices of $q$ reveal interesting clusterings of the vertices. We eliminate the issue of scale by requiring $q^tq = 1$. Then

$$F(q) = \sum_{i \leftrightarrow j} w_{ij} [q(i) - q(j)]^2$$

is a plausible clustering criterion, as minimizing $F(q)$ encourages placing pairs of vertices with large edge weights close together. This interpretation requires qualification, as $q = v_0$ is a global minimizer of $F$ that places all vertices at the same location and therefore is devoid of interest. However, if we restrict attention to $q \in e^\perp$, then we force some $q(i)$ to be positive and others to be negative, thereby improving the prospect that we will obtain interesting clusters of vertices.

Writing $L = [\ell_{ij}]$, we find that

$$F(q) = \sum_{i \leftrightarrow j} w_{ij} [q(i) - q(j)]^2 = -\frac{1}{2} \sum_{i,j=1}^n \ell_{ij} [q(i) - q(j)]^2$$

$$= -\frac{1}{2} \sum_{i,j=1}^n \ell_{ij} q(i)^2 + \sum_{i,j=1}^n q(i)\ell_{ij} q(j) - \frac{1}{2} \sum_{i,j=1}^n \ell_{ij} q(j)^2$$

122
\[
= -\frac{1}{2} \sum_{i=1}^{n} q(i)^2 \left( \sum_{j=1}^{n} \ell_{ij} \right) + q^t L q - \frac{1}{2} \sum_{j=1}^{n} q(j)^2 \left( \sum_{i=1}^{n} \ell_{ij} \right)
= q^t L q.
\]

Now we can apply Theorem 2.2 to conclude that \( q = v_1 \) minimizes \( F(q) \) among all \( q \in \mathbb{R}^n \) such that \( q^t q = 1 \) and \( q^t v_0 = 0 \). The eigenvector \( v_1 \), sometimes called the Fiedler vector of \( G \), is widely used to facilitate clustering graph vertices.

There remains the problem of passing from \( v_1 \) to a partition of \( V \). Having already interpreted \( v_1 \) as an embedding of \( G \) in \( \mathbb{R} \), a natural way to obtain a partition is to apply a clustering algorithm to the entries of \( v_1 \). Typically, we rely on the Fiedler vector when we desire \( k = 2 \) clusters of vertices. In doing so, we greatly simplify the search for an optimal partition, for let

\[
x_1 \leq x_2 \leq \cdots \leq x_{n-1} \leq x_n
\]

denote the ordered entries of \( v_1 \). Any plausible clustering criterion will produce a partition of the form

\[
C_1 = \{x_1, \ldots, x_{n_1}\} \quad \text{and} \quad C_2 = \{x_{n_1+1}, \ldots, x_n\}.
\]

As there are only \( n - 1 \) possible partitions of this form, one can determine the optimal partition by direct enumeration.

For example, suppose that we apply 2-means clustering to the entries of \( v_1 \) to determine a partition of \( V \). Using the kernel formulation of \( k \)-means clustering derived in Section 9.4, let \( H \) denote the \((n_1 + n_2) \times 2\) data matrix

\[
H = \begin{bmatrix}
1 & 0 \\
\vdots & \vdots \\
1 & 0 \\
0 & 1 \\
\vdots & \vdots \\
0 & 1
\end{bmatrix},
\]

in which case

\[
N = H^t H = \begin{bmatrix}
n_1 & 0 \\
0 & n_2
\end{bmatrix}.
\]

The problem of finding the optimal partition then reduces to the problem of choosing \( n_1 \) (and therefore \( n_2 = n - n_1 \)) so as to maximize

\[
\text{trace} \left( N^{-1/2} H^t v_1 v_1^t H N^{-1/2} \right) = \text{trace} \left( \left[ \frac{1}{\sqrt{n_1}} \sum_{i=1}^{n_1} x_i \right] \left[ \frac{1}{\sqrt{n_2}} \sum_{i=n_1+1}^{n} x_i \right] \right)
\]

\[
= \frac{1}{n_1} \left( \sum_{i=1}^{n_1} x_i \right)^2 + \frac{1}{n_2} \left( \sum_{i=n_1+1}^{n} x_i \right)^2
\]

\[
= \frac{1}{n_1} \left( \sum_{i=1}^{n_1} x_i \right)^2 + \frac{1}{n_2} \left( - \sum_{i=1}^{n_1} x_i \right)^2
\]

\[
= \frac{n}{n_1 n_2} \left( \sum_{i=1}^{n} x_i \right)^2.
\]
The squared sum in this expression is maximized by choosing \( n_1 \) so that \( C_1 \) contains the negative entries of \( v_1 \) and \( C_2 \) contains the positive entries of \( v_1 \). However, the influence of the leading coefficient, which is minimized by \( n_1 = n_2 \), may lead us to prefer a different partition, as in Exercise 11.1.

### 10.2 Several Eigenvectors

The rationale used in Section 10.1 to extract the Fiedler vector can be iterated to extract additional eigenvectors of \( L \). Let \( 0 < \lambda_1 \leq \cdots \leq \lambda_d \) denote the \( d \) smallest positive eigenvalues of \( L \) and let \( v_1, \ldots, v_d \) denote corresponding orthonormal eigenvectors. Again applying Theorem 2.2, we note that each \( v_i \) minimizes \( F(q) \) among all \( q \in \mathbb{R}^n \) such that \( q^t q = 1 \) and \( q^t v_0 = \cdots = q^t v_{i-1} = 0 \). Spectral clustering embeds the vertices of \( G \) in \( \mathbb{R}^d \) using \( v_1, \ldots, v_d \) as coordinate axes, then clusters the embedded points, typically by \( k \)-means clustering.

If \( d = 1 \), then it does not matter how we scale the single coordinate axis. However, if \( d > 1 \), then it does matter how we scale the \( d \) coordinate axes relative to each other. Let \( \lambda_i = \sigma_i^2 \) and recall that a combinatorial Laplacian eigenvmap constructs the configuration

\[
X = \left[ \frac{v_1}{\sigma_1} \mid \cdots \mid \frac{v_d}{\sigma_d} \right].
\]

In this configuration, the scale factors decrease as the eigenvectors diminish in importance:

\[
\frac{1}{\sigma_1} \geq \cdots \geq \frac{1}{\sigma_d}
\]

In contrast, a common practice in spectral clustering is to apply \( k \)-means clustering to the configuration

\[
Q = \left[ v_1 \mid \cdots \mid v_d \right].
\]

In this configuration, the same scale factor is used for each of the selected eigenvectors. Which configuration should one prefer?

It is instructive to consider the use of \( v_1, \ldots, v_{n-1} \), in which case the configuration \( X \) has pairwise inner products

\[
XX^t = \sum_{i=1}^{n-1} \frac{1}{\lambda_i} v_i v_i^t = L^\dagger
\]

and \( k \)-means clustering corresponds to maximizing \( \text{trace} \left( L^\dagger H^t (H^t H)^{-1} H^t \right) \) subject to certain 0-1 constraints on the entries of \( H \). The \( v_i \) are the principal components that result from embedding the expected commute times of a random walk on \( G \), and the \( 1/\lambda_i \) are their corresponding variances. Each principal component is scaled by its standard deviation and the \( d \)-dimensional principal component representation accounts for the proportion

\[
\sum_{i=1}^d \frac{1}{\lambda_i} \div \sum_{i=1}^{n-1} \frac{1}{\lambda_i}
\]

of total variation. The problem of potentially using too many principal components is in some sense self-correcting, as there are diminishing returns to including additional components.

In contrast, if we use \( v_1, \ldots, v_{n-1} \), then the configuration \( Q \) has pairwise inner products

\[
QQ^t = \sum_{i=1}^{n-1} v_i v_i^t
\]
and $k$-means clustering corresponds to maximizing $\text{trace}(QQ' H (H' H)^{-1} H')$. Clustering the $(n-1)$-dimensional configuration $Q = [v_1 \cdots v_{n-1}]$ is equivalent to clustering the $n$-dimensional configuration $Q_0 = [\vec{0} \ v_1 \cdots v_{n-1}]$, hence equivalent to clustering the translated $n$-dimensional configuration $\bar{Q}_0 = [e/\sqrt{n} \ v_1 \cdots v_{n-1}]$. But $\bar{Q}_0$ is the matrix of eigenvectors of $L$, hence orthogonal. Thus, $Q_0 \bar{Q}_0 = I$ and $k$-means clustering corresponds to maximizing

$$\text{trace} \left( H (H' H)^{-1} H' \right) = \text{trace} \left( (H' H)^{-1} H' H \right) = k.$$ 

In this setting, all partitions are equally attractive and $k$-means clustering accomplishes nothing.

In light of the preceding, if we do not scale the $v_i$ then we can only find meaningful clusters by choosing $d < n - 1$. Doing so is equivalent to scaling $v_{d+1}, \ldots, v_{n-1}$ by zero. Using $v_1, \ldots, v_d$ with scaling is equivalent to using $v_1, \ldots, v_{n-1}$ with scale factors

$$\frac{1}{\sigma_1} \geq \cdots \geq \frac{1}{\sigma_d} > 0 = \cdots = 0,$$

while using $v_1, \ldots, v_d$ without scaling is equivalent to using $v_1, \ldots, v_{n-1}$ with scale factors

$$1 = \cdots = 1 > 0 = \cdots = 0.$$

The latter may be viewed as a crude approximation of the former, but it is difficult to articulate a preference for it. The danger in using the latter is that, by choosing $d$ too large, one may introduce unwanted noise into the search for clusters without the mitigating influence of scale factors that discount the noise. Anecdotal evidence suggests that this is indeed a problem in practice.

### 10.3 Edge Cutting

A more direct approach to the problem of partitioning $V$ is to remove edges from $E$ so as to create a graph with $k$ distinct components. Such edge removals are called cuts, and researchers have proposed a variety of criteria for choosing which cuts to make. We consider two important types of cuts and explore their relation to spectral clustering.

Following Luxberg’s tutorial on spectral clustering [40], albeit with several changes in notation, let $C \subset V$ denote a subset of vertices in $G$. Two standard ways of measuring the size of $C$ are

1. $(C)$, the number of vertices in $C$; and
2. $\text{vol}(C) = \sum_{r \in C} t_r$, the volume of $C$, where $t_r$ is the degree of vertex $r$.

Let $C^c$ denote the vertices in $V$ that do not lie in $C$ and set

$$W(C, C^c) = \sum_{r \in C} \sum_{s \in C^c} w_{rs},$$

the cost of cutting $G$ into two components with vertex sets $C$ and $C^c$. The cost of partitioning $G$ into $k$ components with vertex sets $C_1, \ldots, C_k$ can then be written as

$$\text{cut}(C_1, \ldots, C_k) = \frac{1}{2} \sum_{i=1}^k W(C_i, C_i^c).$$
We might partition \( G \) by choosing \( C_1, \ldots, C_k \) to minimize \( \left( C_1, \ldots, C_k \right) \), but doing so tends to cut individual vertices from a large central component, producing dramatically unbalanced partitions. To remedy this deficiency, previous researchers have proposed modified cut criteria that divide each \( W \left( C_i, C_i^c \right) \) by the size of \( C_i \). Thus, two popular cut criteria are the following.

1. The ratio cut criterion of Hagen and Kahng [17]:

\[
Rcut \left( C_1, \ldots, C_k \right) = \frac{1}{2} \sum_{i=1}^{k} \frac{W \left( C_i, C_i^c \right)}{\# \left( C_i \right)}
\]

2. The normalized cut criterion of Shi and Malik [36]:

\[
Ncut \left( C_1, \ldots, C_k \right) = \frac{1}{2} \sum_{i=1}^{k} \frac{W \left( C_i, C_i^c \right)}{\text{vol} \left( C_i \right)}
\]

We consider each in turn.

### 10.3.1 Ratio Cuts

Let \( T = \text{diag}(W e) \) denote the degree matrix of \( G \) and let \( L = T - W \) denote the corresponding combinatorial Laplacian matrix. As in the kernel formulation of \( k \)-means clustering, explicated in Section 9.4, let \( H = [h_{rs}] \) denote the \( n \times k \) data matrix defined by setting \( h_{rs} = 1 \) if \( x_r \in C_s \) and \( h_{rs} = 0 \) otherwise. Recall that \( N = H^t H \) is the \( k \times k \) diagonal matrix whose diagonal entries are the cluster sizes, \( n_i = \# \left( C_i \right) \).

It is easily calculated that the diagonal entries of the \( k \times k \) matrix \( H^t T H \) are

\[
\sum_{r \in C_i} t_r = \sum_{r \in C_i} \sum_{s \in V} w_{rs},
\]

whereas the diagonal entries of the \( k \times k \) matrix \( H^t W H \) are

\[
\sum_{r \in C_i} \sum_{s \in C_i} w_{rs}.
\]

It follows that the diagonal entries of the \( k \times k \) matrix \( H^t L H \) are

\[
\sum_{r \in C_i} \sum_{s \notin C_i} w_{rs} = W \left( C_i, C_i^c \right),
\]

hence that the diagonal entries of the \( k \times k \) matrix

\[
N^{-1/2} H^t L H N^{-1/2}
\]

are

\[
W \left( C_i, C_i^c \right) / \# \left( C_i \right).
\]

We conclude that

\[
2Rcut \left( C_1, \ldots, C_k \right) = \text{trace} \left( N^{-1/2} H^t L H N^{-1/2} \right) = \text{trace} \left( L H \left( H^t H \right)^{-1} H^t \right),
\]

(10.1)
and that ratio cut partitioning is the problem of minimizing this expression subject to certain 0-1 constraints on the entries of $H$.

It is instructive to compare the discrete optimization problem of choosing $H$ to minimize (10.1) to the discrete optimization problem of choosing $H$ to maximize

$$\text{trace} \left( L^\dagger H \left( H^\dagger H \right)^{-1} H^\dagger \right) = \text{trace} \left( N^{-1/2} H^\dagger L^\dagger H N^{-1/2} \right).$$

(10.2)

In Section 9.4, we derived (10.2) as the objective to be maximized when performing $k$-means clustering on a combinatorial Laplacian eigenmap. We now argue that these two clustering techniques are intimately related.

Notice that the $k$ columns of $HN^{-1/2}$ are orthonormal. Suppose that we were to relax the restrictions on $H$, replacing $HN^{-1/2}$ with any $n \times k$ matrix $U$ whose columns are orthonormal. Let $\lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{n-1}$ denote the eigenvalues of $L$ and let $v_0 = e/\sqrt{n}, v_1, \ldots, v_{n-1}$ denote the corresponding eigenvectors. By Theorem 2.2,

$$U_0 = \begin{bmatrix} v_0 & \cdots & v_{k-1} \end{bmatrix}$$

minimizes $\text{trace}(U^tLU)$. Of course, as argued in our discussion of the Fiedler vector, $v_0 = e/\sqrt{n}$ is useless for clustering. If we require the columns of $U$ to lie in $e^\perp$, then $\text{trace}(U^tLU)$ is minimized by

$$U_1 = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}.$$ But Theorem 2.2 also implies that $U_1$ maximizes $\text{trace}(U^tL^\dagger U)$! Taken together, these observations suggest that partitioning a graph by the ratio cut criterion has essentially the same objective as performing $k$-means clustering on the graph’s combinatorial Laplacian eigenmap.

Because the above relation derives from relaxing the restrictions on $H$, it does not follow that the partition produced by performing $k$-means clustering on the combinatorial Laplacian eigenmap necessarily optimizes the ratio cut criterion, as illustrated by the following example.

**Example 10.1** Let $G$ be the weighted undirected graph depicted in Figure 10.1 and suppose that we seek to partition the vertices of $G$ into $k = 2$ clusters. If $\epsilon = 1$, then each cut is equally expensive,

$$Rcut(C_1,C_2) = \frac{1}{\#(C_1)} + \frac{1}{\#(C_2)},$$

and the ratio cut criterion is minimized by the cut that most evenly balances $\#(C_1)$ and $\#(C_2)$. In this case the optimal ratio cut is the 23 cut, resulting in $C_1 = \{1, 2\}$ and $C_2 = \{3, 4\}$. As $\epsilon$ is decreased, the 12 cut becomes less expensive than the 23 or 34 cuts. The ratio
cut criterion balances the appeal of a less expensive cut against an aversion to small clusters. Once $\epsilon$ is sufficiently small, the former consideration dominates, resulting in $C_1 = \{1\}$ and $C_2 = \{2, 3, 4\}$. Some calculation will be needed to determine exactly how small $\epsilon$ must be for the ratio cut criterion to prefer the 12 cut to the 23 cut.

Similar considerations arise if one clusters the vertices of $G$ by applying 2-means clustering to the 3-dimensional combinatorial eigenmap of $G$. If $\epsilon = 1$, then it turns out (see Exercise 11.3.1) that the squared Euclidean distances between the points in the eigenmap are stored in

$$D_2 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}.$$ 

It then follows from Exercise 9.5.C4 that the optimal 2-means clustering is $C_1 = \{1, 2\}$ and $C_2 = \{3, 4\}$.

Recall that the squared Euclidean distances between the points in the 3-dimensional combinatorial eigenmap of $G$ are proportional to the expected commute times of a random walk on $G$. Decreasing $\epsilon$ increases the expected commute times between the first and the other vertices. Once the corresponding $\|x_1 - x_j\|^2$ are sufficiently large, the optimal 2-means clustering will be $C_1 = \{1\}$ and $C_2 = \{2, 3, 4\}$. Again, some calculation will be needed to determine exactly how small $\epsilon$ must be for the 2-means criterion to prefer the 12 cut to the 23 cut.

Both the ratio cut criterion and 2-means clustering of the 3-dimensional combinatorial eigenmap replace the 23 cut with the 12 cut as $\epsilon$ decreases. For a more precise analysis, note that the combinatorial Laplacian matrix of $G$ is

$$L = \begin{bmatrix} \epsilon & -\epsilon & 0 & 0 \\ -\epsilon & 1 + \epsilon & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \quad (10.3)$$

Some calculation (see Exercise 11.3.2) reveals that the pseudoinverse of $L$ is

$$L^\dagger = G = \frac{1}{\epsilon} \frac{1}{16} \begin{bmatrix} 9 & -3 & -3 & -3 \\ -3 & 1 & 1 & 1 \\ -3 & 1 & 1 & 1 \\ -3 & 1 & 1 & 1 \end{bmatrix} + \frac{1}{16} \begin{bmatrix} 5 & 5 & -3 & -7 \\ 5 & 5 & -3 & -7 \\ -3 & -3 & 5 & 1 \\ -7 & -7 & 1 & 13 \end{bmatrix}. \quad (10.4)$$

The three possible cuts of $G$ define partition matrices

$$H_{12} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad H_{23} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad H_{34} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

leading to matrices

$$J_{12} = H_{12}N_{12}^{-1}H_{12}^t = \frac{1}{3} \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix},$$
\[ J_{23} = H_{23}N_{23}^{-1}H_{23}^t = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \text{and} \]
\[ J_{34} = H_{34}N_{34}^{-1}H_{34}^t = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}. \]

The cut that minimizes the ratio cut criterion is the cut for which
\[
(\operatorname{trace}(LJ_{12}), \operatorname{trace}(LJ_{23}), \operatorname{trace}(LJ_{34})) = \left(\frac{4}{3}\epsilon, 1, \frac{4}{3}\epsilon\right)
\]
is minimal; hence, the 23 cut if \(\epsilon \in (0.75, 1)\) and the 12 cut if \(\epsilon \in (0, 0.75)\). The cut preferred by 2-means clustering of the 3-dimensional combinatorial eigenmap is the cut for which
\[
(\operatorname{trace}(L^tJ_{12}), \operatorname{trace}(L^tJ_{23}), \operatorname{trace}(L^tJ_{34})) = \left(\frac{3}{4}\epsilon + \frac{5}{12}, \frac{11}{4}\epsilon + \frac{5}{12}, \frac{13}{12}\epsilon + \frac{13}{12}\right)
\]
is maximal; hence, the 23 cut if \(\epsilon \in (0.6, 1)\) and the 12 cut if \(\epsilon \in (0, 0.6)\). Although the two techniques behave similarly, they prefer different cuts when \(\epsilon \in (0.60, 0.75)\).

Further insight is gained by examining the 3-dimensional combinatorial Laplacian eigenmap of \(G\). We know that \(L\) has real eigenvalues \(\lambda_0 = 0 < \lambda_1 \leq \lambda_2 \leq \lambda_3\). The eigenvalues must satisfy
\[
\det (L - \lambda I) = \lambda \left[ \lambda^3 - 4\lambda^2 + (3 + \epsilon)\lambda - 2\epsilon \right] = 0.
\]
One can deduce that the cubic equation \(\lambda^3 - 4\lambda^2 + (3 + \epsilon)\lambda - 2\epsilon = 0\) has three distinct roots; hence, \(\lambda_1 < \lambda_2 < \lambda_3\). Furthermore, as \(\epsilon \to 0\), \(\lambda_3 \to 3\), \(\lambda_2 \to 1\), and \(\lambda_1 \to 0\). If \(v\) is an eigenvector that corresponds to one of these eigenvalues, then
\[
\begin{bmatrix} 0 & 0 & -1 & 1 - \lambda \\ 0 & -1 & 2 - \lambda & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} v = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\]
and it follows that \(v^t\) is proportional to
\[
\begin{bmatrix} -\lambda - 3 & \lambda - 1 & \lambda^2 - 3\lambda + 1 & 1 - \lambda \end{bmatrix}.
\]
As \(\epsilon \to 0\), these eigenvectors tend to
\[
\bar{v}_3 = \frac{1}{\sqrt{6}} \begin{bmatrix} 0 \\ 1 \\ -2 \\ 1 \end{bmatrix}, \quad \bar{v}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}, \quad \text{and} \quad \bar{v}_1 = \frac{1}{\sqrt{12}} \begin{bmatrix} -3 \\ 1 \\ 1 \\ 1 \end{bmatrix}.
\]

The combinatorial Laplacian eigenmap of \(G\) is
\[
X = \begin{bmatrix} \frac{x_1^t}{\sqrt{\lambda_1}} & \frac{x_2^t}{\sqrt{\lambda_2}} & \frac{x_3^t}{\sqrt{\lambda_3}} \end{bmatrix} = \begin{bmatrix} x_1^t \\ x_2^t \\ x_3^t \\ x_4^t \end{bmatrix}.
\]
with pairwise squared Euclidean distances

\[ d_{ij}^2 = \|x_i - x_j\|^2 = \frac{(v_{1i} - v_{1j})^2}{\lambda_1} + \frac{(v_{2i} - v_{2j})^2}{\lambda_2} + \frac{(v_{3i} - v_{3j})^2}{\lambda_3}. \]

For \( r = 1, 2, 3 \), let

\[ \Lambda_r = (\lambda_r - 3)^2 (\lambda_r - 1)^2 + (\lambda_r^2 - 3\lambda_r + 1)^2 + (1 - \lambda_r)^2 + 1. \]

Then, as \( \epsilon \to 0 \),

\[
\begin{align*}
  d_{34}^2 &= \frac{(1 - \lambda_1 - 1)^2/\Lambda_1}{\lambda_1} + \frac{(1 - \lambda_2 - 1)^2/\Lambda_2}{\lambda_2} + \frac{(1 - \lambda_3 - 1)^2/\Lambda_2}{\lambda_3} \to 0 + \frac{1}{12} + \frac{3}{6} = 1, \\
  d_{23}^2 &= \frac{(\lambda_1^2 - 3\lambda_1 + 1 - 1 + \lambda_1)^2/\Lambda_1}{\lambda_1} + \frac{(\lambda_2^2 - 3\lambda_2 + 1 - 1 + \lambda_2)^2/\Lambda_2}{\lambda_2} + \frac{(\lambda_3^2 - 3\lambda_3 + 1 - 1 + \lambda_3)^2/\Lambda_2}{\lambda_3} \to \frac{0}{12} + \frac{1}{2} + \frac{3}{6} = 1, \text{ and} \\
  d_{24}^2 &= \frac{(\lambda_1^2 - 3\lambda_1 + 1 - 1)^2/\Lambda_1}{\lambda_1} + \frac{(\lambda_2^2 - 3\lambda_2 + 1 - 1)^2/\Lambda_2}{\lambda_2} + \frac{(\lambda_3^2 - 3\lambda_3 + 1 - 1)^2/\Lambda_2}{\lambda_3} \to \frac{0}{12} + \frac{4}{2} + \frac{0}{6} = 2, \\
  d_{12}^2 &= \frac{(-\lambda_1 - 3)(\lambda_1 - 1) - (\lambda_1^2 - 3\lambda_1 + 1))^2/\Lambda_1}{\lambda_1} + \frac{(-\lambda_2 - 3)(\lambda_2 - 1) - (\lambda_2^2 - 3\lambda_2 + 1))^2/\Lambda_2}{\lambda_2} + \frac{(-\lambda_3 - 3)(\lambda_3 - 1) - (\lambda_3^2 - 3\lambda_3 + 1))^2/\Lambda_2}{\lambda_3} \to \infty, \\
  d_{13}^2 &= \frac{(-\lambda_1 - 3)(\lambda_1 - 1) - 1 + \lambda_1^2/\Lambda_1}{\lambda_1} + \frac{(-\lambda_2 - 3)(\lambda_2 - 1) - 1 + \lambda_2^2/\Lambda_2}{\lambda_2} + \frac{(-\lambda_3 - 3)(\lambda_3 - 1) - 1 + \lambda_3^2/\Lambda_2}{\lambda_3} \to \infty, \text{ and} \\
  d_{14}^2 &= \frac{(-\lambda_1 - 3)(\lambda_1 - 1) - 1)^2/\Lambda_1}{\lambda_1} + \frac{(-\lambda_2 - 3)(\lambda_2 - 1) - 1)^2/\Lambda_2}{\lambda_2} + \frac{(-\lambda_3 - 3)(\lambda_3 - 1) - 1)^2/\Lambda_2}{\lambda_3} \to \infty.
\]

Thus, as \( \epsilon \to 0 \), \( x_2, x_3, x_4 \) tend to the vertices of a right isosceles triangle and \( x_1 \) becomes ever more distant, so that 2-means clustering produces \( C_1 = \{1\} \) and \( C_2 = \{2, 3, 4\} \).
10.3.2 Normalized Cuts

10.4 Exercises

1. Let $\mathcal{G} = (V, E)$ denote the weighted undirected graph depicted in Figure 10.2.
   
   (a) Determine $L$, the combinatorial Laplacian matrix of $\mathcal{G}$.
   
   (b) Compute $v_1$, the Fiedler vector of $\mathcal{G}$.
   
   (c) If we use $v_1$ to partition $V$ into $k = 2$ clusters of vertices, then which clusterings must we consider?
   
   (d) Use the $k$-means clustering criterion to partition $V$.

2. For $\epsilon = 1$ in Example 10.1, compute the following quantities:
   
   (a) $L$, the combinatorial Laplacian matrix of $\mathcal{G}$;
   
   (b) $L^\dagger$, the pseudoinverse of $L$;
   
   (c) $X$, the 3-dimensional combinatorial Laplacian eigenmap of $\mathcal{G}$; and
   
   (d) $D_2(X)$, the matrix of squared Euclidean distances between the points in $X$.

3. Show that $L$ in (10.3) and $G$ in (10.4) satisfy the Moore-Penrose conditions, thereby establishing that $L^\dagger = G$.

4. For $\epsilon = 0.01$ in Example 10.1, compute the 3-dimensional combinatorial Laplacian eigenmap of $\mathcal{G}$ and the corresponding matrix of pairwise squared Euclidean distances.
Chapter 11

Probability Models for Clustering

Chapters 8–10 have described various attempts to operationalize an intuitive understanding of internal cohesion and external separation. Each of the methods described therein defines (whether explicitly or implicitly) a different notion of cluster. In this sense, the problem of clustering is ill-defined.

Suppose that finite subsets of feature vectors are random samples of different populations. Instead of defining clusters as finite subsets, one might then identify clusters with probability distributions. This correspondence may or may not conform to our understanding of internal cohesion and external separation, as illustrated in Example 11.1. Nevertheless, the introduction of a probability model allows one to state the ill-defined problem of clustering as a well-defined problem of statistical inference. Doing so has various advantages; for example, one can then state the problem of choosing $k$ (the number of clusters/populations) as a problem of model selection.

Example 11.1

The following sections describe two plausible probability models for clustering. For both models, let $P_1, \ldots, P_k$ denote probability measures on $(\mathbb{R}^q, \mathcal{B})$ with probability density functions $p_1, \ldots, p_k$. The $P_i$ are populations, or classes, and $i = 1, \ldots, k$ are class labels.

We will restrict attention to experiments in which each population is a $q$-variate normal distribution. If $\mu \in \mathbb{R}^q$ and $\Sigma$ is a symmetric positive definite $q \times q$ matrix, then $\text{MVN}(\mu, \Sigma)$ denotes the $q$-variate normal distribution with mean vector $\mu$ and covariance matrix $\Sigma$, i.e., the probability distribution on $(\mathbb{R}^q, \mathcal{B})$ with probability density function

$$p(x) = [\det 2\pi\Sigma]^{-1/2} \exp \left[ -\frac{1}{2} (x - \mu)^t \Sigma^{-1} (x - \mu) \right].$$

Both models assume that $P_i = \text{MVN}(\mu_i, \Sigma_i)$. Typically, $\alpha$ can be any element of the simplex of probability vectors (so that $k - 1$ parameters are required to specify $\alpha$) and each $\mu_i$ can be any element of $\mathbb{R}^q$ (requiring $kq$ parameters to specify $\mu_1, \ldots, \mu_k$), but the $\Sigma_i$ are often restricted. Here are some common covariance assumptions:

- $\Sigma_i = \lambda \Sigma_0$. Each covariance matrix is a common scalar multiple $\lambda > 0$ of a known covariance matrix $\Sigma$, e.g., $\Sigma_0 = I$. This model requires 1 parameter to specify its covariance structure.

- $\Sigma_i = \lambda_i \Sigma_0$. Each covariance matrix is a scalar multiple $\lambda_i > 0$ of a known covariance matrix $\Sigma$, e.g., $\Sigma_0 = I$. This model requires $k$ parameters to specify its covariance structure.
• \(\Sigma_i = \Sigma\). This is the standard covariance structure assumed in classical multivariate statistics. Each mixture component has the same covariance matrix \(\Sigma\), which can be any element of the cone of symmetric positive definite matrices. This model requires \(q(q + 1)/2\) parameters to specify its covariance structure.

• \(\Sigma_i = \lambda_i \Sigma\). Each covariance matrix is a scalar multiple \(\lambda_i > 0\) of a common covariance matrix \(\Sigma\), which can be any element of the cone of symmetric positive definite matrices. This model requires \(q(q + 1)/2 + k - 1\) parameters to specify its covariance structure.

• Each \(\Sigma_i\) can be any element of the cone of symmetric positive definite matrices. This model requires \(kq(q + 1)/2\) parameters to specify its covariance structure.

Each of the preceding is a special case of a general framework proposed by Banfield and Raftery [2] in which 

\[\Sigma_i = \lambda_i V_i A_i V_i^t.\]

In this framework, \(V_i\) is the orthogonal matrix whose columns are the eigenvectors of \(\Sigma_i\), \(A_i\) is a diagonal matrix whose diagonal entries are proportional to the eigenvalues of \(\Sigma_i\), and \(\lambda_i\) is a constant of proportionality. See [4] for a thorough investigation of various covariance structures.

### 11.1 Label Model

#### 11.1.1 Classification Likelihood Function

For \(r = 1, \ldots, n\) and \(\theta_r \in \{1, \ldots, k\}\), assume that \(x_r \sim P_{\theta_r}\). The \(\theta_r\) are the label parameters that specify the correct class label for each \(x_r\). The number of label parameters equals the number of observations, so the label model is not a parsimonious description of the observations. However, the label parameters do specify a partition of the observations, which is precisely what clustering demands.

We assume that the \(x_r\) are independently sampled and that \(P_i = \text{MVN}(\mu_i, \Sigma_i)\). Having observed \(x_1, \ldots, x_n\), the likelihood function for the parameters

\[\theta_1, \ldots, \theta_n; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k\]

is

\[L_1(\theta_1, \ldots, \theta_n; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k|x_1, \ldots, x_n) = \prod_{r=1}^{n} p_{\theta_r}(x_r) = \prod_{r=1}^{n} \left|\det 2\pi \Sigma_{\theta_r}\right|^{-1/2} \exp \left[-\frac{1}{2} (x_r - \mu_{\theta_r})^t \Sigma_{\theta_r}^{-1} (x_r - \mu_{\theta_r})\right],\]

sometimes called the classification likelihood. A maximum likelihood estimate of these parameters is a maximizer of \(L_1\).

#### 11.1.2 Model-Based Clustering

A natural strategy for maximizing \(L_1\) is variable alternation. On one hand, if \(\theta_1, \ldots, \theta_n\) are fixed, then the problem of finding \(\mu_1, \ldots, \mu_k\) and \(\Sigma_1, \ldots, \Sigma_k\) that maximize \(L_1\) is the problem
of constructing maximum likelihood estimates from \( k \) independent multivariate normal samples. The maximum likelihood estimate of population mean \( \mu_i \) is the corresponding sample mean,

\[
\hat{\mu}_i = \sum_{\theta_r = i} x_r / n_i, \quad \text{where} \quad n_i = \sum_{\theta_r = i} 1.
\]

The covariance estimates depend on the restrictions placed on covariance structure. With no restrictions,

\[
\hat{\Sigma}_i = \frac{1}{n_i} \sum_{\theta_r = i} (x_r - \hat{\mu}_i) (x_r - \hat{\mu}_i)^t;
\]

if \( \Sigma_i = \Sigma \), then

\[
\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{k} n_i \hat{\Sigma}_i;
\]

if \( \Sigma_i = \lambda I \), then

\[
\hat{\lambda} = \frac{1}{nq} \sum_{r=1}^{n} \|x_r - \hat{\mu}_{\theta_r}\|^2.
\]

On the other hand, \( \mu_1, \ldots, \mu_k \) and \( \Sigma_1, \ldots, \Sigma_k \) are fixed, then the problem of finding \( \theta_1, \ldots, \theta_n \) that maximize \( L_1 \) is the problem of assigning each \( x_r \) to a population \( P_i = \text{MVN}(\mu_i, \Sigma_i) \). The algorithm that performs these two optimizations in alternation is summarized in Figure 11.1. Under suitable regularity conditions, it monotonically increases and converges to a stationary point of \( L_1 \).

The algorithm summarized in Figure 11.1 exemplifies model-based clustering. Its structure evokes the structure of Lloyd’s algorithm for \( k \)-means clustering, summarized in Figure 9.1. Both algorithms construct initial cluster representations, then alternate between determining the clusters that correspond to these representations and updating the representations. In fact, suppose that \( \Sigma_i = \lambda I \). Then

\[
\sum_{r=1}^{n} \log \left( 2\pi \det \hat{\Sigma}_{\theta_r} \right) + (x_r - \hat{\mu}_{\theta_r})^t \hat{\Sigma}_{\theta_r} (x_r - \hat{\mu}_{\theta_r}) = nq \log \left( 2\pi \hat{\lambda} \right) + \frac{1}{\hat{\lambda}} \sum_{r=1}^{n} \|x_r - \hat{\mu}_{\theta_r}\|^2
\]

is minimized by setting each \( \theta_r \) equal to the \( i \) for which \( \|x_r - \mu_i\| \) is minimal, i.e., by assigning \( x_r \) to the nearest of \( k \) means. With this simplified covariance structure, model-based clustering simplifies to Lloyd’s algorithm for \( k \)-means clustering.

Recognizing that \( k \)-means clustering is a special case of model-based clustering provides valuable insight into the behavior of the former. From the model-based perspective, \( k \)-means clustering attempts to model data with normal distributions having identical spherical covariance structure. This explains the tendency of \( k \)-means clustering to prefer roughly spherical clusters of roughly comparable size, and reveals the limitations of \( k \)-means clustering when more complicated covariance structures are warranted.

### 11.2 Mixture Model

#### 11.2.1 Finite Mixture Distributions

Let \( \alpha = (\alpha_1, \ldots, \alpha_k) \) denote a probability vector, i.e., \( \alpha_1 + \cdots + \alpha_k = 1 \) and \( \alpha_i \geq 0 \). The mixture experiment

\[
x \sim P = \alpha_1 P_1 + \cdots + \alpha_k P_k
\]
Assume that \( \theta_r \in \{1, \ldots, k\} \) and that \( x_r \sim \text{MVN}(\mu_{\theta_r}, \Sigma_{\theta_r}) \). To partition \( x_1, \ldots, x_n \) into \( k \) clusters, construct the following sequence of iterates:

1. Construct initial estimates \( \hat{\mu}_1, \ldots, \hat{\mu}_k; \hat{\Sigma}_1, \ldots, \hat{\Sigma}_k \).

2. Do until termination:

   (a) Determine clusters by choosing \( \hat{\theta}_1, \ldots, \hat{\theta}_n \) to minimize

\[
-2 \log \mathcal{L}_1 \left( \theta_1, \ldots, \theta_n; \hat{\mu}_1, \ldots, \hat{\mu}_k; \hat{\Sigma}_1, \ldots, \hat{\Sigma}_k \mid x_1, \ldots, x_n \right)
= \sum_{r=1}^n \log \left( \det 2\pi \hat{\Sigma}_{\theta_r} \right) + \left( x_r - \hat{\mu}_{\theta_r} \right)^t \hat{\Sigma}_{\theta_r} \left( x_r - \hat{\mu}_{\theta_r} \right).
\]

   (b) Update cluster representations by computing

\[
\hat{\mu}_i = \frac{\sum_{\hat{\theta}_r = i} x_r}{n_i}, \quad \text{where} \quad n_i = \sum_{\hat{\theta}_r = i} 1,
\]

and \( \hat{\Sigma}_i \), where \( \hat{\Sigma}_i \) depends on the covariance structure.

---

**Figure 11.1:** An algorithm for model-based clustering. Each cluster is identified with a multivariate normal distribution.

is defined in Figure 11.2. In this model, the \( P_i \) are the mixture components and the \( \alpha_i \) are mixing proportions. Again, we assume that \( P_i = \text{MVN}(\mu_i, \Sigma_i) \). Notice that the probability density function that corresponds to \( P \) is

\[
p = \alpha_1 p_1 + \cdots + \alpha_k p_k.
\]

(See Exercise 11.3.2.)

Unlike the label model, the mixture model specifies a particular mechanism for choosing the population from which \( x_r \) is sampled. The mixture model is considerably more parsimonious than the label experiment, replacing \( n \) label parameters with \( k - 1 \) mixing parameters. However, whereas the label parameters specify a partition of \( x_1, \ldots, x_n \), it is not obvious how to construct a partition from a mixture model.

### 11.2.2 Estimating Normal Mixture Parameters

If \( P \) is a mixture and \( x_1, \ldots, x_n \sim P \), then the ill-defined problem of partitioning the random sample \( x_1, \ldots, x_n \) into clusters may be less interesting than the problem of estimating the mixture components and mixing proportions of \( P \). Having observed \( x_1, \ldots, x_n \), the likelihood function for the normal mixture parameters \( \alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k \) is

\[
\mathcal{L}_2 (\alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k \mid x_1, \ldots, x_n) = \prod_{r=1}^n p (x_r) = \prod_{r=1}^n \prod_{i=1}^k \alpha_i \left[ \det 2\pi \Sigma_i \right]^{-1/2} \exp \left[ -\frac{1}{2} (x_r - \mu_i)^t \Sigma_i^{-1} (x_r - \mu_i) \right]
\]

and a maximum likelihood estimate of the parameters is a maximizer of \( \mathcal{L}_2 \).
Let $P_1, \ldots, P_k$ denote probability measures on $(\mathbb{R}^q, B)$ and let $\alpha = (\alpha_1, \ldots, \alpha_k)$ denote a probability vector. The mixture experiment
\[ x \sim P = \alpha_1 P_1 + \cdots + \alpha_k P_k \]
is defined by the following 2-step procedure:

1. First draw $i \sim \text{Multinomial}(\alpha)$; then
2. Draw $x \sim P_i$.

Figure 11.2: A finite mixture experiment. In the case of supervised learning (Part IV), both $i$ and $x$ are observed; in the case of unsupervised learning (Chapter 11), only $x$ is observed.

It is difficult to maximize $\mathcal{L}_2$ directly, but a clever trick is available. Consider Bernoulli random variables defined by
\[ Y_{ri} = \begin{cases} 1 & \text{if } x_r \text{ was drawn from } P_i \\ 0 & \text{otherwise} \end{cases} \]
Mixture experiments are constructed so that $(y_{r1}, \ldots, y_{rk}) \sim \text{Multinomial}(\alpha)$, with marginal density function
\[ \prod_{i=1}^{k} \alpha_{yi}^{y_{ri}}. \]
The conditional density of $x_r$ given $(y_{r1}, \ldots, y_{rk})$ can be written as
\[ \prod_{j=1}^{k} p_j (x_r|\mu_j, \Sigma_j)^{y_{rj}}; \]
hence, the joint density of $x_r, y_{r1}, \ldots, y_{rk}$ can be written as
\[ \prod_{i=1}^{k} \alpha_{yi}^{y_{ri}} \cdot \prod_{j=1}^{k} p_j (x_r|\mu_j, \Sigma_j)^{y_{rj}} = \prod_{i=1}^{k} [\alpha_i p_i(x_r|\mu_i, \Sigma_i)]^{y_{ri}}. \]
In a mixture experiment the $y_{ri}$ are not observed; however, if they were known, then we could obtain maximum likelihood estimates of $\alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k$ by maximizing
\[ \ell (\alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k) = \log \prod_{r=1}^{n} \prod_{i=1}^{k} [\alpha_i p_i(x_r|\mu_i, \Sigma_i)]^{y_{ri}} = \sum_{r=1}^{n} \sum_{i=1}^{k} y_{ri} \log [\alpha_i p_i(x_r|\mu_i, \Sigma_i)]. \]
This optimization is straightforward, resulting in
\[ n_i = \sum_{r=1}^{n} y_{ri}, \quad \hat{\alpha}_i = n_i/n, \quad \text{and} \quad \hat{\mu}_i = \sum_{r=1}^{n} y_{ri} x_r/n_i. \]
Again, the covariance estimates depend on the restrictions placed on covariance structure.
With no restrictions,
\[ \hat{\Sigma}_i = \frac{1}{n_i} \sum_{r=1}^{n} (y_{ri}x_r - \hat{\mu}_i) (y_{ri}x_r - \hat{\mu}_i)^t; \]
if $\Sigma_i = \Sigma$, then

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{k} n_i \hat{\Sigma}_i.$$  

Of course, the $y_{ri}$ are not observed. Suppose that we knew $\alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k$, in which case we could compute

$$\hat{y}_{ri} = E(Y_{ri}|x_1, \ldots, x_r) = \frac{\alpha_i p_i(x_r|\mu_i, \Sigma_i)}{\sum_{j=1}^{k} \alpha_j p_j(x_r|\mu_j, \Sigma_j)}.$$  

We might then *impute* the missing $y_{ri}$, replacing each with $\hat{y}_{ri}$. Thus, a dilemma. On one hand: it would be easy to estimate $\alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k$ if only we knew the unobserved $y_{ri}$. On the other hand: it would be easy to impute the unobserved $y_{ri}$ if only we knew the unknown $\alpha; \mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k$. The solution is irresistible: construct initial parameter estimates, use these estimates to impute the unobserved $y_{ri}$, then use the imputed $y_{ri}$ to construct better parameter estimates, etc. This is the celebrated EM algorithm of Dempster, Laird, and Rubin [10], which successively alternates imputation (E for expectation) and maximum likelihood estimation (M for maximization). Wu [42] demonstrated that, under suitable regularity conditions, the sequence of parameter estimates constructed by the EM algorithm has monotonically increasing likelihood and converges to a stationary point of $L_2$.

### 11.2.3 Clustering Normal Mixture Samples

The EM algorithm produces maximum likelihood estimates of the parameters of a normal mixture model, but it does not produce an explicit partition of $x_1, \ldots, x_n$. Such a partition, however, is easily extracted from EM calculations by recognizing the $\hat{y}_{ri}$ as conditional probabilities of cluster membership. A natural clustering assigns each $x_r$ to the $i$ for which $\hat{y}_{ri}$ is maximal.

### 11.3 Exercises

1. Suppose that $x_1, \ldots, x_n \sim \text{MVN}(\mu, \lambda I)$, where $\mu \in \mathbb{R}^q$ and $\lambda \in (0, \infty)$ are unknown. Derive the maximum likelihood estimates of $\mu$ and $\lambda$.

2. Suppose that $P_i$ has probability density function $p_i$, $i = 1, \ldots, k$. Show that $P = \alpha_1 P_1 + \cdots + \alpha_k P_k$ has probability density function $p = \alpha_1 p_1 + \cdots + \alpha_k p_k$, i.e., that

$$P(X \in A) = \int_A p(x) \, dx$$

for any rectangle $A \subset \mathbb{R}^q$. 
To estimate mixing proportions $\alpha_1, \ldots, \alpha_k$ and population parameters
$\mu_1, \ldots, \mu_k; \Sigma_1, \ldots, \Sigma_k$ for the normal mixture experiment
\[ x_1, \ldots, x_n \sim \sum_{i=1}^{k} \alpha_i \text{MVN} (\mu_i, \Sigma_i), \]
construct the following sequence of iterates:

1. Construct initial estimates $\hat{\alpha}_1, \ldots, \hat{\alpha}_k; \hat{\mu}_1, \ldots, \hat{\mu}_k; \hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$.

2. Do until termination:
   (a) E-step: impute
   \[ \hat{y}_{ri} = \frac{\hat{\alpha}_i p_i \left( x_r | \hat{\mu}_i, \hat{\Sigma}_i \right)}{\sum_{j=1}^{k} \hat{\alpha}_j p_j \left( x_r | \hat{\mu}_j, \hat{\Sigma}_j \right)}. \]
   (b) M-step: update the parameter estimates by computing
   \[ n_i = \frac{\sum_{r=1}^{n} \hat{y}_{ri}}{\hat{\alpha}_i} = n_i / n, \quad \hat{\mu}_i = \frac{\sum_{r=1}^{n} \hat{y}_{ri} x_r / n_i}{\hat{\alpha}_i}, \]
   and $\hat{\Sigma}_i$, where $\hat{\Sigma}_i$ depends on the covariance structure.

Figure 11.3: The EM algorithm for normal mixture experiments.
Part IV

Classification
Suppose that, for each of \( n \) objects, we observe a feature vector \( x_r \in \mathbb{R}^q \) and a label \( y_r \in \{1, \ldots, g\} \). The labels define classes of objects, and the fundamental problem of classification is one of class prediction: if we encounter another object and observe its feature vector \( x \) but not its class label \( y \), then how might we use \( x \) to assign a label (guess \( y \)) to that object? Two examples illustrate the diverse applications of classification.

**Example 11.2** According to the American Cancer Society, prostate cancer is the second leading cause of cancer death among American men. Unfortunately, the widely used Prostate-Specific Antigen (PSA) test is not extremely reliable. Prostate biopsies are definitive, but highly invasive. A noninvasive procedure that is less prone to error than the PSA test would be welcome.

A medical researcher studies \( n \) male patients, each of which is determined by biopsy to either have or not have prostate cancer. Each patient also contributes a urine specimen, which is subjected to mass spectrometry analysis. For each patient, the analysis reveals the mass:charge ratios of \( q \) protein fragments. The researcher hopes that the mass:charge profiles of the patients with cancer will differ from the mass:charge profiles of the healthy patients. If they do—if it is possible to predict disease status from urine specimens, then it may be possible to improve on the PSA test without resorting to biopsy.

In this example, the goal is clinical. If successful, future patients will be diagnosed (class labels will be assigned) on the basis of their mass:charge profiles (feature vectors).

**Example 11.3** Since its construction in the 1960s, Glen Canyon Dam has had profound effects on the ecology of the corridor below the dam, where the Colorado River flows through Marble and Grand Canyons. To best manage the flow of water through the dam, the Bureau of Land Management (BLM) needs to understand the nature of these effects.

In the 1980s, BLM commissioned a suite of multyear studies of the ecological effects of Glen Canyon Dam. One such study concerned effects on riparian bird species. At each of ten study sites, professional ornithologists and their assistants located a number of active bird nests. The species (class label) of the bird using the nest (object) was identified, and a number of measurements (feature vector) were taken, e.g., height of nest, density of foliage, nearby species of trees and shrubs, etc. Discriminant analysis related the labels to the feature vectors, allowing researchers to understand the nesting habitat preferences of different bird species.

In this example, the goal was not clinical. Professional ornithologists can reliably label bird nests by species without resorting to time-consuming habitat measurements. Rather, the goal of understanding habitat structure was realized by examining the ways in which habitat could be used to discriminate between species.

Our approach to classification involves a probability model. As in Section 11.2, assume that \( x_1, \ldots, x_n \in \mathbb{R}^q \) are independently drawn from a mixture distribution

\[
P = \alpha_1 P_1 + \cdots + \alpha_g P_g.
\]

Let \( y_r \in \{1, \ldots, g\} \) denote the class label that corresponds to \( x_r \), i.e., \( y_r = i \) indicates that that \( x_r \) was drawn from \( P_i \). In contrast to Chapter 11, we now assume that the \( y_r \) are known. In this supervised setting,

- \( \alpha_i = P(Y = i) \) is the prior probability of drawing \( x \) from population \( i \);
The Bayes rule assigns to each feature vector \( x \) probability, i.e., the label \( i \) error rate of the classifier \( \hat{y} \) denote the loss that will accrue from assigning label \( j \). Given \( x \in \mathbb{R}^d \) and \( y \in \{1, \ldots, g\} \), let

\[
L(y, \hat{y}(x)) = \begin{cases} 0 & \hat{y}(x) = y \\ 1 & \hat{y}(x) \neq y \end{cases}
\]

denote the loss that will accrue from assigning label \( y \) to feature vector \( x \). Then the expected error rate of the classifier \( \hat{y} \) is the probability of misclassification error,

\[
P\left( L(Y, \hat{y}(X)) = 1 \right) = E_p L(Y, \hat{y}(X))
\]

\[
= \alpha_1 E_{P_1} L(1, \hat{y}(X)) + \cdots + \alpha_g E_{P_g} L(g, \hat{y}(X))
\]

\[
= \alpha_1 P_1 (\hat{y}(X) \neq 1) + \cdots + \alpha_g P_g (\hat{y}(X) \neq g)
\]

\[
= \alpha_1 (1 - R_{11}) + \cdots + \alpha_g (1 - R_{gg}),
\]

and the expected success rate is

\[
1 - E_p L(Y, \hat{y}(X)) = \alpha_1 R_{11} + \cdots + \alpha_g R_{gg}.
\]

One useful benchmark for assessing the performance of \( \hat{y} \) is randomization. Suppose that we assign labels by randomly sampling \( \{1, \ldots, g\} \) with multinomial probabilities \( \alpha_1, \ldots, \alpha_g \). Then \( R_{ii} = \alpha_i \), yielding an expected success rate of \( \alpha_1^2 + \cdots + \alpha_g^2 \).

A second useful benchmark is obtained by ignoring \( x \) and setting \( \hat{y}(x) \) equal to the most probable class, i.e., the label \( i \) for which \( \alpha_i \) is maximal. The expected success rate is then \( \max\{\alpha_1, \ldots, \alpha_g\} \). This classifier has a higher success rate than randomization (see Exercise 13.5.1), but it requires knowledge of the prior probabilities.

A third useful benchmark is the Bayes classification rule. Let \( p(x, y) \) denote the joint probability density function of the random vector \( (X, Y) \sim P \), let \( p(x) \) denote the marginal probability density function of the random vector \( X \), and let \( p_i(x) \) denote the conditional probability density function of the random vector \( X \mid Y = i \). Then \( p(x, i) = \alpha_i p_i(x) \) and the posterior probability of \( Y = i \) given \( X = x \) is

\[
P(Y = i \mid X = x) = \frac{p(x, i)}{p(x)} = \frac{\alpha_i p_i(x)}{p(x)}.
\]

The Bayes rule assigns to each feature vector \( x \) the class label \( i \) with the largest posterior probability, i.e., the label \( i \) for which \( \alpha_i p_i(x) \) is largest. A standard result from statistical
decision theory states that the Bayes rule minimizes the expected loss, i.e., the probability of misclassification error, among all decision rules.

In practice, the $\alpha_i$ and $p_i$ are unknown and optimal performance is unattainable. However, a number of useful classification procedures can be derived by constructing estimates $\hat{\alpha}_i$ and $\hat{p}_i(x)$, then assigning the class label that maximizes $\hat{\alpha}_i \hat{p}_i(x)$.

**Example 11.4** Suppose that $\alpha_1 = \cdots = \alpha_g$, in which case the Bayes rule assigns the label that maximizes $p_i(x)$. Let $P_i = \text{MVN}(\mu_i, \Sigma)$, in which case

$$p_i(x) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_i)^t \Sigma^{-1} (x - \mu_i)\right].$$

For each $x$, $p_i(x)$ is maximized by choosing $i$ to minimize

$$(x - \mu_i)^t \Sigma^{-1} (x - \mu_i).$$

If, as is typically the case, $\mu$ and $\Sigma$ are unknown, then let $\bar{x}_i$ and $S_i$ denote the sample mean vector and sample covariance matrix of the feature vectors with label $i$ and let

$$S = \frac{1}{n - g} \sum_{i=1}^g (n_i - 1) S_i$$

denote the pooled sample covariance. Substituting $\bar{x}_i$ for $\mu_i$ and $S$ for $\Sigma$, we obtain the following classification rule:

Assign to each feature vector $x$ the class label $i$ that minimizes $(x - \bar{x}_i)^t S^{-1} (x - \bar{x}_i)$.

This rule exemplifies linear discriminant analysis (LDA), the subject of Chapter 13.

Classifiers are usually constructed by using a training sample to select one member of a family of possible decision rules. A classification technique is then a function that constructs a particular classifier from a particular training sample. In Example 11.4, the classification technique of linear discriminant analysis constructs the classifier “assign the label $i$ that minimizes $(x - \bar{x}_i)^t S^{-1} (x - \bar{x}_i)$ to the feature vector $x$” from the training sample $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. We can indicate the dependence of this rule on the training sample by writing $\hat{\phi}(\cdot; T)$.

It is conceptually appealing to measure the performance of a classifier by its misclassification error rate. It is rarely possible to determine exact error rates, so we are obliged to estimate them. Doing so requires care. The *nominal misclassification rate* of a classifier $\hat{\phi}$ is the proportion of errors in the training sample, i.e.,

$$\frac{1}{m} \sum_{(x_r, y_r) \in T} L\left(y_r, \hat{\phi}(x_r; T)\right),$$

where $m = \#(T)$ is the number of objects in $T$. However, if $\hat{\phi}$ was constructed to perform well on $T$, then the nominal rate will tend to underestimate the true misclassification rate,

$$EL\left(Y, \hat{\phi}(X; T)\right).$$

Statisticians have devised various ways of producing better estimates of misclassification rates.
To construct unbiased estimates of misclassification rates, we need to separate the training sample from the validation sample used to assess its performance. An obvious possibility is to randomly partition the sample \((x_1, y_1), \ldots, (x_n, y_n)\) into a training sample \(T\) of size \(m\) and a validation sample \(T^c\) of size \(n - m\). The training sample is used to construct \(\hat{\phi}(\cdot; T)\), which is then tested on the validation sample. The proportion of misclassified objects in the validation sample is then an unbiased estimate of

\[
EL \left( Y, \hat{\phi}(X; T) \right).
\]

Notice, however, an unfortunate dilemma. If \(m\) is too small, then the classifier may perform badly; but if \(m\) is too large, then not many objects are used for validation and the estimated misclassification rate may be highly variable. A typical choice is \(m/n \approx 2/3\).

The aforementioned dilemma can be partly circumvented by \(V\)-fold cross-validation, which randomly partitions the sample \((x_1, y_1), \ldots, (x_n, y_n)\) into subsets \(S_1, \ldots, S_V\) of approximately equal size. For \(j = 1, \ldots, V\), let

\[
T_{-j} = \bigcup_{\ell \neq j} S_\ell
\]

and let \(j(r)\) denote the subset that contains object \(r\). The cross-validation estimate of the misclassification rate is then

\[
CV = \frac{1}{n} \sum_{r=1}^n L \left( y_r, \hat{\phi}(x_r; T_{-j(r)}) \right).
\]

Notice that cross-validation requires construction of \(V\) classifiers. Every object is used in testing, but no object is used to construct the classifier tested on that object. It is usually recommended that one choose \(V \in \{5, \ldots, 10\}\), although \(V = n\) ("leave-one-out" cross-validation) is often used in practice.
Chapter 12

Nearest Neighbor Classifiers

12.1 Nearest Neighbor Rules

Recall that the Bayes rule assigns to each feature vector \( x \) the class label \( i \) with the largest posterior probability, i.e., the label \( i \) for which \( \alpha_i p_i(x) \) is largest. In practice, neither the \( \alpha_i \) nor the \( p_i \) are known. A variety of useful classification rules can be derived by estimating \( \alpha_i \) and \( p_i \) in various ways.

Assuming that the \((x_1, y_1), \ldots, (x_1, y_1)\) were independently generated by a mixture experiment, the obvious estimate of \( \alpha_i \) is the observed proportion of times that label \( i \) appears in the sample:

\[
\hat{\alpha}_i = \frac{n_i}{n}, \quad \text{where} \quad n_i = \# \{y_r = i\}.
\]

To obtain a nonparametric estimate of \( p_i(x) \), let \( B(x; \epsilon) \) denote a ball of radius \( \epsilon \) centered at \( x \), let \( \text{vol}(\epsilon) \) denote the volume of \( B(x; \epsilon) \), and let \( k(i) \) denote the number of \( x_r \in B(x; \epsilon) \) for which \( y_r = i \). Then \( k(i)/n_i \) is the proportion of all feature vectors drawn from \( p_i \) that lie in \( B(x; \epsilon) \), and

\[
\hat{p}_i(x) = \frac{k(i)/n_i}{\text{vol}(\epsilon)} \quad \text{estimates} \quad p_i(x).
\]

One can then approximate the Bayes rule by choosing \( i \) to maximize

\[
\hat{\alpha}_i \hat{p}_i(x) = \frac{n_i k(i)/n_i}{\text{vol}(\epsilon)} = \frac{k(i)}{n \text{vol}(\epsilon)},
\]

in which case it suffices to choose \( i \) to maximize \( k(i) \). This is a simple, conceptually appealing rule: choose that label that appears most frequently in an \( \epsilon \)-neighborhood of an unlabelled feature vector. But how to select \( \epsilon \)?

We might select any \( \epsilon > 0 \), although some \( \epsilon \) are clearly too small (because \( B(x; \epsilon) \) contains none of the observed \( x_r \)) and some \( \epsilon \) are clearly too large (because the labels of \( x_r \) that are distant from \( x \) are not useful for classifying \( x \)). Notice, however, that the values of \( k(1), \ldots, k(g) \) remain constant over intervals of \( \epsilon \) for which \( k = k(1) + \cdots + k(g) \) remain constant. There are only finitely many such intervals, each of which is associated with the number \( k \) of \( x_r \) deemed neighbors of \( x \). It suffices, therefore, to consider only \( \epsilon \) equal to the distance from \( x \) to the nearest \( x_r \), to the second nearest \( x_r \), etc. By so doing, one can formulate the classification rule in terms of the \( k \) nearest neighbors of \( x \), as in Figure 12.1. The choice of \( k \) is often made by leave-one-out cross-validation on a training sample.
Given labelled feature vectors $x_1, \ldots, x_n \in \mathbb{R}^q$, the KNN classifier is constructed as follows:

1. Fix $k$, the number of nearest neighbors to be considered.
2. For any $x \in \mathbb{R}^q$, let $k(i)$ denote the number of $k$ neighbors with label $i$.
3. Assign to $x$ the label $i$ for which $k(i)$ is largest.

Figure 12.1: The $k$ nearest neighbor (KNN) classifier. The choice of $k$ is often made by leave-one-out cross-validation on a training sample.

KNN rules are often described as elections, in which the $k$ nearest neighbors vote for labels. In practice, one must anticipate the possibility of a tie, in which two (or more) of the leading labels receive the same number of votes. If $g = 2$, then ties can be avoided by choosing $k$ odd. More generally, various tie-breaking schemes have been proposed in the literature, including randomization and letting the nearest-neighbor decide. The probability of ties can be reduced by abandoning the principle of “one neighbor one vote”. Various methods have been proposed for weighting votes, typically so that the votes of the nearest neighbors receive the greatest weight.

12.2 Editing KNN Rules for Better Performance

12.3 Editing KNN Rules for Greater Parsimony

12.4 Distance Metric Learning
Chapter 13

Linear Discriminant Analysis

13.1 Linear Classification

A classifier \( \hat{\phi} : \mathbb{R}^q \to \{1, \ldots, g\} \) partitions \( \mathbb{R}^q \) into decision regions \( \{ x \in \mathbb{R}^q : \hat{\phi}(x) = i \} \) and the decision boundaries of those regions. If the decision boundaries of \( \hat{\phi} \) are piecewise linear, then we say that \( \hat{\phi} \) is a linear classifier. We begin by considering the simplest case of a linear classifier, which occurs when \( g = 2 \) and the decision boundary is a \((q - 1)\)-dimensional hyperplane \( H \) that separates \( \mathbb{R}^q \) into half-spaces \( H_i = \{ x \in \mathbb{R}^q : \hat{\phi}(x) = i \} \).

Let \( \bar{w} \) denote the minimizer of \( \|x\|^2 \) among all \( x \in H \) and let \( w \) denote any positive scalar multiple of \( \bar{w} \). By the normal equations, \( w \) is orthogonal to \( H \), i.e., if \( x \in H \), then \( w^t(x - \bar{w}) = 0 \). Furthermore, because \( \dim H = q - 1 \), \( x \in H \) only if \( w^t(x - \bar{w}) = 0 \). Hence, the separating hyperplane \( H \) is characterized by the equation

\[
H = \{ x \in \mathbb{R}^q : w^tx = \theta \},
\]

where \( \theta = w^t\bar{w} \). The half-spaces are then defined by

\[
H_1 = \{ x \in \mathbb{R}^q : w^tx < \theta \} \quad \text{and} \quad H_2 = \{ x \in \mathbb{R}^q : w^tx > \theta \},
\]

or vice versa.

We can now write \( H \) as \( H(w, \theta) \), but notice that \( H \) is not identified with a unique \((w, \theta)\): for any \( \lambda > 0 \), \( H(\lambda w, \lambda \theta) = H(w, \theta) \). To ensure a unique identification, we must impose a normalizing condition, e.g., that \( \theta = 1 \), or \( w^te = 1 \), or \( w^tw = 1 \). Having done so, the problem of choosing an optimal separating hyperplane can be stated as the problem of choosing an optimal \((w, \theta)\).

There are two obvious ways to measure how far a feature vector \( x \) lies from the decision boundary \( H(w, \theta) \):

1. Compute the Euclidean distance between \( x \) and \( H(w, \theta) \), i.e., the minimal Euclidean distance between \( x \) and the nearest feature vector that lies in \( H(w, \theta) \). Let \( \bar{x} \) denote the projection of \( x \) into \( H(w, \theta) \), so that the desired distance is \( \|x - \bar{x}\| \). Let \( \hat{x} \) denote the projection of \( x \) into the 1-dimensional linear subspace

\[
L = H(w, \theta)^\perp = \{ x = tw : t \in \mathbb{R} \},
\]

and notice that

\[
\|x - \bar{x}\| = \|\hat{x} - \bar{w}\|.
\]
2. Compute \(|w^t x - \theta|\). Set \(u = w/\|w\|\), write \(L = \{x = tu : t \in \mathbb{R}\}\), and apply Exercise 3.5.1 to conclude that
\[\hat{x} = (u^t x) u = \left(\frac{w^t x}{\|w\|^2}\right) w/\|w\|^2 \quad \text{and} \quad \bar{w} = \hat{w} = \left(\frac{w^t \bar{w}}{\|w\|^2}\right) \theta w/\|w\|^2.\]

Hence,
\[
\|\hat{x} - \bar{w}\| = \left\|\left(\frac{w^t x}{\|w\|^2} - \theta w\right) / \|w\|\right\| = \left|\frac{w^t x - \theta}{\|w\|}\right|
\]
revealing that the two proposed measures are equivalent (and identical if \(\|w\| = 1\)).

The preceding remarks demonstrate that, if \(g = 2\) and the decision boundary is a separating hyperplane, then classification of \(x \in \mathbb{R}^q\) can be reduced to classification of \(\hat{x} \in L\). The 1-dimensional linear subspace \(L\) is an example of a discriminant space.

### 13.2 Fisher’s Best Linear Discriminator

Suppose that \(g = 2\) and we restrict attention to classifiers whose decision boundaries are separating hyperplanes \(H(a, \theta)\) with \(\|a\| = 1\). A training sample comprises \(x_{11}, \ldots, x_{1n_1}\) with labels \(y_{1r} = 1\) and \(x_{21}, \ldots, x_{2n_2}\) with labels \(y_{1r} = 2\). How should we choose \((a, \theta)\)?

Projection into \(L = \{x = ta : t \in \mathbb{R}\}\) replaces each \(x_{ir} \in \mathbb{R}^q\) with \(a^t x_{ir} \in \mathbb{R}\).

### 13.3 Discriminant Coordinates

The exposition that follows will be simplified by the use of double subscripts. Let \(n_i\) denote the number of feature vectors with label \(i \in \{1, \ldots, g\}\) and let \(x_{ir} \in \mathbb{R}^q\) denote feature vector \(r\) among the \(n_i\) feature vectors with label \(i\). Clearly \(n_1 + \cdots + n_g = n\) and we can sum an expression \(\xi (i, r)\) over the \(n_i\) feature vectors by computing
\[
\sum_{i=1}^{g} \sum_{r=1}^{n} \xi(i, r).
\]
For example, the centroid of all \(n\) feature vectors is
\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{g} \sum_{r=1}^{n} x_{ir},
\]
while the centroid of the \(n_i\) feature vectors with label \(i\) is
\[
\bar{x}_i = \frac{1}{n_i} \sum_{r=1}^{n_i} x_{ir}.
\]

Suppose that we wish to perform a univariate analysis of variance (anova) in the case of a single measurement variable \((q = 1)\). There are three quantities of interest:

1. Total sum of squares,
\[
T_1 = \sum_{i=1}^{g} \sum_{r=1}^{n_i} (x_{ir} - \bar{x}.)^2,
\]
which for each \(ir\) compares \(x_{ir}\) to \(\bar{x}.\). Notice that \(T_1/(n - 1)\) is the sample variance for the entire set of \(x_{ir}\).
2. Within-group sum of squares,

\[ W_1 = \sum_{i=1}^{g} \sum_{r=1}^{n_i} (x_{ir} - \bar{x}_i)^2, \]

which for each \( ir \) compares \( x_{ir} \) to \( \bar{x}_i \). Notice that \( W_1/(n-g) \) is the pooled within-group sample variance.

3. Between-group sum of squares,

\[ B_1 = \sum_{i=1}^{g} \sum_{r=1}^{n_i} (\bar{x}_i - \bar{x}_\cdot)^2 = \sum_{i=1}^{g} n_i (\bar{x}_i - \bar{x}_\cdot)^2, \]

which for each \( ir \) compares \( \bar{x}_i \) to \( \bar{x}_\cdot \).

These quantities are related by the equation \( T_1 = W_1 + B_1 \), which is simply the Pythagorean Theorem applied to a right triangle with vertices in \( \mathbb{R}^n \). The test statistic

\[ F_1 = \frac{B_1/(g-1)}{W_1/(n-g)} = \frac{n-g B_1}{g-1 W_1} \]

quantifies the strength of the evidence against the null hypothesis of equal population means.

Analogous quantities exist in the case of a multivariate analysis of variance (manova) on observed values of the measurement variables \( M_1, \ldots, M_q \):

1. Total sum of squares and products,

\[ T = \sum_{i=1}^{g} \sum_{r=1}^{n_i} (x_{ir} - \bar{x}_\cdot) (x_{ir} - \bar{x}_\cdot)^t. \]

Notice that \( T/(n-1) \) is the sample covariance matrix for the entire set of \( x_{ir} \). Let \( \lambda_1 \geq \cdots \geq \lambda_q \geq 0 \) denote the eigenvalues of \( T \) and let \( v_1, \ldots, v_q \) denote corresponding orthonormal eigenvectors. The \( v_1, \ldots, v_q \) are the principal components of the entire set of feature vectors.

2. Within-group sum of squares and products,

\[ W = \sum_{i=1}^{g} \sum_{r=1}^{n_i} (x_{ir} - \bar{x}_i) (x_{ir} - \bar{x}_i)^t = \sum_{i=1}^{g} (n_i - 1) S_i = (n - g)S, \]

where \( S_i \) is the sample covariance matrix for group \( i \) and \( S \) is the pooled within-group sample covariance matrix. We will assume that \( W \) is invertible.

3. Between-group sum of squares and products,

\[ B = \sum_{i=1}^{g} n_i (\bar{x}_i - \bar{x}_\cdot) (\bar{x}_i - \bar{x}_\cdot)^t. \]

Notice that \( B/(n-1) \) is the sample covariance matrix of a data set that consists of \( n_i \) copies of \( \bar{x}_i, i = 1, \ldots, g \). Because it requires no more than \( g - 1 \) dimensions to describe \( g \) distinct points, \( k = \text{rank}(B) \leq g - 1 \).
As in the univariate case, $T = W + B$.

The following fact will become important. Suppose that $W = I$, in which case the $k$ largest eigenvalues of $B$ are $\lambda_1 - 1 \geq \cdots \geq \lambda_k - 1 > 0$. The corresponding eigenvectors of $B$ are then $v_1, \ldots, v_k$, the first $k$ principal components of the $x_{ir}$, and each $\bar{x}_i - \bar{x}. \in \text{span}\{v_1, \ldots, v_k\}$.

Suppose that $a \in \mathbb{R}^q$ and we form the measurement variable $a^t M = a_1 M_1 + \cdots + a_q M_q$. To perform a univariate anova on the observed values $a^t M$, we require the following quantities:

1. Total sum of squares,

$$T_a = \sum_{i=1}^g \sum_{r=1}^{n_i} (a^t x_{ir} - a^t \bar{x}.)^2 = \sum_{i=1}^g \sum_{r=1}^{n_i} a^t (x_{ir} - \bar{x}.)(x_{ir} - \bar{x}.)^t a = a^t T a.$$

2. Within-group sum of squares,

$$W_a = \sum_{i=1}^g \sum_{r=1}^{n_i} (a^t x_{ir} - a^t \bar{x}_i)^2 = \sum_{i=1}^g \sum_{r=1}^{n_i} a^t (x_{ir} - \bar{x}_i)(x_{ir} - \bar{x}_i)^t a = a^t W a.$$

3. Between-group sum of squares,

$$B_a = \sum_{i=1}^g n_i (a^t \bar{x}_i - a^t \bar{x}.)^2 = \sum_{i=1}^g n_i a^t (\bar{x}_i - \bar{x}.)(\bar{x}_i - \bar{x}.)^t a = a^t B a.$$

The corresponding test statistic is

$$F_a = \frac{n - g a^t B a}{g - 1 a^t W a}.$$

Linear discriminant analysis depends on finding $a \in \mathbb{R}^q$ that maximize $F_a$.

Let $W^{1/2}$ denote the symmetric square root of $W$ and set $\tilde{a} = W^{1/2}a$, so that

$$F_a = \frac{a^t W^{1/2} W^{-1/2} B W^{-1/2} W^{1/2} a}{a^t W^{1/2} W^{1/2} a} = \tilde{a}^t W^{-1/2} B W^{-1/2} \tilde{a} = \left( \tilde{a}^t \frac{\parallel \tilde{a} \parallel}{\parallel a \parallel} \right)^t W^{-1/2} B W^{-1/2} \left( \frac{\tilde{a}}{\parallel \tilde{a} \parallel} \right).$$

To find $a \in \mathbb{R}^q$ that maximizes $F_a$, it suffices to find $\tilde{a} \in \mathbb{R}^q$ of unit length that maximizes

$$\tilde{a}^t W^{-1/2} B W^{-1/2} \tilde{a},$$

then compute $a = W^{-1/2} \tilde{a}$. To do so, we apply Theorem 2.2 to $W^{-1/2} B W^{-1/2}$. Restricting attention to orthonormal vectors of coefficients, the first eigenvector of $W^{-1/2} B W^{-1/2}$ is the desired $\tilde{a}_1$, the second eigenvector of $W^{-1/2} B W^{-1/2}$ is the desired $\tilde{a}_2$, etc. The corresponding $a_j = W^{-1/2} \tilde{a}_j$ are the discriminant coordinates of the $x_{ir}$.\footnote{In classical multivariate analysis, the discriminant coordinates are often called canonical variates, but this less-descriptive phrase is also used in canonical correlation analysis. Gnanadesikan (1977) appropriated the phrase crimcoords from a witty high school student intern, who quipped that one wouldn’t want to call such well-behaved quantities “discoords”. (The student, K. Wachter, went on to become an eminent statistician and demographer.)}

Notice that, if $W = I$, then $a_j = \tilde{a}_j$ are the first $k$ eigenvectors of $B$, hence (by a previous remark) the first $k$ eigenvectors of $T$. For this special case, the discriminant coordinates of the $x_{ir}$ coincide with the first $k$ principal components of the $x_{ir}$.

To compute discriminant coordinates directly, we rely on

**Lemma 13.1** The matrices $MN$ and $NM$ have the same nonzero eigenvalues. Furthermore, if $v$ is a nontrivial eigenvector of $MN$, then $w = Nv$ is a nontrivial eigenvector of $NM$. 


Theorem 13.1 Suppose that $(MN)v = \lambda v \neq 0$, in which case $N(MN)v = \lambda Nv$. Let $w = Nv$. Then

$$N(MN)v = (NM)Nv = (NM)w = \lambda w.$$ 

Furthermore, $Mw = MNv = \lambda v \neq 0$, hence $w \neq 0$. \hfill \Box

Now suppose that $v = \tilde{a}$ is a nontrivial eigenvector of $(W^{-1/2}B)W^{-1/2}$. By Lemma 13.1,

$$w = W^{-1/2}v = W^{-1/2}\tilde{a} = a$$

is an eigenvector of $W^{-1}B$. Moreover, the eigenproblem $W^{-1}Bv = \lambda v$ is equivalent to the generalized eigenproblem

$$Bv = \lambda Wv. \quad (13.1)$$

Thus, discriminant coordinates can be computed directly by solving (13.1), e.g., using the \texttt{R} package \texttt{geigen}.

Let $\lambda_1 \geq \cdots \geq \lambda_k > 0$ denote the strictly positive eigenvalues of (13.1) and let $v_1, \ldots, v_k$ denote the corresponding orthonormal eigenvectors. The $v_i$ are unit vectors that coincide with the desired $a_i$, so $v_i = a_i/\|a_i\|$ and we much compute $\|a_i\|$ if we are to recover $a_i$. We know that

$$v_i^t Bv = \frac{a_i^t}{\|a_i\|} \frac{B}{\|a_i\|} \frac{a_i}{\|a_i\|^2} = \frac{a_i^t B a_i}{\|a_i\|^2}$$

and that

$$a_i^t B a_i = \left(W^{-1/2}a_i\right)^t B \left(W^{-1/2}a_i\right) = \tilde{a}_i^t \left(W^{-1/2}BW^{-1/2}\tilde{a}_i\right) = \tilde{a}_i^t (\lambda_i \tilde{a}_i) = \lambda_i,$$

so $\|a_i\|^2 = \lambda_i/a_i^t B a_i$ and

$$a_i = \left(\lambda_i/a_i^t B a_i\right)^{1/2} v_i.$$

The entire construction of discriminant coordinates is summarized in Figure 13.1.

### 13.4 Linear and Quadratic Discriminant Analysis

Consider the classification rule that assigns to $x \in \mathbb{R}^q$ the label of the $\bar{x}_i$ that is nearest $x$ with respect to (squared) Mahalanobis distance, i.e.,

$$\arg \min_i \left( x - \bar{x}_i \right)^t W^{-1} \left( x - \bar{x}_i \right).$$

Replace $x_{ir}$ with $z_{ir} = W^{-1/2}x_{ir}$, in which case $x_{ir} = W^{1/2}z_{ir}$ and

$$\left( x - \bar{x}_i \right)^t W^{-1} \left( x - \bar{x}_i \right) = \left( z - \bar{z}_i \right)^t W^{1/2}W^{-1/2} W^{1/2} \left( z - \bar{z}_i \right) = \left( z - \bar{z}_i \right)^t \left( z - \bar{z}_i \right) = \|z - \bar{z}_i\|^2.$$

The transformation $z_{ir} = W^{-1/2}x_{ir}$ allows classification based on Euclidean distance. Furthermore, the within-groups sum of squares and products matrix for the $z_{ir}$ is

$$\sum_{i=1}^g \sum_{r=1}^{n_i} \left( z_{ir} - \bar{z}_i \right)^t \left( z_{ir} - \bar{z}_i \right) = \sum_{i=1}^g \sum_{r=1}^{n_i} W^{-1/2} \left( x_{ir} - \bar{x}_i \right)^t \left( x_{ir} - \bar{x}_i \right) W^{-1/2} = W^{-1/2}WW^{-1/2} = I,$$
Let $X_1, \ldots, X_g$ denote $n_i \times q$ data matrices. Set $n = n_1 + \cdots + n_g$ and

$$X^t = \begin{bmatrix} X_1^t & \cdots & X_g^t \end{bmatrix}.$$

1. Compute the centered data matrix $\tilde{X} = PX$, where $P$ is the $n \times n$ projection matrix of the form $I - ee^t/n$, then $T = \tilde{X}^t\tilde{X}$.

2. For $i = 1, \ldots, g$, compute the centered data matrices $\tilde{X}_i = P_iX_i$, where $P_i$ is the $n_i \times n_i$ projection matrix of the form $I - ee^t/n_i$, then

$$W = \tilde{X}_1^t\tilde{X}_1 + \cdots + \tilde{X}_g^t\tilde{X}_g$$

and $B = T - W$.

3. Compute $\lambda_1 \geq \cdots \geq \lambda_k > 0$, the strictly positive eigenvalues of the generalized eigenproblem $Bv = \lambda Wv$, and corresponding orthonormal eigenvectors $v_1, \ldots, v_k$.

4. Compute

$$a_i = \left(\frac{\lambda_i}{a_i^tBa_i}\right)^{1/2}v_i$$

for $i = 1, \ldots, k$, and form $A = \begin{bmatrix} a_1 & \cdots & a_k \end{bmatrix}$. Then $XA$ represents $X$ in discriminant coordinates.

Figure 13.1: Representing multivariate data in discriminant coordinates.

in which case the discriminant coordinates of the $z_{ir}$ coincide with the first $k$ principal components of the $z_{ir}$. In fact, the discriminant coordinates of the $z_{ir}$ are the $\tilde{a}$ computed in Section 13.3 and the representation of the $z_{ir}$ with respect to $\tilde{a}_1, \ldots, \tilde{a}_k$ is identical to the representation of the $x_{ir}$ with respect to $a_1, \ldots, a_k$:

$$\tilde{a}^t z = \left(W^{1/2}a\right)^t W^{-1/2}x = a^t W^{1/2} = a^tx$$

It is convenient to work with the $z_{ir}$, in which parametrization $\tilde{a}_1, \ldots, \tilde{a}_k$ are also the eigenvectors of $B$. As noted in Section 13.3, $B/(n - 1)$ is the sample covariance matrix of a data set that consists of $n_i$ copies of $\bar{z}_i$, $i = 1, \ldots, g$. It follows that $\tilde{a}_1, \ldots, \tilde{a}_k$ are the principal components of these data, and that each $\bar{z}_i - \bar{z}_.. \in \text{span}\{\tilde{a}_1, \ldots, \tilde{a}_k\}$. Furthermore, because $\tilde{a}_1, \ldots, \tilde{a}_k$ are the first $k$ principal components of the $z_{ir}$,

$$\tilde{z} = \begin{bmatrix} \tilde{a}_1^t \\ \vdots \\ \tilde{a}_k^t \end{bmatrix} (z - \bar{z}_..) = \tilde{A}^t (z - \bar{z}_..)$$

is the projection of $\tilde{z} = z - \bar{z}_.. \in \mathbb{R}^q$ into $\text{span}\{\tilde{a}_1, \ldots, \tilde{a}_k\}$. It then follows (from the Pythagorean Theorem) that the classification rule that assigns to $z \in \mathbb{R}^q$ the label of the $\bar{z}_i \in \mathbb{R}^q$ that is nearest $z$ with respect to (squared) Euclidean distance is identical to the
classification rule that assigns to \( \hat{z} \in \mathbb{R}^k \) the label of the \( \hat{z}_i \in \mathbb{R}^k \) that is nearest \( \hat{z} \) with respect to (squared) Euclidean distance. See Exercise 13.5.4.

Because the discriminant coordinates of the \( z_{ir} \) coincide with the principal components of the \( z_{ir} \), the same classification rule can be stated in the discriminant space. The discriminant and principal component spaces have different origins, but the same (unit) coordinate directions. The distances between pairs of points are identical:

\[
\| \hat{z} - \hat{z}_i \|^2 = \| (\hat{z} - \bar{z}_{..}) - (\hat{z}_i - \bar{z}_{..}) \|^2 \\
= [z - \bar{z}_{..}]^t \hat{A} \hat{A}^t [z - \bar{z}_{..}] \\
= [\hat{A}^t z - \hat{A}^t \bar{z}_i]^t [\hat{A}^t z - \hat{A}^t \bar{z}_i] \\
= \| \hat{A}^t z - \hat{A}^t \bar{z}_i \|^2.
\]

Thus, the classification rule that assigns to \( \hat{z} \in \mathbb{R}^k \) the label of the \( \hat{z}_i \in \mathbb{R}^k \) that is nearest \( \hat{z} \) with respect to (squared) Euclidean distance is identical to the classification rule that assigns to \( \hat{A}^t z \in \mathbb{R}^k \) the label of the \( \hat{A}^t \bar{z}_i \in \mathbb{R}^k \) that is nearest \( \hat{A}^t \hat{z} \) with respect to (squared) Euclidean distance.

### 13.5 Exercises

1. Let \((\pi_1, \ldots, \pi_g)\) be a probability vector. Show that \( \max\{\pi_1, \ldots, \pi_g\} \geq \pi_1^2 + \cdots + \pi_g^2 \).

2. Verify that \( T = W + C \) in the multivariate analysis of variance (manova).

3. The iris data in Exercise 1.5.1 were famously used by Sir Ronald Fisher to demonstrate linear discriminant analysis. For these data, compute \( a_1, a_2, a_3 \in \mathbb{R}^4 \), the \( g-1 = 3 \) discriminant coordinates. Construct a scatter plot with axes \( a_1^t M \) and \( a_2^t M \), using the same color scheme as in 1.5.1. Compare this representation of the iris data with the pairwise scatter plots constructed in 1.5.1.

4. Show that

\[
\| z - \bar{z}_1 \|^2 \leq \| z - \bar{z}_2 \|^2 \quad \text{iff} \quad \| \hat{z} - \hat{z}_1 \|^2 \leq \| \hat{z} - \hat{z}_2 \|^2.
\]
Chapter 14

Support Vector Machines
Part V

Appendices
Appendix A

Abstract Spaces

This appendix summarizes the basic definitions of various abstract spaces. A good source for most of this material is any book on functional analysis, e.g., [37]

Let $\Xi$ denote a set, e.g., the set of real numbers, $\mathbb{R}$, or the set of vertices in a simple weighted graph. The Cartesian product of $\Xi$ with itself, $\Xi^2 = \Xi \times \Xi$, is the set of ordered pairs of elements of $\Xi$. For example, $\mathbb{R}^2$ is the Cartesian plane. By ordered we mean, for example, that $(1, 2)$ and $(2, 1)$ are distinct elements of $\mathbb{R}^2$, i.e., distinct points in the Cartesian plane. More generally, $\Xi^k$ is the set of ordered $k$-tuples of elements of $\Xi$.

A binary operation on $\Xi$ is a function that assigns an element of $\Xi$ to each element of $\Xi \times \Xi$. A group is a nonempty set $\Xi$ equipped with a binary operation, here denoted $+$, that satisfies the following properties:

1. The associative law: $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$.
2. Existence of an identity element: there exists $\vec{0} \in \Xi$ such that $x + \vec{0} = \vec{0} + x = x$ for every $x \in \Xi$.
3. Existence of inverse elements: corresponding to each $x \in \Xi$ is an element $-x \in \Xi$ such that $x + (-x) = (-x) + x = \vec{0}$.

A group is commutative iff $x_1 + x_2 = x_2 + x_1$.

Let $\Xi$ be a commutative group and suppose that an element $\alpha x \in \Xi$ is associated with each $(\alpha, x) \in \mathbb{R} \times \Xi$ in such a way that the following properties hold:

1. $1x = x$
2. $\alpha(\beta x) = (\alpha\beta)x$
3. $(\alpha + \beta)x = \alpha x + \beta x$
4. $\alpha(x_1 + x_2) = \alpha x_1 + \alpha x_2$

We then say that $\Xi$ is a real vector space or a real linear space. If $\Xi$ is a real vector space, then an element of of $\Xi$ is a vector and $+$ is vector addition. An element of $\mathbb{R}$ is a scalar and the operation that assigns $\alpha x$ to $(\alpha, x)$ is scalar multiplication. This book assumes some familiarity with various properties of real vector spaces.

Let $L$ denote a subset of the real vector space $\Xi$. If $L$ is closed under both vector addition ($x, y \in L$ entails $x + y \in L$) and scalar multiplication ($x \in L$ entails $\alpha x \in L$), then $L$ is itself...
a real vector space and we say that $L$ is a \textit{linear subspace} of $\Xi$. Notice that a linear subspace of $\Xi$ necessarily contains $\vec{0} \in \Xi$.

Given vectors $x_1, \ldots, x_k \in \Xi$ and scalars $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$, the vector

$$
\sum_{i=1}^{k} \alpha_i x_i = \alpha_1 x_1 + \cdots + \alpha_k x_k
$$

is a \textit{linear combination} of the $x_i$. If at least one $\alpha_i \neq 0$, then we say that the linear combination is \textit{nontrivial}. The vectors are \textit{linearly dependent} iff any one of them can be written as a linear combination of the others; otherwise, they are \textit{linearly independent}. Equivalently, $x_1, \ldots, x_k$ are linearly independent iff $\alpha_1 x_1 + \cdots + \alpha_k x_k = \vec{0}$ entails $\alpha_1 = \cdots = \alpha_k = 0$.

The set of all linear combinations of $x_1, \ldots, x_k$ is their \textit{span}, denoted $\text{span}(x_1, \ldots, x_k)$. If $L = \text{span}(x_1, \ldots, x_k)$, then it is easily checked that $L$ is a linear subspace. If $x_1, \ldots, x_k$ are linearly independent, then they form a \textit{basis} for $L$ and $k$ is the \textit{dimension} of $L$.

Given $A \subseteq \Xi$ and $\vec{a} \in \Xi$, let

$$
A - \vec{a} = \{ x \in \Xi : x + \vec{a} \in A \}.
$$

The set $A$ is an \textit{affine linear subspace} or \textit{hyperplane} iff there exists $\vec{a} \in \Xi$ such that $L = A - \vec{a}$ is a linear subspace. The dimension of the affine linear subspace $A$ is the dimension of the linear subspace $L = A - \vec{a}$.

The linear combination $\alpha_1 x_1 + \cdots + \alpha_k x_k$ is a \textit{convex combination} iff each $\alpha_i \geq 0$ and $\alpha_1 + \cdots + \alpha_k = 1$. The convex combination defined by $\alpha_i = 1/k$,

$$
\bar{x} = \frac{1}{k} \sum_{i=1}^{k} x_i = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{k} x_i
$$

is the \textit{centroid} of $x_1, \ldots, x_k$. The set of convex combinations of $x_1$ and $x_2$ is the line segment that connects them, $[x_1, x_2]$, and the centroid $(x_1 + x_2)/2$ is the midpoint of $[x_1, x_2]$. A set $K \subseteq \Xi$ is \textit{convex} iff $x_1, x_2 \in K$ entails $[x_1, x_2] \subseteq K$. If $K$ is a convex set, then every convex combination of $x_1, \ldots, x_k \in K$ lies in $K$. Notice that hyperplanes are convex sets.

Let $d : \Xi \times \Xi \to \mathbb{R}$ denote a real-valued function with domain $\Xi \times \Xi$. The function $d$ is a \textit{distance function} or \textit{metric} on $\Xi$ iff $d$ satisfies the following conditions:

1. $d(x_1, x_2) \geq 0$
2. $d(x_1, x_2) = 0$ iff $x_1 = x_2$
3. $d(x_1, x_2) = d(x_2, x_1)$
4. $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$

The fourth condition is known as the \textit{triangle inequality}. We say that the pair $(\Xi, d)$ is a \textit{metric space}.

Let $\Xi$ denote a real linear space. A function $c : \Xi \to \mathbb{R}$ is a \textit{norm} on $\Xi$ iff it satisfies the following properties:

1. $c(x) \geq 0$
2. $c(x) = 0$ iff $x = \vec{0}$
3. \( c(\alpha x) = \alpha c(x) \)

4. \( c(x_1 + x_2) \leq c(x_1) + c(x_2) \)

We usually write \( c(x) = \|x\| \) and say that the pair \((\Xi, \| \cdot \|)\) is a normed linear space. The function \( d : \Xi \times \Xi \to \mathbb{R} \) defined by

\[
d(x_1, x_2) = \|x_1 - x_2\|
\]

is a distance function; hence, a normed linear space is necessarily a metric space.

Let \( \Xi \) denote a real linear space. A function \( b : \Xi \times \Xi \to \mathbb{R} \) is an inner product on \( \Xi \) iff it satisfies the following properties:

1. \( b(x, x) \geq 0 \)
2. \( b(x, x) = 0 \) iff \( x = 0 \)
3. \( b(x_1, x_2) = b(x_2, x_1) \)
4. \( b(x_1, x_2 + x_3) = b(x_1, x_2) + b(x_1, x_3) \)
5. \( b(x_1, \alpha x_2) = \alpha b(x_1, x_2) \)

We usually write \( b(x_1, x_2) = \langle x_1, x_2 \rangle \) and say that the pair \((\Xi, \langle \cdot, \cdot \rangle)\) is an inner product space. The function \( c : \Xi \to \mathbb{R} \) defined by

\[
c(x) = \langle x, x \rangle^{1/2}
\]

is a norm on \( \Xi \); hence, an inner product space is necessarily a normed linear space. The Cauchy-Schwartz inequality states that

\[
|\langle x_1, x_2 \rangle| \leq \|x_2\| \|x_2\|,
\]

with equality iff \( x_1 = \alpha x_2 \).
Appendix B
Matrices

A real \( m \times n \) matrix \( A \) is a rectangular array of real numbers with \( m \) rows and \( n \) columns.\(^1\) We index the entries of \( A \) by identifying the row \( i \) and the column \( j \) in which the entry appears, writing \( A = [a_{ij}] \). We define matrix addition by \( A + B = [a_{ij} + b_{ij}] \) and scalar multiplication by \( \alpha A = [\alpha a_{ij}] \). The transpose of \( A \) is the \( n \times m \) matrix \( A^t = [a_{ji}] \). If \( m = n \) and \( A^t = A \), then we say that \( A \) is symmetric.

A column vector is an \( k \times 1 \) matrix and a row vector is a \( 1 \times k \) matrix. When representing an ordered \( k \)-tuple of real numbers, \( a = (\alpha_1, \ldots, \alpha_k) \), in matrix notation we adhere to a useful convention and write \( a \) as a column vector:

\[
a = \begin{bmatrix}
\alpha_1 \\
\vdots \\
\alpha_k
\end{bmatrix}
\]

If we wish to store the coordinates of \( a \) in a row vector, then we write

\[
\begin{bmatrix}
\alpha_1 \\
\cdots \\
\alpha_k
\end{bmatrix} = a^t.
\]

We repeat: in matrix notation, \( k \)-tuples are always represented as column vectors.

The dot product between \( k \)-tuples \( a = (\alpha_1, \ldots, \alpha_k) \) and \( b = (\beta_1, \ldots, \beta_k) \) is the real number \( \langle a, b \rangle = \alpha_1 \beta_1 + \cdots + \alpha_k \beta_k \).

We use the dot product to define matrix multiplication. If \( A \) is \( m \times k \) and \( B \) is \( k \times n \), then the product \( AB = [c_{ij}] \) is the \( m \times n \) matrix defined by setting \( c_{ij} \) equal to the dot product of row \( i \) of \( A \) and column \( j \) of \( B \). Using matrix notation, we can then write the dot product as a special case of matrix multiplication: \( \langle a, b \rangle = a^t b \). It is easily checked that matrix multiplication is associative, i.e., \( A(BC) = (AB)C \), and that \( (AB)^t = B^t A^t \). If \( A \) and \( B \) are symmetric, then \( (AB)^t = B^t A^t = BA \); in general, however, \( AB \) and \( BA \) may differ. The trace of a square matrix is the sum of its diagonal entries and it is easily verified that

\[
\text{trace}(AB) = \text{trace}(BA).
\]

The diagonal entries of a square matrix \( A = [a_{ij}] \) are the entries \( a_{ii} \). A square matrix is diagonal iff its nondiagonal entries vanish.

\(^1\)This book is concerned exclusively with real matrices.
Appendix C

Graphs

Informally, a graph is a pair $G = (V, E)$ of vertices and edges.

More formally, let $V$ denote a nonempty finite set, the elements of which are the vertices of $G$. Let $E$ denote a finite family of pairs of vertices, the edges of $G$. If $E$ comprises unordered pairs of vertices, then $G$ is undirected and we write the edge between vertices $v_1$ and $v_2$ as $v_1 \leftrightarrow v_2$. If $E$ comprises ordered pairs of vertices, then $G$ is directed and we write the edge from vertex $v_1$ to vertex $v_2$ as $v_1 \rightarrow v_2$.

We say that $G$ is an undirected or directed graph iff the elements of $E$ are distinct. If an unordered or ordered pair of vertices appears more than once, then we say that $G$ is a multigraph. If the elements of $E$ are distinct, then we can indicate the elements of $E$ by the adjacency matrix $A = [a_{ij}]$ of $G$. If $G$ is undirected, then $a_{ij} = a_{ji} = 1$ iff $v_i \leftrightarrow v_j \in E$. Otherwise, $a_{ij} = 0$. If $G$ is directed, then $a_{ij} = 1$ iff $v_i \rightarrow v_j \in E$. Otherwise, $a_{ij} = 0$.

An edge of the form $v_1 \leftrightarrow v_1$ or $v_1 \rightarrow v_1$ is a loop. If $E$ contains no loops, then $G$ is simple. Our primary concern will be with simple undirected graphs; occasionally we will also consider simple directed graphs.

If $G$ is undirected, then a walk from vertex $v_i$ to vertex $v_j$ is a finite sequence of edges of the form

\[ v_i \leftrightarrow v_1, v_1 \leftrightarrow v_2, \ldots, v_{m-1} \leftrightarrow v_m, v_m \leftrightarrow v_j, \]

abbreviated

\[ v_i \leftrightarrow v_1 \leftrightarrow v_2 \leftrightarrow \cdots \leftrightarrow v_m \leftrightarrow v_j. \]

If $G$ is directed, then a walk is of the form

\[ v_i \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_m \rightarrow v_j. \]

We say that the walk starts at $v_i$ and stops at $v_j$. The vertices $v_1, \ldots, v_m$ are intermediate between $v_i$ and $v_j$.

A path is a walk with distinct intermediate vertices, none of which are $v_i$ or $v_j$. Notice that any graph admits infinitely many walks; however, because $V$ is finite, the number of possible paths in $G$ is finite. A circuit is a path for which $v_i = v_j$.

Suppose that a walk starts at $v_i$ and stops at $v_j$. If the walk contains a circuit, then we can shorten the walk by replacing the circuit with the vertex at which the vertex starts and stops. It follows that there must be a path from $v_i$ to $v_j$. A graph is connected iff there is a path from any $v_i \in V$ to any $v_j \in V$.

A graph $G = (V, E)$ is weighted iff we have associated with each edge in $E$ a nonnegative real number, the edge weight $w_{ij} \geq 0$. If $G$ is undirected, then $w_{ij} = w_{ji}$; if $G$ is directed,
then $w_{ij}$ and $w_{ji}$ may differ. Typically we require $w_{ij} > 0$ and reserve $w_{ij} = 0$ to indicate that no edge connects vertices $i$ and $j$. We can then store the edge weights in the $n \times n$ weight matrix $W = [w_{ij}]$. Here are two obvious properties of $W$:

1. If $G$ is a simple graph, then each $w_{ii} = 0$, i.e., $W$ is hollow.

2. If $G$ is an undirected graph, then $W$ is symmetric.

We can extend the concept of a weight matrix to unweighted graphs by assigning identical weights of $w_{ij} = 1$ to each edge of the graph. With this convention, the weight matrix of an unweighted graph $G$ is the adjacency matrix of $G$.

The tools of matrix analysis are of enormous value in studying graphs. Let $G$ be an undirected simple graph with weight matrix $W = [w_{ij}]$. The row sums of $W$ are

$$t_{ii} = \sum_{k=1}^{n} w_{ik} = \sum_{k \neq i} w_{ik},$$

the sum of the weights of the edges that connect vertex $i$ to other vertices. Because $W$ is symmetric, its row sums are also its column sums. For $i \neq j$, let $t_{ij} = 0$. The $n \times n$ diagonal matrix $T = [t_{ij}]$ contains the row/column sums of $W$ on its diagonal. In matrix form, $T = \text{diag}(W)$. The matrix $T$ is the degree matrix of $G$ and $t_{ii}$ is the degree of vertex $i$.

The combinatorial Laplacian matrix of $G$ is the symmetric matrix $L = [\ell_{ij}] = T - W$. By construction, $\ell_{ii} = t_{ii}$, $\ell_{ij} = -w_{ij}$ for $i \neq j$, and $Le = 0$. Thus, $e$ is an eigenvector of $L$ with eigenvalue 0 and $L$ is rank deficient.

Let $e_i \in \mathbb{R}^n$ denote the unit vector in coordinate direction $i$, e.g., $e_2 = (0, 1, 0, \ldots, 0)$. For $i \neq j$, the rank-1 matrix $E_{ij} = [e_{ij}] = (e_i - e_j)(e_i - e_j)^t$ has exactly four nonzero entries: $e_{ii} = e_{jj} = 1$ and $e_{ij} = e_{ji} = -1$; hence, we can write

$$L = \sum_{i<j} w_{ij}E_{ij}.$$ 

Noting that each $w_{ii} \geq 0$ and that each $E_{ij}$ is positive semidefinite, we deduce that $L$ is positive semidefinite.

Suppose that $G$ is connected, so that each $t_{ii} > 0$. Let $a = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$ and suppose that $a$ is an eigenvector of $L$ with eigenvalue 0. Then

$$2La = \sum_{i,j} w_{ij} (e_i - e_j)(e_i - e_j)^t a = \sum_{i,j} w_{ij} (\alpha_i - \alpha_j)(e_i - e_j) = 0$$

and it follows that, for $r = 1, \ldots, n$,

$$0 = \sum_{j \neq r} w_{rj} (\alpha_r - \alpha_j) - \sum_{i \neq r} w_{ir} (\alpha_i - \alpha_r) = 2 \sum_{j \neq r} w_{rj} (\alpha_r - \alpha_j) = 2t_{rr}\alpha_r - 2 \sum_{j \neq r} w_{rj}\alpha_j.$$ 

Rearranging terms, we deduce that each $\alpha_r$ is a convex combination of the other $\alpha_j$:

$$\alpha_r = \sum_{j \neq r} \frac{w_{rj}}{t_{rr}} \alpha_j.$$
We claim that $\alpha_1 = \cdots = \alpha_n$, for suppose not. Let $\bar{\alpha} = \max(\alpha_1, \ldots, \alpha_n)$, let $I_1 = \{i : \alpha_i = \bar{\alpha}\}$, let $I_2 = \{i : \alpha_i < \bar{\alpha}\}$, and suppose that $I_2$ is not empty. Because $G$ is connected, there must be an edge between vertices $r$ and $s$ for some $r \in I_1$ and $s \in I_2$. But in that case,

$$\sum_{j \neq r} \frac{w_{rj}}{t_{rr}} \alpha_j < \alpha_r$$

because $\alpha_s < \alpha_r$ and $w_{rs} > 0$. Thus, we have the following well-known result.

**Theorem C.1** Let $G$ be a connected undirected simple graph with combinatorial Laplacian matrix $L$. Let $\lambda_n \geq \cdots \geq \lambda_1$ denote the eigenvalues of $L$. Then $\lambda_1 = 0$ and the only eigenvectors of $L$ with eigenvalue $\lambda_1$ are of the form $\alpha e$, hence $\lambda_2 > 0$ and the other eigenvectors of $L$ are orthogonal to $e$. 

Bibliography


